M 2016

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

CLASSIFICAÇÕES GEOMECÂNICAS APLICADAS A TALUDES ROCHOSOS

CONTRIBUIÇÃO PARA A SUA UTILIZAÇÃO COM APLICAÇÕES MÓVEIS.

DIOGO MIGUEL GOMES ASSIS

DISSERTAÇÃO DE MESTRADO APRESENTADA
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM
ENGENHARIA CIVIL

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

CLASSIFICAÇÕES GEOMECÂNICAS APLICADAS A TALUDES ROCHOSOS

Contribuição para a sua utilização com aplicações móveis

DIOGO MIGUEL GOMES ASSIS

Dissertação submetida para satisfação parcial dos requisitos do grau de

MESTRE EM ENGENHARIA CIVIL — ESPECIALIZAÇÃO EM GEOTECNIA

Orientador: Professor Doutor José Eduardo Tavares Quintanilha de Menezes

SETEMBRO DE 2016

MESTRADO INTEGRADO EM ENGENHARIA CIVIL 2015/2016

DEPARTAMENTO DE ENGENHARIA CIVIL

Tel. +351-22-508 1901

Fax +351-22-508 1446

✉ miec@fe.up.pt

Editado por

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rua Dr. Roberto Frias

4200-465 PORTO

Portugal

Tel. +351-22-508 1400

Fax +351-22-508 1440

✉ feup@fe.up.pt

🌍 http://www.fe.up.pt

Aos meus pais e irmã

Não sou da altura que me vêem, mas sim da altura que os meus olhos podem ver.

Fernando Pessoa

AGRADECIMENTOS

A realização desta dissertação não teria sido possível sem a contribuição fundamental de algumas pessoas, motivo pelo qual devo reconhecer a ajuda recebida.

Desta forma, gostaria de agradecer ao meu orientador, Professor Doutor José Quintanilha de Menezes, pela motivação e interesse demonstrado na realização deste trabalho, nomeadamente nas sugestões apresentadas para o desenvolvimento da aplicação.

Às amizades criadas ao longo dos últimos anos, particularmente à Rita, ao Luís, à Adriana e ao Francisco. A estes deixo uma palavra de apreço por estarem sempre presentes em todos os momentos da minha vida académica.

Por fim, agradeço à minha família por todo o apoio dado ao longo do meu percurso. Ao meu pai pelo incentivo em alcançar os objetivos propostos; à minha mãe por todo o carinho dado e por me ter proporcionado as melhores condições durante o meu percurso académico; à minha irmã, pela inspiração que é e sempre será e por ser um dos suportes da minha vida.

RESUMO

Com a realização deste trabalho pretendeu-se desenvolver uma ferramenta de cálculo, destinada a dispositivos móveis, que permita avaliar a qualidade do talude rochoso, através da classificação geomecânica *Slope Mass Rating* (SMR).

Numa primeira fase é feita uma contextualização das classificações geomecânicas existentes, na qual, dada a sua relevância no tema, se detalhou os parâmetros avaliados na classificação *Rock Mass Rating* (RMR). Devido à sua reduzida aplicabilidade em taludes rochosos estudou-se o sistema SMR, que consiste numa adaptação da classificação anterior para a análise pretendida.

Previamente ao desenvolvimento da aplicação, foi efetuado um levantamento das aplicações *Android* existentes na *Google Play Store* relacionadas com classificações geomecânicas, nomeadamente aplicadas a taludes, de modo a identificar as limitações existentes nas mesmas e desenvolver uma aplicação original.

Recorrendo ao programa *Android Studio*, é desenvolvida a aplicação *Slope Mass Rating*, com o objetivo de auxiliar a utilização desta classificação em obra sugerindo o suporte a aplicar. Esta inclui a utilização das capacidades do dispositivo móvel para definir as orientações do talude e da descontinuidade, incorporando, assim, uma bússola de geólogo. Este desenvolvimento culmina com a publicação da aplicação na *Google Play Store*.

PALAVRAS-CHAVE: Classificações Geomecânicas, Taludes Rochosos, RMR, SMR, *Android*, Aplicação Móvel.

ABSTRACT

The focus of this work was to develop a calculation tool for mobile devices, that allows the evaluation of rock slopes using the geomechanical classification Slope Mass Rating (SMR).

The first step was to address the existing geomechanical classifications, more specifically the Rock Mass Rating (RMR) and its parameters. Due to the reduced applicability of RMR to rock slopes, the focus was given to the study of the SMR, that is an adaptation of RMR to the scope of this project.

Before the implementation of the tool, Google Play Store was used to perform a rigorous search on existing Android apps related to geomechanical classifications as well as their applicability to slopes and liabilities, in order to develop a unique tool.

Through the use of the Android Studio, the Slope Mass Rating app was developed to facilitate the use of this classification on site. The app suggests the most appropriate support method and takes advantage of the mobile devices features, defining slope and discontinuity orientations, by using the geological compass. The resulting app has been made available on the Google Play Store.

KEYWORDS: Geomechanical Classification, Rock Slope, RMR, SMR, Android, Mobile App.

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS DOS ANEXOS

ANEXO I

ANEXO II

ANEXO III

ANEXO IV

ANEXO V

ANEXO VI

ANEXO VII

ANEXO VIII

ANEXO IX

ANEXO X

ANEXO XI

ANEXO XII

ANEXO XIII

ANEXO XIV

ÍNDICE TABELAS

SÍMBOLOS, ACRÓNIMOS E ABREVIATURAS

SÍMBOLOS

A – Ângulo auxiliar no cálculo de F_1	[º]
B – Ângulo auxiliar no cálculo de F_2	[º]
C – Ângulo auxiliar no cálculo de F_3	[º]
d – Espaçamento entre descontinuidades	[mm]
E – Módulo de deformabilidade	[GPa]
F_1 – Fator de ajustamento 1	
F_2 – Fator de ajustamento 2	
F_3 – Fator de ajustamento 3	
F_4 – Fator de ajustamento 4	
I_{50} – Índice carga pontual para provetes com 50 mm de diâmetro	
J_v – Índice volumétrico	
N – Carga máxima aplicada	[kN]
α_j – Dip direction da descontinuidade	[º]
α_s – Dip direction do talude	[º]
β_j – Dip da descontinuidade	[º]
β_s – Dip do talude	[º]
σ_c – Resistência à compressão uniaxial	[MPa]

ABREVIATURAS

App – Aplicação

Fig – Figura

ID – Identidade

ACRÓNIMOS

API – Application Programming Interface

GSI – Geological Strength Index

IDE – Integrated Development Environment

iOS – Iphone Operating System

RMR – Rock Mass Rating

RMR_b – Rock Mass Rating básico

RQD – Rock Quality Designation

SMR – Slope Mass Rating

XML – Extensible Markup Language

1
INTRODUÇÃO

1.1. ENQUADRAMENTO E OBJETIVOS

As classificações geomecânicas são uma importante ferramenta de caraterização do maciço rochoso, permitindo uma avaliação da qualidade do mesmo para fins de Engenharia. Assim, face à necessidade de caracterizar o maciço em obras subterrâneas, surgem classificações como o *Rock Mass Rating*, o *Q system* e o *Geological Strength Index*. Estas resultam da análise de diversos casos de estudo realizados pelos respetivos autores, avaliando, essencialmente, os seguintes parâmetros:

- Resistência da rocha;
- Índice RQD (*Rock Quality Designation*);
- Caracterização das descontinuidades;
- Presença de água;
- Estado de tensão.

Deste modo, através da observação direta do maciço e da realização de pequenos ensaios, são atribuídos índices de qualidade, a partir dos quais se estimam parâmetros de resistência e de deformabilidade, bem como o suporte recomendado a aplicar. Dado o carácter empírico que estas classificações apresentam é necessário analisar o resultado obtido com elevado sentido crítico, uma vez que estas apresentam limitações e não substituem a avaliação do maciço por métodos mais precisos. No entanto, devido aos resultados obtidos revelarem grande utilidade, nomeadamente no dimensionamento de suporte em túneis, as classificações existentes foram especificamente adaptadas para minas, taludes e fundações.

A classificação geomecânica SMR surge, assim, como uma expansão do sistema RMR, avaliando a qualidade de taludes rochosos. Para tal, além dos parâmetros associados à qualidade do maciço rochoso, são analisados os possíveis modos de instabilidade, definindo-se diferentes fatores de ajustamento associados à orientação das descontinuidades e ao método de escavação.

Assim, a realização deste trabalho teve como principal objetivo o desenvolvimento de uma ferramenta de cálculo, destinada a dispositivos móveis, que auxilie a implementação de classificações geomecânicas em taludes rochosos em obra.

1.2. ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está organizada em 5 capítulos, sendo este primeiro referente à introdução, no qual é feita uma abordagem geral ao tema bem como o delineamento dos objetivos propostos para a realização deste trabalho.

No capítulo 2 é apresentada a classificação geomecânica RMR, destacando os vários parâmetros avaliados e as possíveis conclusões face ao índice obtido. É ainda descrita a classificação adaptada para taludes rochosos, o SMR, incluindo as propostas de cálculo apresentadas por diferentes autores.

No capítulo 3 são analisadas as aplicações disponíveis na *Google Play Store* referentes ao cálculo de classificações geomecânicas, nomeadamente no contexto de taludes rochosos.

O capítulo 4 apresenta o desenvolvimento de uma ferramenta de cálculo da classificação SMR, destinada a dispositivos móveis, utilizando o programa *Android Studio*. É explicado o funcionamento do programa utilizado assim como é descrita detalhadamente a aplicação elaborada e apresentadas algumas decisões tomadas ao longo da conceção da mesma. No final deste capítulo encontra-se disponível o *link* para *download* da versão atual da aplicação na *Google Play Store*.

Por último, no capítulo 5 são abordadas as conclusões deste trabalho e apresentadas algumas propostas para desenvolvimentos futuros.

2

CLASSIFICAÇÕES GEOMECÂNICAS APLICADAS A TALUDES ROCHOSOS

2.1. INTRODUÇÃO ÀS CLASSIFICAÇÕES GEOMECÂNICAS

2.1.1. ENQUADRAMENTO HISTÓRICO

As classificações geomecânicas resultam da necessidade de caracterizar o maciço rochoso, auxiliando a resolução de problemas de engenharia. Desenvolvidas empiricamente, as várias classificações existentes têm como princípio a observação direta do maciço rochoso e a realização de ensaios, *in situ* ou em laboratório, resultando na atribuição de um índice de qualidade. Destas, apesar de não existir uma escolha unânime, têm sido preferencialmente adotadas a classificação *Rock Mass Rating* (RMR) (Bieniawski, 1989), o sistema Q (Barton *et al.*, 1974) e o *Geological Strength Index* (GSI) (Hoek, 2007).

A introdução das classificações geomecânicas revelou ter bastante utilidade, já que permite uma rápida caraterização do maciço rochoso, estimam parâmetros de resistência e de deformabilidade, bem como a sugestão do suporte a aplicar. No entanto, a utilização destas classificações deve ser criteriosa, uma vez que não substituem os processos de cálculo analítico. Como tal, tendo em conta que cada sistema de classificação atribui maior importância a diferentes parâmetros, é recomendada a utilização de pelo menos dois métodos comparando, assim, os resultados obtidos (Hoek, 2007).

Sendo uma das referências das classificações geomecânicas, devido à sua grande adoção na caraterização de maciços rochosos em diferentes obras de engenharia, o processo de cálculo da classificação RMR será apresentado de seguida.

2.1.2. *ROCK MASS RATING*

O sistema RMR representa uma classificação geomecânica de maciços rochosos, indicando um índice de qualidade do mesmo e uma proposta de suporte a executar, sendo aplicável a vários tipos de escavação: túneis, minas, taludes e fundações.

Introduzido por Bieniawski em 1973 (Bieniawski, 1973) com base na sua experiência na escavação de túneis em maciços rochosos, a classificação RMR foi sofrendo atualizações na forma de cálculo e no suporte proposto, tendo sido publicado em 1989 (Bieniawski, 1989) a última versão que se resume à avaliação dos seguintes parâmetros:

- Resistência da rocha intacta;
- *Rock Quality Designation*;
- Espaçamento das descontinuidades;
- Condições das descontinuidades;

- Presença de água;
- Orientação das descontinuidades.

A classificação é realizada mediante a atribuição de um peso a cada parâmetro em estudo, correspondendo o somatório dos diferentes pesos ao índice RMR. Este pode variar entre 0 e 100, sendo o seu valor proporcional à qualidade do maciço rochoso.

Através da análise de 351 casos de estudo, Bieniawski procurou relacionar o valor deste índice com parâmetros de resistência, como coesão ou ângulo de atrito, e o tempo médio de auto sustentação duma cavidade subterrânea (Tabela 1) (Bieniawski, 1989). Na Fig. 1 é exemplificada uma das correlações empíricas, relacionando o tempo de auto sustentação, em minas e túneis, e o vão existente, para diferentes valores do índice RMR.

Tabela 1 – Classificação RMR (Bieniawski, 1989)

Classe	Descrição	Intervalo RMR	Coesão (kPa)	Ângulo de atrito	Tempo médio de auto sustentação
I	Muito bom	100 – 81	> 400	> 45º	20 anos para 15 metros de vão
II	Bom	80 – 61	300 – 400	35º – 45º	1 ano para 10 metros de vão
III	Médio	60 – 41	200 – 300	25º – 35º	1 semana para 5 metros de vão
IV	Mau	40 – 21	100 – 200	15º – 25º	10 horas para 2,5 metros de vão
V	Muito mau	< 20	< 100	< 15º	30 minutos para 10 metros de vão

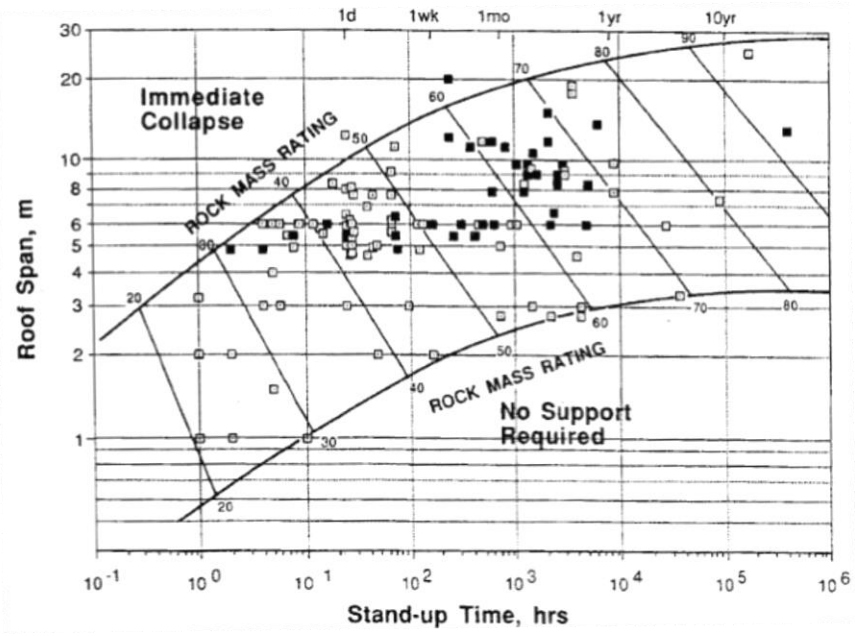


Fig. 1 – Relação entre tempo de auto-sustentação e o vão existente em minas e túneis (Bieniawski, 1989)

Através do índice RMR obtido é ainda possível correlacionar com o módulo de deformabilidade (E) através das expressões (1) e (2), propostas por Bieniawski e Serafim e Pereira, respetivamente (Bieniawski, 1989).

$$E = 2RMR - 100, \text{ para } RMR > 50 \quad (1)$$

$$E = 10^{\frac{RMR-10}{40}} \quad (2)$$

Seguidamente serão analisados os diferentes parâmetros utilizados na avaliação da classificação RMR.

2.1.2.1. Resistência da rocha intacta

Sendo a classificação RMR tipicamente usada em locais no subsolo, como minas e túneis, é fundamental conhecer a resistência à compressão uniaxial. Esta característica mecânica pode ser obtida através da recolha de amostras, e posterior análise em laboratório, ou através de ensaios *in situ*.

Em laboratório, é possível realizar um ensaio de resistência à compressão uniaxial (Fig. 2), fazendo-se comprimir uma amostra cilíndrica através das suas bases, onde, mediante o valor da carga máxima (N) imposta e do diâmetro do provete, é determinada a resistência em estudo (3).

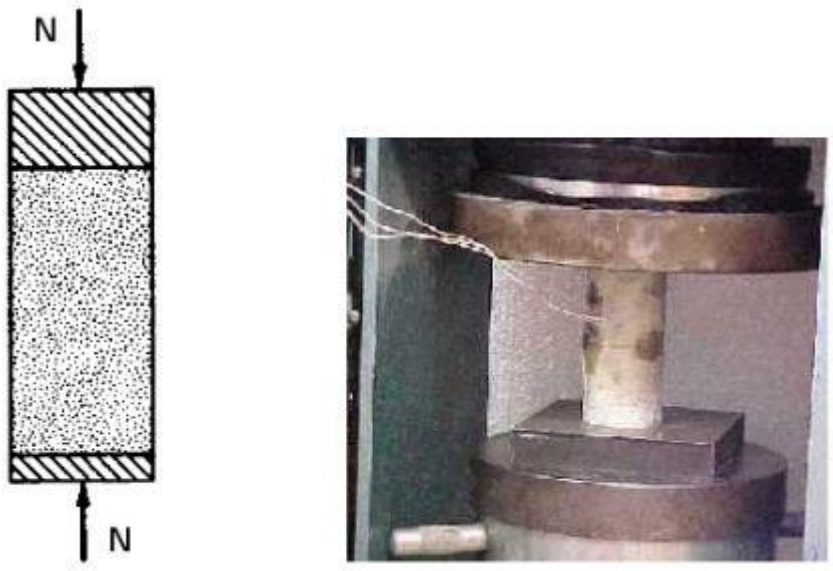


Fig. 2 – Ensaio laboratorial de resistência à compressão uniaxial

$$\sigma_c = \frac{N}{\pi r^2} \quad (3)$$

Devido à complexidade exigida no ensaio à compressão uniaxial pela utilização de uma prensa hidráulica é possível, em alternativa, recorrer a um ensaio mais simples, o ensaio de carga pontual (Fig. 3), que traduz o valor da resistência à compressão uniaxial de forma aproximada. Este ensaio, também

conhecido como *Point Load Test*, consiste em comprimir o provete com uma carga P, através de duas ponteiras metálicas, obtendo-se o índice de carga pontual (4) (Wyllie e Mah, 2005).

Devido à complexidade exigida no ensaio à compressão uniaxial é possível, em alternativa, recorrer a um ensaio mais simples, que traduz o valor da resistência à compressão uniaxial de forma aproximada, o ensaio de carga pontual (Fig. 3). Este ensaio, também conhecido como *Point Load Test*, consiste em comprimir o provete na face lateral com uma carga P, através de duas ponteiras metálicas, obtendo-se o índice de carga pontual

$$I_{50} = \frac{P}{D^2} \tag{4}$$

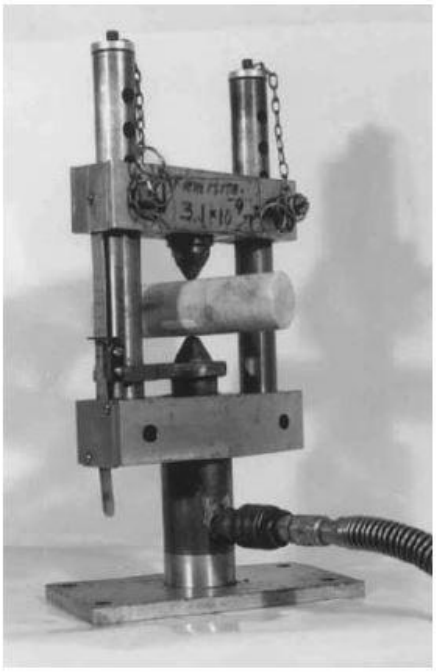


Fig. 3 – Ensaio laboratorial de carga pontual (Wyllie e Mah, 2005)

Conhecido o índice de carga pontual é possível determinar a resistência à compressão uniaxial da rocha, multiplicando o valor do índice obtido por um fator a (5), que usualmente está compreendido entre 20 e 25. Através de uma regressão linear desenvolvida empiricamente por Bieniawski (1974) (Fig. 4), este verificou que o valor de a se aproxima de 24. No entanto, este ensaio é inválido para índices de carga pontual inferiores à unidade, sendo necessário recorrer ao ensaio de compressão uniaxial para determinar a sua resistência.

$$\sigma_c = a \times I_{50} \tag{5}$$

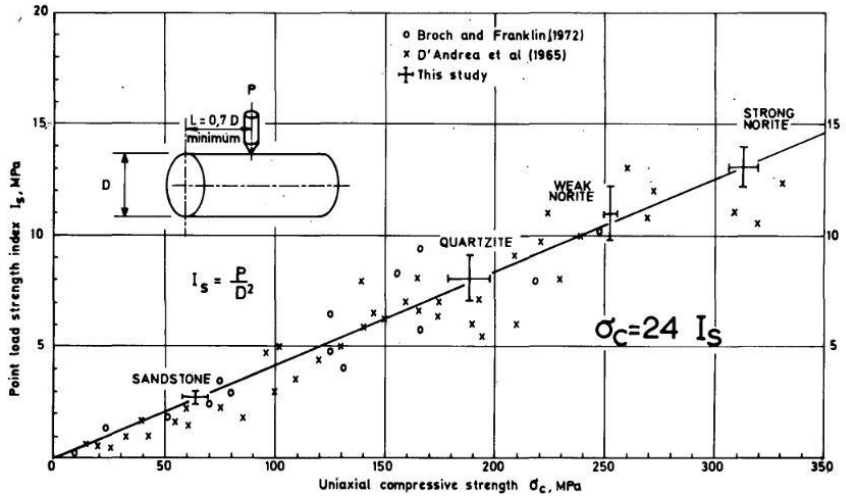


Fig. 4 – Relação entre o índice carga pontual e a resistência à compressão uniaxial (Bieniawski, 1974)

Na indisponibilidade de recolha de amostras para posterior análise em laboratório, e uma vez que uma das vantagens do sistema RMR é a possibilidade de cálculo em obra, é possível determinar a resistência à compressão uniaxial *in situ*, recorrendo ao martelo de *Schmidt*, avaliando a dureza da superfície da rocha onde, por relação com o peso volúmico associado ao tipo de rocha em estudo, e obtendo uma estimativa da resistência à compressão uniaxial do maciço (Fig. 5) (Hoek, 2007).

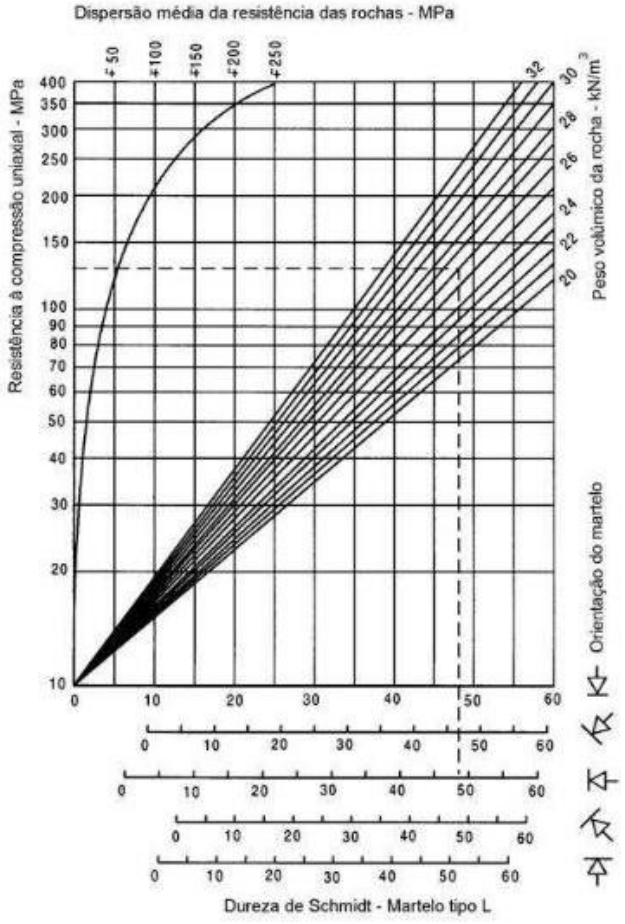


Fig. 5 – Estimativa da resistência à compressão uniaxial (Hoek, 2007)

A evolução do peso da resistência da rocha intacta é apresentada na Tabela 2, sendo atribuído determinado peso a um intervalo de valores de resistência. Esta tabela resulta da simplificação da Fig. 6, na qual é relacionado o peso com a resistência à compressão uniaxial. De forma a incluir na tabela a variação do índice de carga pontual, a resistência à compressão uniaxial foi determinada pela expressão (4).

Tabela 2 – Ponderação da resistência da rocha intacta na avaliação do RMR (Bieniawski, 1989)

Resistência da rocha intacta	I_{50} [Mpa]	Usar compressão uniaxial			1 – 2	2 – 4	4 – 10	> 10
	σ_c [Mpa]	< 1	1 – 5	5 – 25	25 – 50	50 – 100	100 – 250	> 250
Peso		0	1	2	4	7	12	15

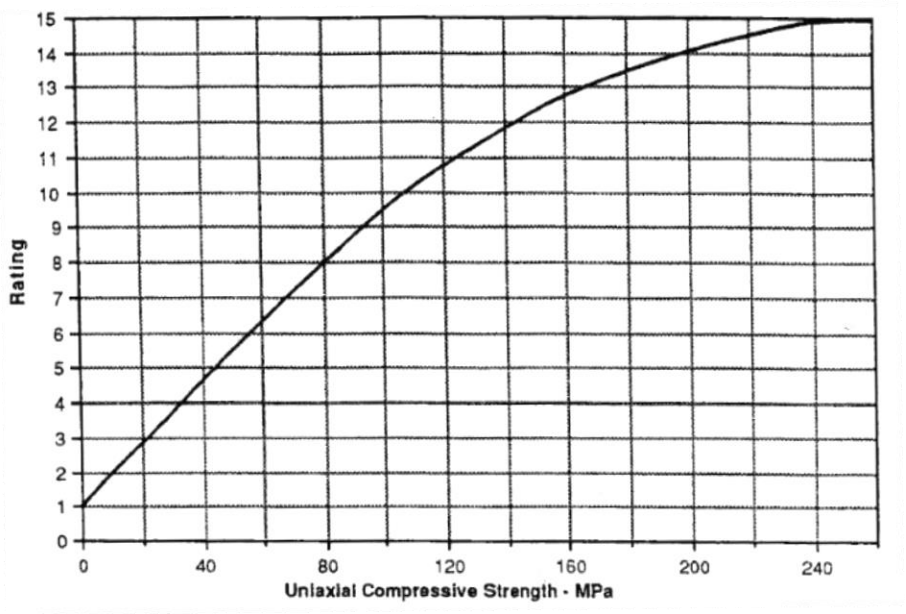


Fig. 6 – Ponderação da resistência à compressão uniaxial na avaliação do RMR (Bieniawski, 1989)

2.1.2.2. *Rock Quality Designation*

O *Rock Quality Designation* (RQD) é um sistema de classificação que avalia, em percentagem, o grau de fracturação do maciço rochoso. O seu cálculo baseia-se no quociente entre a soma do comprimento dos fragmentos da amostra superiores a 10 centímetros e o comprimento total da amostra, tal como exemplificado na Fig. 7 (Deere e Deere, 1988).

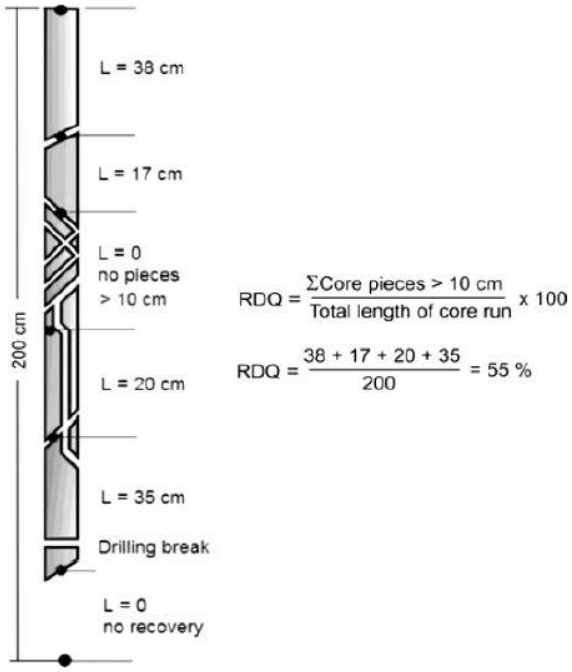


Fig. 7 – Exemplificação da avaliação do índice RQD (Hoek, 2007)

Através do valor deste índice é possível descrever qualitativamente o estado de fracturação do maciço rochoso, estando apresentado na Tabela 3 as classificações atribuídas pelos autores. Esta escala abrange uma qualidade do maciço desde muito fraco até excelente, mediante um índice RQD inferior a 25 % ou superior a 90 %, respetivamente.

Tabela 3 – Classificação da qualidade do maciço em função do RQD (Deere e Deere, 1988)

RQD [%]	Qualidade do maciço
< 25	Muito fraco
25 – 50	Fraco
50 – 75	Razoável
75 – 90	Bom
90 – 100	Excelente

Na impossibilidade de recolha de amostras, o índice RQD pode ser calculado através da correlação empírica proposta por Palmstrom (1982) (expressão 6 e 7), onde J_v representa o número de descontinuidades por metro cúbico.

$$RQD = 115 - 3{,}3\,J_v \text{ , para } J_v > 4{,}5 \quad (6)$$

$$RQD = 100 \text{ , para } J_v \leq 4{,}5 \quad (7)$$

De forma análoga ao parâmetro analisado anteriormente, Bieniawski esboçou a relação entre o peso e o índice RQD (Fig. 8), resumindo a informação na Tabela 4 para uma utilização mais simplificada.

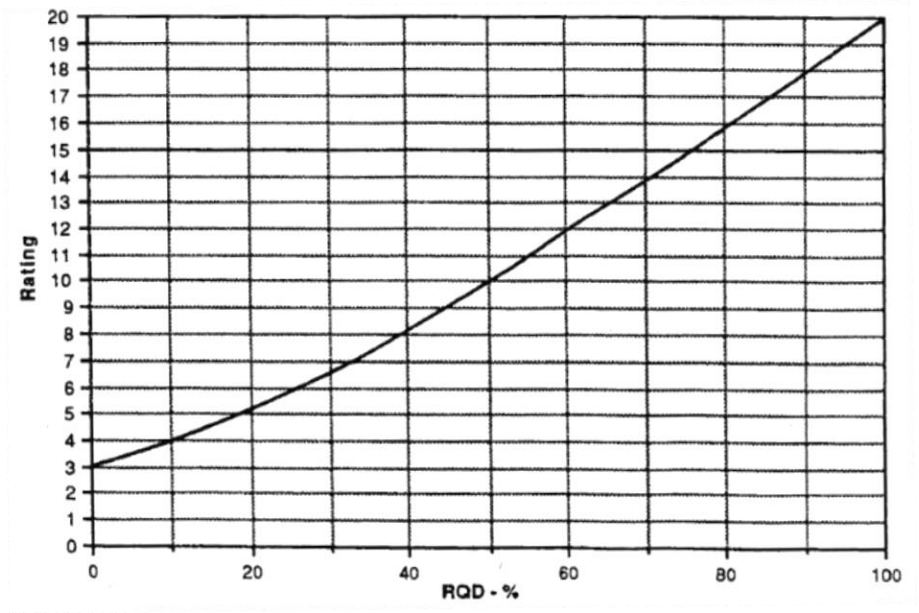


Fig. 8 – Ponderação do índice RQD na avaliação do RMR (Bieniawski, 1989)

Tabela 4 – Ponderação do índice RQD na avaliação do RMR (Bieniawski, 1989)

RQD [%]	< 25	25 – 50	50 – 75	75 – 90	90 – 100
Peso	3	8	13	17	20

2.1.2.3. Espaçamento entre descontinuidades

O terceiro parâmetro a avaliar no sistema RMR é referente ao espaçamento entre descontinuidades, em que uma maior distância entre as mesmas beneficia a estabilidade do maciço rochoso. Na Fig. 9 observa-se a evolução do peso com a variação do espaçamento, simplificando-se a análise na Tabela 5.

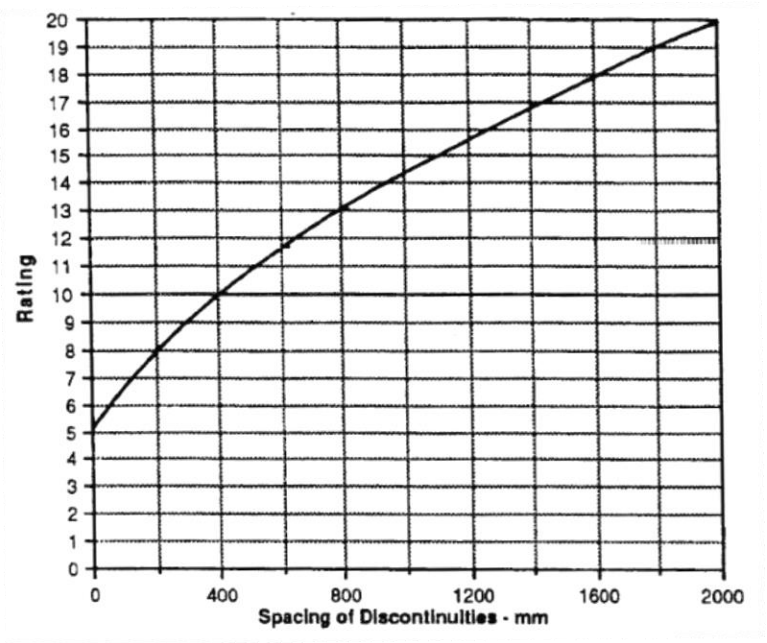


Fig. 9 – Ponderação do espaçamento entre descontinuidades na avaliação do RMR (Bieniawski, 1989)

Tabela 5 – Ponderação do espaçamento entre descontinuidades na avaliação do RMR (Bieniawski, 1989)

Espaçamento entre descontinuidades [mm]	< 60	60 – 200	200 – 600	600 – 2000	> 2000
Peso	5	8	10	15	20

2.1.2.4. Condição das descontinuidades

Neste parâmetro é avaliada a condição das descontinuidades, tendo em conta a caracterização destas através do seu comprimento, separação (abertura), rugosidade, enchimento e alteração. Para tal, Bieniawski propôs dois métodos de classificação: um mais geral, em que as descontinuidades são caracterizadas através de descrições padrão (Tabela 6), e um detalhado, caraterizando individualmente cada aspeto (Tabela 7).

Tabela 6 – Ponderação da condição geral das descontinuidades na avaliação do RMR (Bieniawski, 1989)

Condição das descontinuidades	Enchimento mole com espessura > 5 mm ou juntas contínuas com separação > 5 mm	Superfícies polidas ou enchimento com espessura < 5 mm ou juntas contínuas com separação 1 – 5 mm	Superfícies ligeiramente rugosas, separação < 1 mm, paredes muito alteradas	Superfícies ligeiramente rugosas, separação < 1 mm, paredes ligeiramente alteradas	Superfícies muito rugosas, não contínuas, sem separação, paredes de rocha não alterada
Peso	0	10	20	25	30

Tabela 7 – Ponderação da condição detalhada das descontinuidades na avaliação do RMR (Bieniawski,1989)

Comprimento	< 1 m	1 – 3 m	3 – 10 m	10 – 20 m	> 20 m
	6	4	2	1	0
Separação (abertura)	Fechada	< 0,1 mm	0,1 – 1 mm	1 – 5 mm	> 5 mm
	6	5	4	1	0
Rugosidade	Muito rugosa	Rugosa	Ligeiramente rugosa	Lisa	Espelhada
	6	5	3	1	0
Enchimento	Nenhum	< 5 mm, duro	> 5 mm, duro	< 5 mm, mole	> 5 mm, mole
	6	4	2	2	0
Alteração	Não alterada	Ligeira	Moderada	Muito alterada	Decomposta
	6	5	3	1	0

2.1.2.5. Influência da água

A influência de água subterrânea pode ser avaliada de diferentes formas: quantitativamente, através da medição do caudal ao longo de 10 metros de comprimento do túnel ou do quociente entre a pressão da água e a tensão principal máxima, ou qualitativamente, recorrendo às condições gerais. Na Tabela 8 é apresentado o peso atribuído para cada tipo de caraterização.

Tabela 8 – Ponderação da influência de água subterrânea na avaliação do RMR (Bieniawski, 1989)

Caudal por 10 metros (l/min)	Nulo	< 10	10 – 25	25 – 125	> 125
Pressão da água / Tensão principal	0	0 – 0,1	0,1 – 0,2	0,2 – 0,5	> 0,5
Condições gerais	Seco	Húmido	Saturado	Gotejante	Escorrência
Peso	15	10	7	4	0

2.1.2.6. Orientação das descontinuidades

O último parâmetro da classificação serve como correção ao índice RMR, uma vez que o valor do seu peso é nulo ou negativo. Assim, considera-se que o somatório dos pesos dos cinco primeiros parâmetros corresponde ao RMR básico, tornando-se RMR total quando corrigido.

Este parâmetro reflete a influência da direção e inclinação das descontinuidades relativamente à escavação do maciço rochoso, sendo esta descrita qualitativamente (Tabela 9). O valor do peso é dependente do tipo de obra de engenharia em análise onde, através de estudos em túneis, taludes e fundações elaborados por diversos autores, Bieniawski propôs a correção apresentada na Tabela 10.

Tabela 9 – Classificação qualitativa do efeito da orientação das descontinuidades (Bieniawski, 1989)

Direção perpendicular ao eixo do túnel				Direção paralela ao eixo do túnel		
Abertura do túnel no sentido da inclinação		Abertura do túnel no sentido inverso da inclinação		Inclinação 45 – 90º	Inclinação 20 – 45º	Inclinação 0 – 20º
Inclinação 45 – 90º	Inclinação 20 – 45º	Inclinação 45 – 90º	Inclinação 20 – 45º			
Muito favorável	Favorável	Razoável	Desfavorável	Muito desfavorável	Razoável	Razoável

Tabela 10 – Ponderação da orientação das descontinuidades na avaliação do RMR (Bieniawski, 1989)

Orientação das descontinuidades		Muito favorável	Favorável	Razoável	Desfavorável	Muito desfavorável
Peso	Túneis e minas	0	-2	-5	-10	-12
	Fundações	0	-2	-7	-15	-25
	Taludes	0	-5	-25	-50	-50

2.2. CLASSIFICAÇÕES GEOMECÂNICAS APLICADAS A TALUDES

2.2.1. ENQUADRAMENTO HISTÓRICO

O sistema RMR revelou-se uma boa ferramenta para descrever a qualidade do maciço rochoso, uma vez que a sua utilização foi adotada globalmente. No entanto, apesar de contemplar diversos tipos de escavação e do cálculo do índice ter sido atualizado tendo em conta diversos estudos de diferentes autores, a classificação RMR encontra-se direcionada para escavações em túneis. Deste modo, foram propostas novas classificações destinadas à análise de minas, fundações e taludes, tendo como base esta classificação.

Relativamente ao estudo de taludes rochosos, constatou-se que a classificação RMR não contemplava o modo de instabilidade presente, o tipo de escavação ocorrida e que o suporte recomendado a aplicar não era apropriado para este caso. Deste modo, vários autores propuseram diferentes classificações destinadas para a escavação de taludes rochosos, tais como a *Slope Mass Rating* (Romana, 1993), *Slope Rock Mass Rating* (Robertson, 1988), *Chinese Slope Mass Rating* (Chen, 1995), *Natural Slope Methodology* (Shuk, 1994), *Slope Stability Probability Classification* (Hack, 1998) e *Modified Slope Stability Probability Classification* (Lindsay *et al.*, 2001).

Das várias classificações mencionadas, a que mais se destacou foi o *Slope Mass Rating*, tendo sido considerada por Bieniawski nas atualizações do cálculo do sistema RMR (Bieniawski, 1989). Além

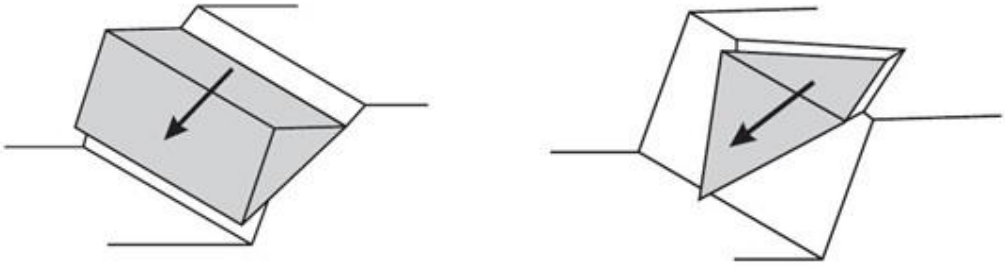
disso, esta classificação foi incluída em livros sobre estabilidade de taludes rochosos, no programa de estudo de cursos de Engenharia e em regulamentos técnicos, comprovando a eficácia deste método (Romana *et al.*, 2015). Desta forma, para a realização deste trabalho, foi estudado o processo de cálculo da classificação SMR.

2.2.2. INSTABILIDADE EM TALUDES ROCHOSOS

Independentemente da classificação utilizada na avaliação do talude rochoso, é necessário identificar os possíveis tipos de rotura. Esta pode variar entre rotura planar, rotura em cunha, rotura por *toppling* ou rotura circular, sendo condicionada pelo grau de fracturação do maciço e pela orientação e distribuição das descontinuidades no talude (González de Vallejo, 2002; Romana, 1993).

A rotura planar (Fig. 10 a) é frequente em taludes com azimute semelhante ao da descontinuidade, podendo provocar o deslizamento do bloco pelo plano da descontinuidade. Assim, quando a inclinação da face do talude é superior à da descontinuidade e quando a resistência ao corte ao longo da superfície desta não é suficiente, pode ocorrer este fenómeno.

De forma semelhante ocorre a rotura em cunha (Fig. 10 b), caracterizada pelo deslizamento do bloco formado pelo plano de duas descontinuidades. Dependendo da geometria do bloco e das tensões de corte mobilizadas, este pode deslizar ao longo da linha interseção das descontinuidades.

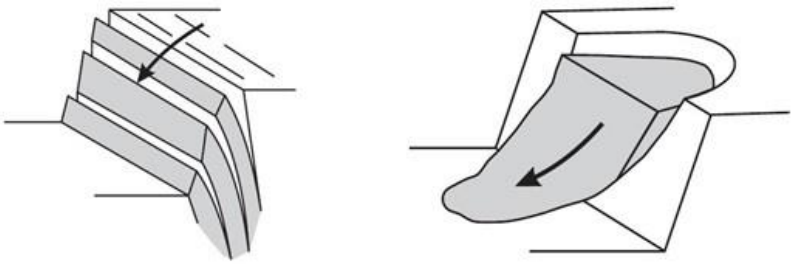


a) b)

Fig. 10 – Representação da rotura: a) planar; b) em cunha (Wyllie e Mah, 2005)

Relativamente à rotura por *toppling* (Fig. 11 a), esta ocorre quando o talude e a descontinuidade têm azimutes aproximadamente paralelos, mas com sentido dos pendores opostos. É frequente em taludes de maciços estratificados onde, através da fracturação dos estratos, os blocos criados tendem a sofrer rotações que provocam a queda dos mesmos.

A rotura circular (Fig. 11 b) ocorre quando o maciço rochoso se encontra muito alterado ou quando apresenta um grande número de descontinuidades, levando a que a superfície de rotura intersete muitas descontinuidades. Esta está associada a índices RMR muito baixos, onde o maciço rochoso se comporta como um solo.



a) b)

Fig. 11 – Representação da rotura: a) por *toppling*; b) circular (Wyllie e Mah, 2005)

2.2.3. *SLOPE MASS RATING*

O sistema SMR é um desenvolvimento da classificação RMR de Bieniawski, sendo uma ferramenta para avaliação da estabilidade de taludes rochosos. Este sistema fornece informações quanto ao tipo de instabilidade que possa existir, recomendando qual o tipo de suporte a aplicar. No entanto, os resultados devem ser analisados com sentido crítico, complementando-os através de métodos analíticos quando necessário (Romana, 1993).

Como referido, esta classificação tem como base o sistema de Bieniawski, somando a parcela de ajustamentos ao RMR básico (somatório das cinco primeiras parcelas da classificação RMR), como apresentado na expressão (8). A parcela de ajustamento é obtida através da soma do produto de três fatores de ajustamento (F_1, F_2, F_3), referentes à relação da orientação descontinuidade – talude, com um quarto fator (F_4), relativo ao método de escavação efetuado. Obtido o índice SMR é possível descrever a qualidade do talude rochoso, bem como conhecer qual o suporte recomendado a aplicar para tais condições.

$$SMR = RMR_b + (F_1 \times F_2 \times F_3) + F_4 \tag{8}$$

2.2.3.1. Fator de ajustamento F_1

O fator de ajustamento F_1 avalia o paralelismo entre os azimutes (α – *dip direction*) da descontinuidade (α_j) e do talude (α_s). Este valor varia entre 1, quando os azimutes são paralelos, e 0,15, quando as direções formam um ângulo superior a 30º e a probabilidade de rotura é menor. Na Tabela 11 é apresentado a variação do valor de F_1, havendo distinção entre a análise de rotura planar e *toppling*.

Tabela 11 – Distribuição do fator de ajustamento F_1 consoante o tipo de rotura (Romana, 1993)

Tipo de rotura		Muito favorável	Favorável	Normal	Desfavorável	Muito desfavorável
Planar	$A = \lvert\alpha_j - \alpha_s\rvert$	> 30º	30 – 20º	20 – 10º	10 – 5º	< 5º
Toppling	$A = \lvert\alpha_j - \alpha_s - 180°\rvert$					
	F_1	0,15	0,4	0,7	0,85	1,0

Apesar da atribuição dos valores de F_1 ter resultado de análises empíricas, Romana propôs uma regressão que relaciona o fator de ajustamento com o ângulo A (9).

$$F_1 = (1 - \sin A)^2 \tag{9}$$

2.2.3.2. Fator de ajustamento F_2

O segundo fator de ajustamento relaciona-se com a possibilidade de rotura planar ao longo da descontinuidade, dependendo assim da inclinação desta (β – *dip*). Deste modo, quando avaliada a rotura por *toppling* o valor de F_2 é unitário.

Tabela 12 – Distribuição do fator de ajustamento F_2 consoante o tipo de rotura (Romana, 1993)

Tipo de rotura		Muito favorável	Favorável	Normal	Desfavorável	Muito desfavorável
Planar	$B = \beta_j$	< 20º	20 – 30º	30 – 35º	35 – 45º	> 45º
	F_2	0,15	0,4	0,7	0,85	1,00
Toppling		1				

De forma análoga ao fator anterior, foi proposto uma correlação entre o fator F_2 e o ângulo B (10), sendo válida para a análise de rotura planar.

$$F_2 = (\tan B)^2 \tag{10}$$

2.2.3.3. Fator de ajustamento F_3

O fator de ajustamento F_3 avalia a relação da inclinação C entre a descontinuidade (β_j) e o talude (β_s), sendo uma adaptação à correção para taludes proposta por Bieniawski (Tabela *10*). Na Tabela 13 é apresentada a proposta de Romana para a distribuição de valores de F_3, novamente com distinção entre a análise de rotura planar ou por *toppling*.

Tabela 13 – Distribuição do fator de ajustamento F_3 consoante o tipo de rotura (Romana, 1993)

Tipo de rotura		Muito favorável	Favorável	Normal	Desfavorável	Muito desfavorável
Planar	$C = \beta_j - \beta_s$	> 10º	10 – 0º	0º	0 – -10º	< -10º
Toppling	$C = \beta_j + \beta_s$	< 110º	110 – 120º	> 120º	-	-
	F_3	0	-6	-25	-50	-60

2.2.3.4. Fator de ajustamento F_4

O último fator de ajustamento estima o desgaste causado no talude devido ao método de escavação utilizado, sendo, evidentemente, mais favorável quando o talude é natural ou quando os processos de escavação são menos destrutivos (Tabela 14).

Tabela 14 - Distribuição do fator de ajustamento F_4 consoante o método de escavação (Romana, 1993)

Método de escavação	Talude natural	Pré-corte	*Smooth blasting*	Explosivos ou mecânico	Desmonte deficiente
F_4	+15	+10	+8	0	-8

2.2.3.5. Análise dos resultados

Uma vez avaliado o RMR básico e os quatros fatores de ajustamento é possível determinar o índice SMR, recorrendo à expressão (8). Para tal é necessário manter o mesmo tipo de rotura em estudo na ponderação dos diferentes fatores de ajustamento e repetir o cálculo para as várias descontinuidades existentes, sendo o valor mínimo obtido o índice SMR associado ao talude. Mediante o valor do índice SMR é possível caraterizar o talude rochoso em cinco classes, estando estas apresentadas na Tabela 15.

$$SMR = RMR_b + (F_1 \times F_2 \times F_3) + F_4 \quad (8)$$

Tabela 15 – Classificação do talude rochoso de acordo com o sistema SMR (Romana, 1993)

Classe	V	IV	III	II	I
SMR	0 – 20	21 – 40	41 – 60	61 – 80	81 – 100
Descrição	Muito mau	Mau	Normal	Boa	Muito boa
Estabilidade	Completamente instável	Instável	Parcialmente instável	Estável	Completamente estável
Rotura	Planar ou circular	Planar ou em cunha	Algumas juntas ou muitas cunhas	Alguns blocos	Nenhuma

Além de classificar o talude rochoso, o sistema SMR indica também qual o suporte recomentado a aplicar. Na Tabela 16 são apresentados os diferentes conjuntos de suporte existentes, na qual se dividiu as classes de caraterização para que a sugestão de suporte fosse mais específica e económica.

Tabela 16 – Suporte recomendado de acordo com o sistema SMR (Romana, 1993)

Classe	SMR	Suporte
I a	91 – 100	Nenhum
I b	81 – 90	Nenhum
II a	71 – 80	Vala na base do talude e/ou vedação Pregagens pontuais ou sistemáticas
II b	61 – 70	Vala na base do talude e/ou vedação Redes metálicas Pregagens pontuais ou sistemáticas
III a	51 – 60	Vala na base do talude e/ou redes metálicas Pregagem pontual ou sistemática Betão projetado pontualmente
III b	41 – 50	Vala na base do talude e/ou redes metálicas Pregagens sistemáticas Ancoragens Betão projetado Muro de suporte e/ou betão de regularização
IV a	31 – 40	Ancoragens Betão projetado Muro de suporte e/ou betão Reescavação Drenagem
IV b	21 – 30	Betão projetado armado Muro de suporte e/ou betão Reescavação Drenagem profunda
V a	11 – 20	Muro gravidade ou parede ancorada Reescavação

2.2.4. MODIFICAÇÕES DO *SLOPE MASS RATING* POR FUNÇÕES CONTÍNUAS

A classificação SMR revelou ser uma ferramenta com grande utilidade para a avaliação de taludes rochosos, já que o seu método de cálculo expedito se traduzia em resultados bastante satisfatórios. No entanto, verificou-se que, na atribuição do valor dos fatores de ajustamento, existiam diferenças significativas entre a utilização das funções discretas (Tabela 11, Tabela 12 e Tabela 13) e as correlações matemáticas. Como tal, Tomás *et al.* propuseram funções alternativas para o cálculo dos fatores de ajustamento F_1 e F_2, e novas funções para o fator de ajustamento F_3 (Tomás *et al.*, 2007).

2.2.4.1. Fator de ajustamento F_1

Relativamente ao cálculo do fator de ajustamento F_1, Tomás propôs a utilização da seguinte expressão (Tomás *et al.*, 2007):

$$F_1 = \frac{16}{25} - \frac{3}{500} \times \tan^{-1}\left(\frac{1}{10} \times (|A| - 17)\right) \quad (11)$$

Na Fig. 12 é possível comparar a função discreta (Tabela *10*) com as funções contínuas propostas por Romana (9) e Tomás (11), verificando-se que a última é a que melhor se ajusta à função original. Esta é válida para qualquer valor de A, apresentando um valor mais conservativo de F_1 que o valor dado pela função proposta por Romana (9).

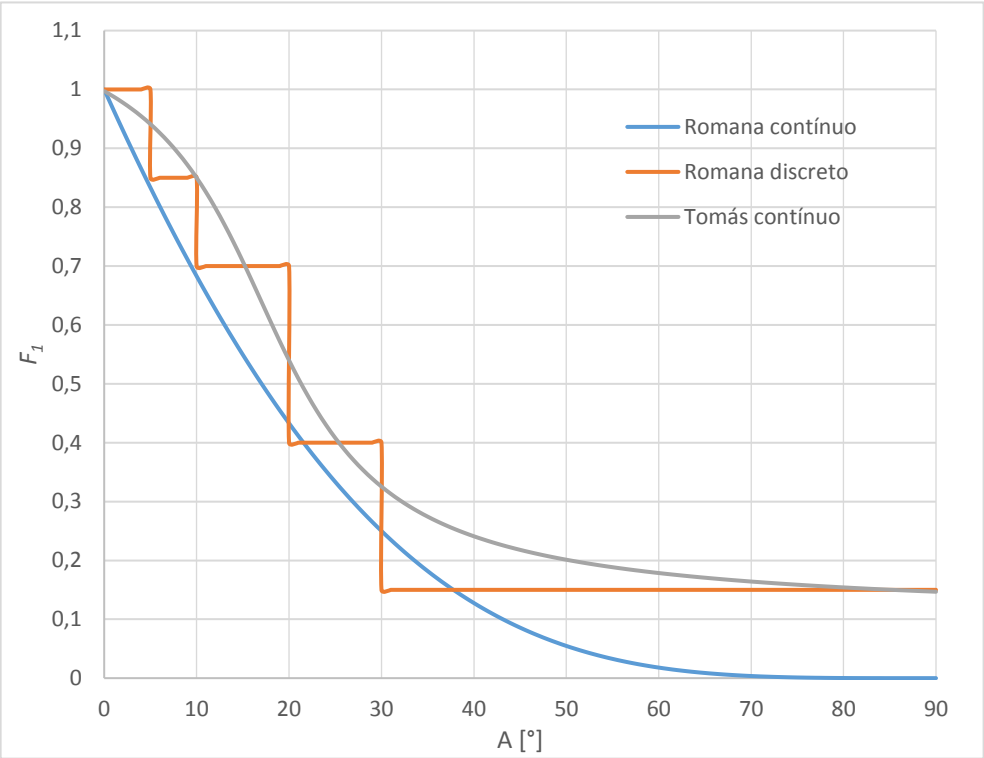


Fig. 12 – Comparação do fator de ajustamento F_1 em função do ângulo A

2.2.4.2. Fator de ajustamento F_2

De forma análoga ao fator F_1, Tomás propôs a seguinte expressão como função alternativa ao cálculo de F_2 (Tomás *et al.*, 2007):

$$F_2 = \frac{9}{16} + \frac{1}{195} \times \tan^{-1}\left(\frac{17}{100} \times B - 5\right) \quad (12)$$

A comparação entre as três funções é apresentada na Fig. 13, sendo evidente que a função contínua proposta por Romana só é válida quando a inclinação da descontinuidade é inferior a 45º. De sentido oposto, verifica-se que a função alternativa proposta por Tomás é válida para todos os valores possíveis de B, aproximando-se com elevada precisão da função discreta (Tabela 11).

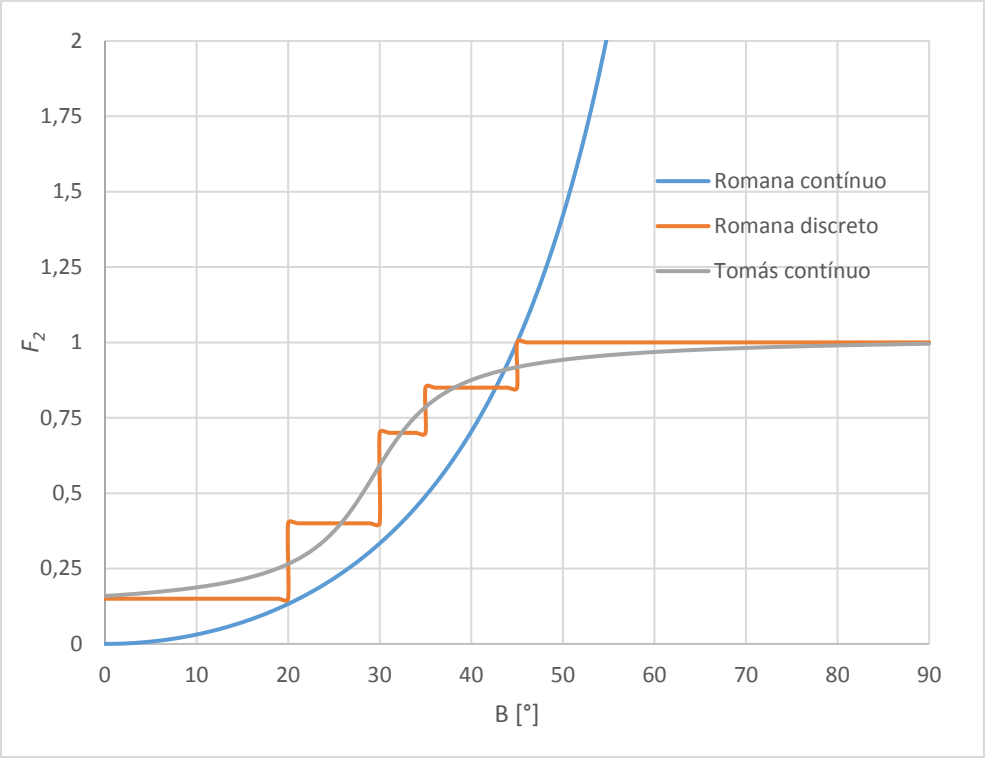


Fig. 13 – Comparação fator de ajustamento F_2 em função do ângulo B

2.2.4.3. Fator de ajustamento F_3

Relativamente ao fator de ajustamento F_3, Romana não propôs o seu cálculo utilizando funções contínuas. No entanto, analisando as funções discretas (Tabela 13), verifica-se que F_3 sofre variações consideráveis apenas numa pequena parcela do seu domínio, onde uma insignificante variação de C pode originar fatores de ajustamento díspares. Deste modo, Tomás propôs funções contínuas para o cálculo do fator de ajustamento na análise de rotura planar e em *toppling*, correspondendo às expressões (13) e (14), respetivamente (Tomás *et al.*, 2007).

$$F_3 = -30 + \frac{1}{3}\tan^{-1} C \tag{13}$$

$$F_3 = -13 - \frac{1}{7}\tan^{-1}(C - 120) \tag{14}$$

Nas Fig. 14 e Fig. 15 são comparadas as funções discretas e contínuas, verificando-se que as últimas são válidas para qualquer valor de C.

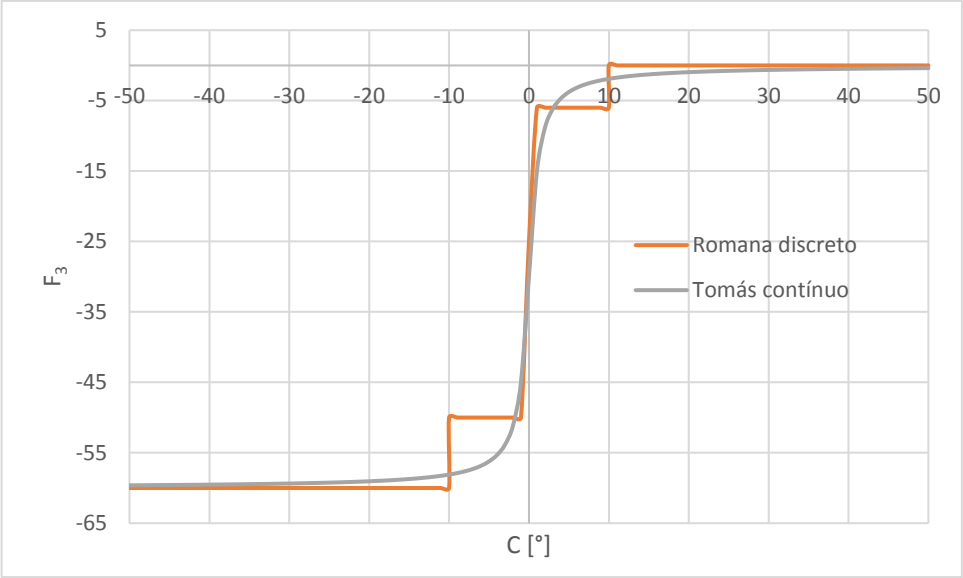


Fig. 14 – Comparação do fator de ajustamento F_3 em função de C, para rotura planar

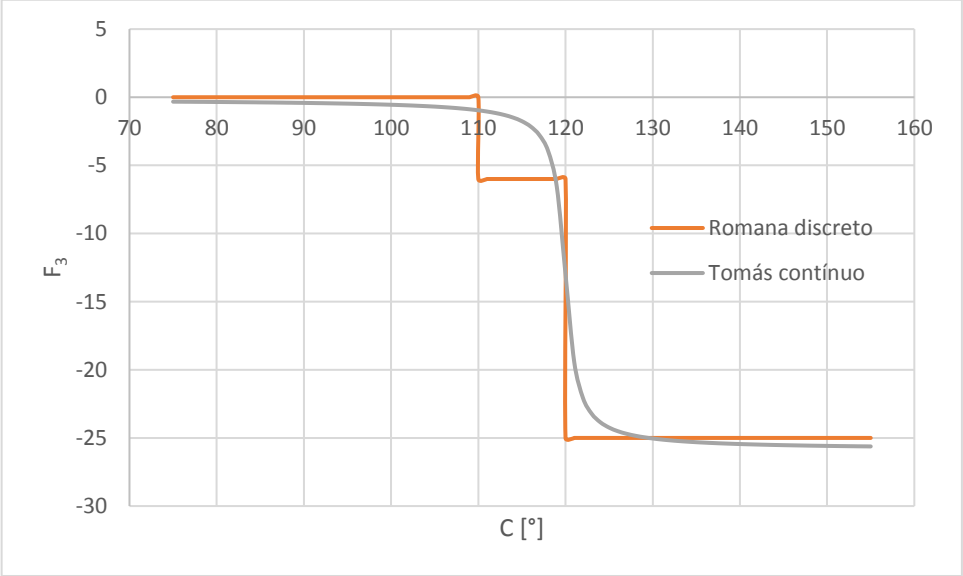


Fig. 15 – Comparação do fator de ajustamento F_3 em função de C, para rotura por *toppling*

3
APLICAÇÕES MÓVEIS IDENTIFICADAS

Dado o interesse em desenvolver uma aplicação (*app*) que auxiliasse o cálculo da classificação SMR, fez-se uma pesquisa na *Google Play Store* sobre a existência de aplicações relacionadas com classificações geomecânicas, nomeadamente aplicadas a taludes rochosos. Deste modo, além de uma análise ao mercado na procura de aplicações semelhantes, foi possível identificar as limitações das existentes, começando-se a idealizar a estrutura da aplicação a desenvolver.

3.1. CLASSIFICAÇÕES GEOMECÂNICAS

Iniciando a procura de aplicações *Android* relativas a classificações geomecânicas, verificou-se a existência de quatro *apps* para esta finalidade, que, pelo número de transferências efetuadas, demonstram um interesse significativo dos utilizadores.

A aplicação com maior relevância na pesquisa foi a *Rock Mass Classification* (2016) (Fig. 16), com mais de 5000 transferências desde agosto de 2013. Esta *app* inclui o cálculo das classificações RQD, RMR, sistema Q e GSI, solicitando ao utilizador de forma simples o parâmetro a definir. No entanto, esta limita-se ao cálculo do índice da classificação escolhida, não apresentando qualquer caracterização do maciço nem do suporte recomendado.

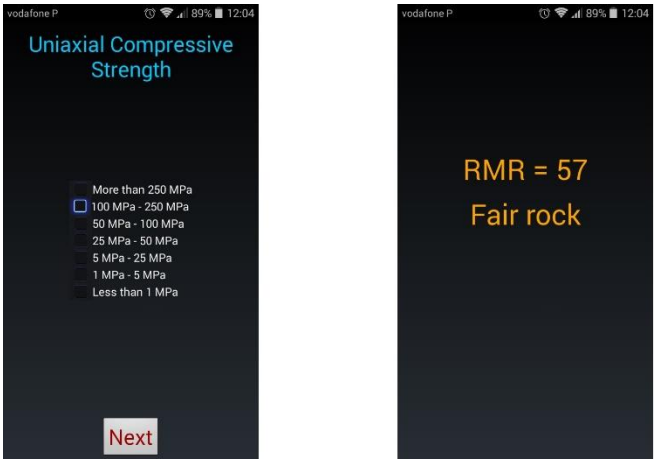


Fig. 16 – Aplicação *Rock Mass Classification*

Outra aplicação com alguma relevância foi a *RMR Calc Free* (2016) (Fig. 17), desenvolvida pela empresa de engenharia geotécnica Terrasolum. Esta tem mais de 1000 transferências e, tal como o nome

índica, resume-se ao cálculo do sistema RMR. Nesta *app*, o utilizador define na mesma apresentação os diferentes parâmetros, através da manipulação de barras. O resultado do índice é apresentado sob a forma de mensagem, bem como as possíveis correlações, como a caraterização do talude, o tempo sem suporte, a coesão e o ângulo de atrito. Esta aplicação possui uma versão completa que, através do pagamento de uma taxa, permite a utilização da aplicação sem exibição de publicidade e o envio dos resultados obtidos, em formato *pdf*, por e-mail.

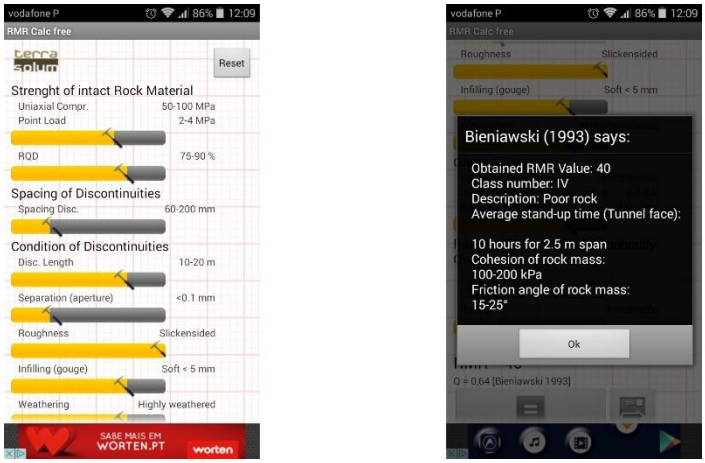
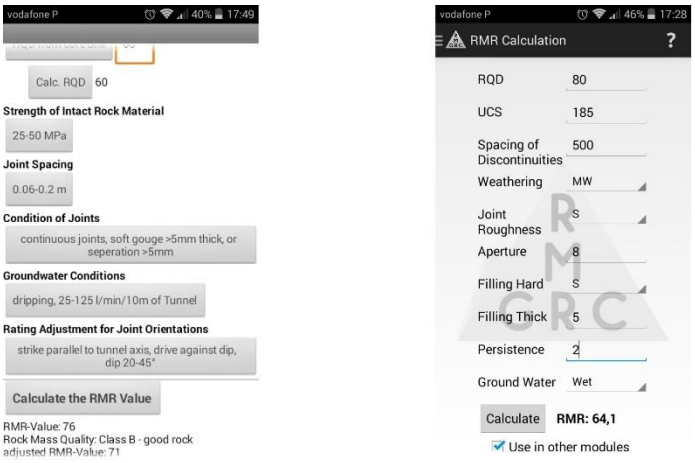


Fig. 17 – Aplicação *RMR Calc Free*

As restantes aplicações, a *GeoToolbox* (2016) (Fig. 18 a) e a *RMR & GRC* (2016) (Fig. 18 b), dado terem apenas cerca de 100 transferências, apresentaram menos relevância na pesquisa. A primeira aplicação, além de possibilitar o cálculo da classificação RMR e do sistema Q, incorpora uma bússola de geólogo. No entanto, possivelmente pela apresentação gráfica pouco atrativa ou pela falta de instruções na sua utilização, não demonstrou interesse dos utilizadores. Relativamente à segunda aplicação, esta inclui o cálculo da classificação RMR e, em função do índice obtido, permite determinar a curva característica do maciço rochoso e o deslocamento do mesmo durante a escavação. Apesar do seu conteúdo ser interessante, o facto de ser a aplicação mais recente e do público dar preferência às que têm mais transferências pode justificar o seu menor sucesso.



a) b)

Fig. 18 – Aplicação: a) *GeoToolbox*; b) *RMR & GRC*

3.2. CLASSIFICAÇÕES GEOMECÂNICAS APLICADAS A TALUDES ROCHOSOS

Relativamente a aplicações destinadas a taludes apenas se encontrou a *Simple Slope* (2016) (Fig. 19), desenvolvida pela Terrasolum. Esta *app* avalia a estabilidade de taludes em maciços terrosos, calculando o fator de segurança pelo método de Bishop. Apesar de evidenciar uma fácil utilização, ao solicitar as variáveis necessárias ao cálculo de forma clara e completando a análise graficamente, o excesso de publicidade exibida interfere na utilização da aplicação. Além disso, funções como modificar a distribuição e valor das cargas aplicadas, alterar a posição do nível freático ou gravar em formato *pdf* os resultados obtidos apenas estão disponíveis mediante o pagamento de uma taxa.

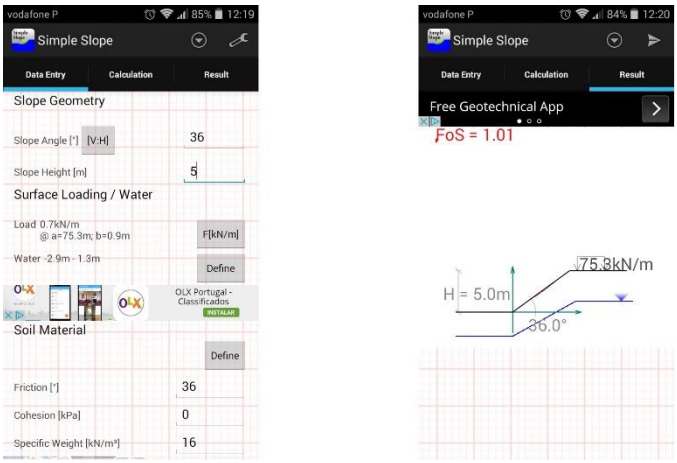


Fig. 19 – Aplicação *Simple Slope*

Visto que a única aplicação *Android* existente se destina a taludes em maciços terrosos, o desenvolvimento de uma *app* que aplicasse classificações geomecânicas a taludes rochosos seria algo inovador. Como tal, alargou-se a pesquisa efetuada para outros dispositivos, possibilitando uma análise mais detalhada ao processo de cálculo a desenvolver. Deste modo, verificou-se a existência da *SMRTool* (2016) (Fig. 20), uma ferramenta de cálculo destinada a computadores, sendo disponibilizada em executável de *Matlab* e em ficheiro *Excel*.

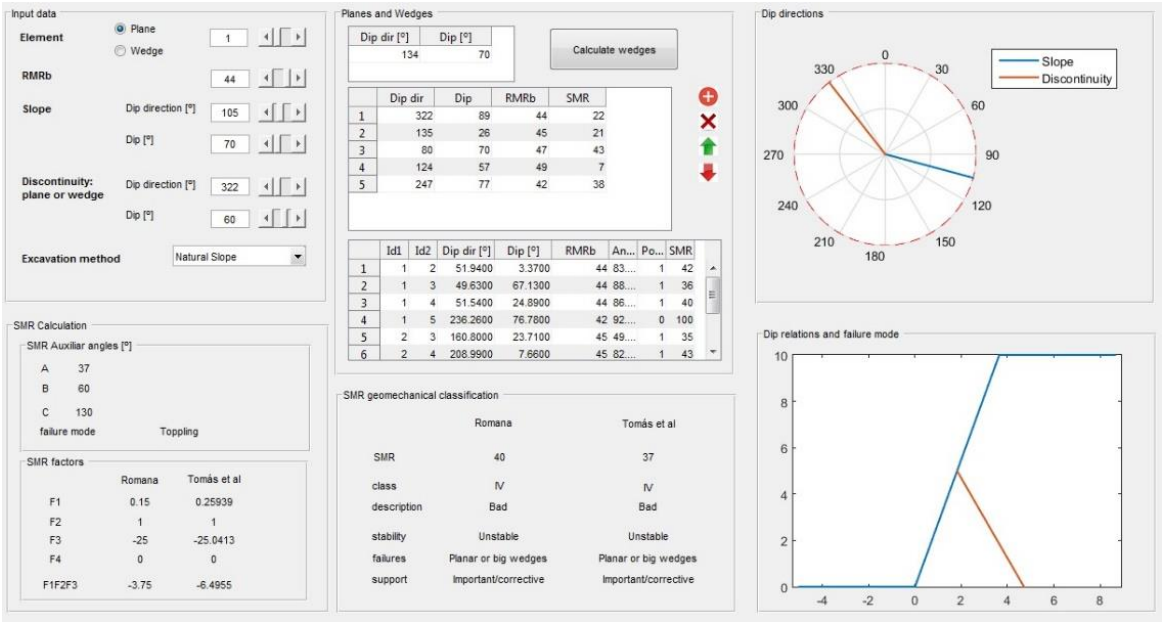


Fig. 20 – Programa *SMRTool*

Através da introdução da orientação do talude e da descontinuidade, do índice RMR básico e do método de escavação, este programa calcula a classificação SMR segundo Romana e Tomás (Romana, 1993; Tomás *et al.*, 2007). Este demostra ser bastante completo, uma vez que compara os dois métodos de cálculo, analisa graficamente as orientações inseridas e carateriza o talude rochoso em diferentes aspetos. Além disso, permite inserir várias descontinuidades, analisando os blocos em cunha formados.

3.3. ANÁLISE COMPARATIVA ENTRE APLICAÇÕES

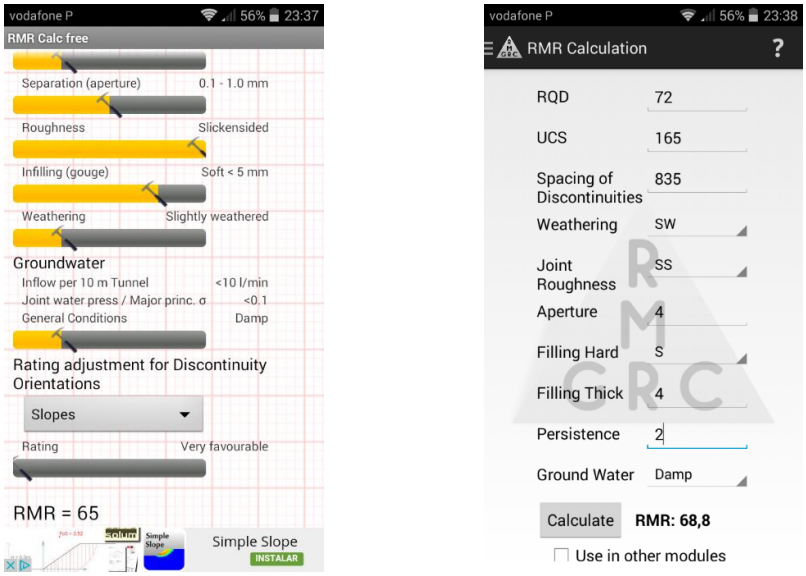
Concluída a pesquisa de aplicações *Android* sobre classificações geomecânicas, decidiu-se utilizá-las na análise de um exemplo de um maciço rochoso, verificando se as mesmas aplicavam corretamente os critérios de Bieniawski e, avaliando do ponto de vista do utilizador, a facilidade de caracterização do maciço. Deste modo, foram definidas as condições do maciço em estudo, que se apresentam na Tabela 17, em conjunto com o peso associado utilizando a classificação RMR.

Tabela 17 – Condição do maciço rochoso

Parâmetro		Valor	Peso
Resistência à compressão uniaxial		165 MPa	12
RQD		72 %	13
Espaçamento entre descontinuidades		835 mm	15
Água Subterrânea		Paredes húmidas	10
Características das descontinuidades	Comprimento	2 m	4
	Abertura	4 mm	4
	Rugosidade	Espelhada	0
	Enchimento	Mole, 4 mm	2
	Alteração	Ligeira	5
		RMR básico	65

Estas condições foram inseridas nas quatro aplicações anteriormente apresentadas, tendo-se obtido valores do RMR básico bastante díspares. A *RMR Calc Free* (Fig. 21 a), com um índice RMR básico de 65, obteve um resultado igual ao esperado, indicando a correta utilização dos critérios de Bieniawski. Para além disso, o facto de a aplicação permitir a caracterização detalhada das descontinuidades, possibilitou a obtenção do mesmo índice. Este tipo de caraterização também está presente na *RMR & GRC* (Fig. 21 b), motivo pelo qual obteve o segundo resultado mais próximo, com um índice de 68,5. A diferença deste valor para o da classificação, e o facto de o resultado ser um número decimal, pode

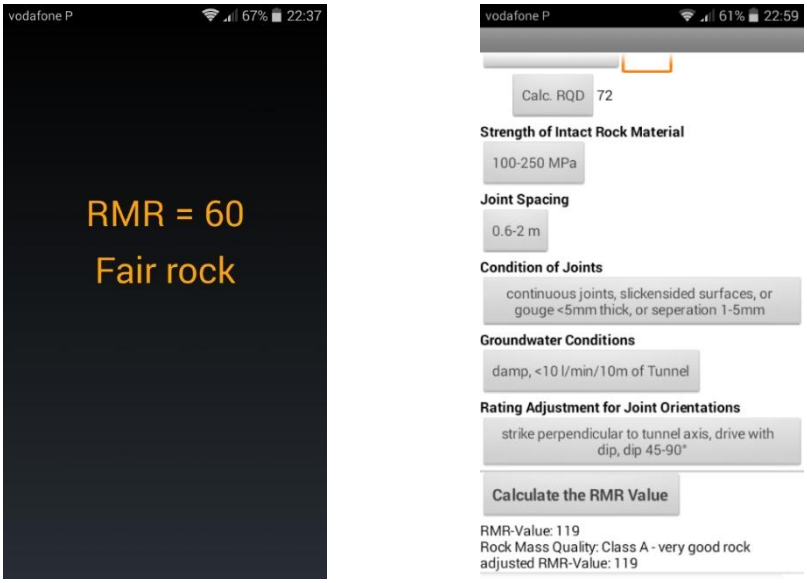
indicar a utilização de funções contínuas no cálculo do peso dos parâmetros da resistência à compressão uniaxial, do índice RQD e do espaçamento entre descontinuidades. Apesar do resultado obtido ser aceitável face ao esperado, a utilização de acrónimos na caracterização das condições do maciço (como SW para *Slightly Weathered* ou SS para *Slickensided*) pode influenciar e dificultar a interpretação do utilizador.



a) b)

Fig. 21 – Cálculo do RMR básico através da aplicação: a) *RMR Calc Free*; b) *RMR & GRC*

A *Rock Mass Classification* (Fig. 22 a), com um índice RMR básico de 60, apresentou um resultado ligeiramente diferente ao esperado, que se justifica com a possibilidade de caraterizar a descontinuidade apenas pelas classificações gerais. A *GeoToolbox* (Fig. 22 b) revela falhas graves na utilização dos critérios de Bieniawski, uma vez que apresenta um índice RMR básico de 119, muito diferente do valor esperado.



a) b)

Fig. 22 Cálculo do RMR básico através da aplicação: a) *Rock Mass Classification*; b) *GeoToolbox*

4

DESENVOLVIMENTO DE UMA APLICAÇÃO MÓVEL

4.1. CONTEXTUALIZAÇÃO

Como verificado no Capítulo 2, o sistema SMR é uma ferramenta útil para a avaliação da estabilidade de taludes rochosos dado que, recorrendo a tabelas e aplicando pequenos cálculos, é possível classificar a qualidade do talude. No entanto, apesar da facilidade de cálculo promover a sua utilização em obra, a necessidade de consulta de diversas tabelas pode ser considerada uma desvantagem. Deste modo, surgiu a hipótese de elaborar uma aplicação para dispositivos móveis suportados por *Android* que substituísse a consulta das diferentes tabelas, simplificando a utilização da classificação SMR.

Uma vez que a principal característica na determinação da classificação RMR e do sistema SMR é a existência de funções discretas que simplificam os seus cálculos, com a utilização de uma folha de cálculo esta vantagem é minimizada. Desta forma, deu-se preferência ao cálculo por funções contínuas, eliminando os patamares existentes nas funções discretas para a evolução dos pesos e fatores, traduzindo-se em variações menos abruptas e em resultados com maior exatidão.

Relativamente ao cálculo da classificação RMR, verifica-se que Bieniawski apenas propôs a utilização de funções discretas, uma vez que, apesar de ter esboçado a evolução do peso em função do parâmetro correspondente, não definiu a sua expressão (Bieniawski, 1989). No entanto, através da identificação de vários pontos do gráfico, foi possível obter uma regressão que se aproximava ao esboço apresentado na Fig. 6, Fig. 8 e Fig. 9. Este processo foi aplicado aos parâmetros da resistência à compressão uniaxial (Fig. 23), RQD (Fig. 24) e espaçamento entre descontinuidades (Fig. 25), para os quais, de forma a verificar a correta regressão, se comparou com as funções discretas apresentadas nas Tabela 2, Tabela 4 e Tabela 5, respetivamente.

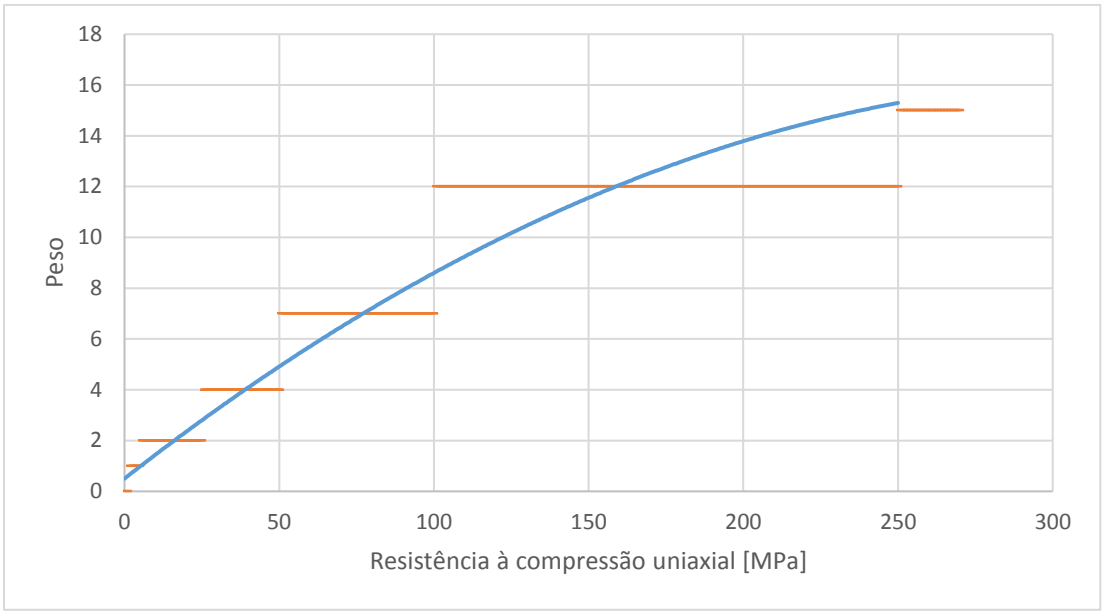


Fig. 23 – Comparação entre funções discretas e contínuas para a ponderação da resistência à compressão uniaxial na avaliação do RMR

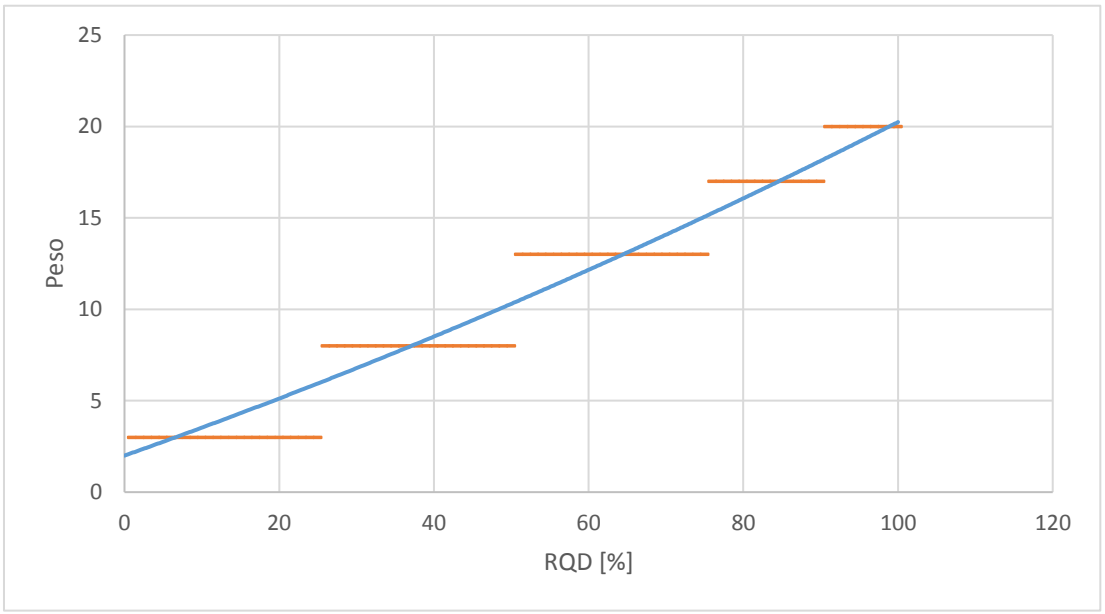


Fig. 24 – Comparação entre funções discretas e contínuas para a ponderação do RQD na avaliação do RMR

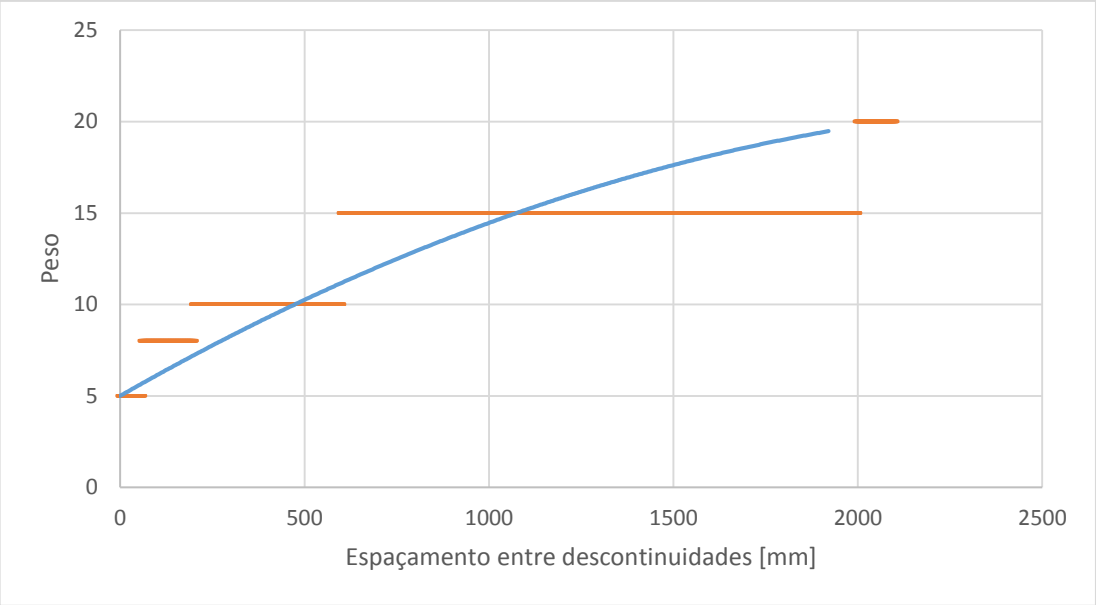


Fig. 25 – Comparação entre funções discretas e contínuas para a ponderação do espaçamento entre descontinuidades na avaliação do RMR

Analisando as Fig. 23, Fig. 24 e Fig. 25 verifica-se que as regressões criadas apresentam uma razoável aproximação à função discreta correspondente. Como tal, o peso dos parâmetros resistência à compressão uniaxial, RQD e espaçamento entre descontinuidades (d) será calculado pelas seguintes expressões, respetivamente:

$$Peso = -0{,}000145\sigma_c^2 + 0{,}095521\sigma_c + 0{,}5 \quad (15)$$

$$Peso = -0{,}000616RQD^2 + 0{,}113103RQD + 3 \quad (16)$$

$$Peso = -0{,}000002d^2 + 0{,}011552d + 5 \quad (17)$$

De forma análoga, no sistema SMR serão utilizadas funções contínuas para o cálculo dos fatores de ajustamento. Assim, e devido à melhor aproximação verificada no Capítulo 2, serão utilizadas as funções propostas por Tomás para a determinação dos diferentes fatores (Tomás *et al.*, 2007).

Com o objetivo de aumentar as funcionalidades da aplicação, será permitido ao utilizador determinar as orientações do talude e da descontinuidade através das capacidades do dispositivo móvel. Apesar dos sensores utilizados sofrerem descalibrações significativas com a aproximação a aparelhos eletrónicos ou objetos metálicos, a calibração pode ser facilmente reposta ao girar o dispositivo segundo os três eixos, voltando a apresentar o valor do ângulo medido com bastante exatidão.

4.2. *ANDROID* E *ANDROID STUDIO*

O *Android* é um sistema operativo móvel desenvolvido pela empresa *Google*, destinado a *smartphones*, *smartwatches*, *tablets*, televisões e automóveis. Lançada a primeira versão em 2008, o *Android* tem sofrido atualizações recorrentes, adicionando novas funcionalidades em virtude do desenvolvimento do *hardware*, com significativas melhorias do aspeto gráfico e correção de erros de versões anteriores.

Associada a um diferente nível API (*Application Programming Interface* – Interface de Programação de Aplicações) para uma identificação mais técnica, a cada versão é também atribuído um nome de

código para uma identificação direcionada ao utilizador comum. Este tem a particularidade de ser identificado com o nome de uma sobremesa, seguindo também uma lógica alfabética. Assim, começando com a versão *Cupcake*, a versão mais recente é a *Marshmallow*, encontrando-se em desenvolvimento a *Nougat*. Na Tabela 18 é possível verificar a evolução do sistema operativo, bem como a distribuição de dispositivos que atualmente utilizam cada versão.

Tabela 18 – Versões *Android* (Android Developers, 2016a)

Versão	Data de lançamento	Nome de código	Nível API	Distribuição
1.5	Abril de 2009	*Cupcake*	3	-
1.6	Setembro de 2009	*Donut*	4	-
2.0	Outubro de 2009	*Eclair*	7	-
2.2	Maio de 2010	*Froyo*	8	0,1 %
2.3.3 – 2.3.7	Dezembro de 2010	*Gingerbread*	10	1,7 %
3	Fevereiro de 2011	*Honeycomb*	11	-
4.0	Outubro de 2011	*Ice cream Sandwich*	15	1,6 %
4.1	Julho de 2012	*Jelly Bean*	16	6,0 %
4.2	Novembro de 2012		17	8,3 %
4.3	Julho de 2013		18	2,4 %
4.4	Outubro de 2013	*KitKat*	19	29,2 %
5	Novembro de 2014	*Lollipop*	21	35,5 %
6	Outubro de 2015	*Marshmallow*	23	15,2 %
7	Agosto de 2016	*Nougat*	24	-

O sucesso deste sistema operativo reflete-se no domínio do mesmo no mercado, uma vez que aproximadamente 80 % dos dispositivos vendidos operam com o *Android*. A preferência por parte dos fabricantes de dispositivos por este sistema operativo deve-se, em parte, ao facto de a *Google* disponibilizar o código fonte do *Android* de forma aberta e gratuita, possibilitando modificações personalizadas para cada marca ou modelo. Para além disso, é também uma escolha preferencial por parte do utilizador, pelo que se trata de um sistema operativo de fácil utilização, bastante intuitivo e que possibilita a compra de dispositivos mais simples a preços acessíveis, comparativamente à complexidade e custo de dispositivos suportados pelo *Windows Phone* ou pelo *Iphone Operating System* (*iOS)*.

Uma outra vantagem do *Android* é a existência do *Android Studio* (Android Developers, 2016b), um IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado) utilizado em computadores para desenvolvimento de aplicações, sendo disponibilizado de forma gratuita pela *Google*, através do *Android Developer*. Este programa é suportado por qualquer sistema operativo (*Windows*, *Mac OS* e *Linux*), possibilitando a qualquer utilizador, com conhecimentos de linguagem de

programação *Java*, a criação de aplicações para utilização nos dispositivos móveis. É ainda possível, através de um registo como programador e do pagamento de uma taxa de 25 dólares, a partilha de aplicações na *Google Play Store*, uma loja de distribuição, de forma gratuita ou remunerada, de aplicações, jogos, filmes, música e livros. Para se perceber a dimensão da *Google Play Store*, esta está disponível a mais de 1000 milhões de utilizadores distribuídos por mais de 190 países, contendo 2 106 529 *apps* gratuitas e 206 096 *apps* pagas (Android Statistics, 2016; Queiróz, 2016).

4.3. DESENVOLVIMENTO DA APLICAÇÃO

Face às vantagens apresentadas e devido ao acesso a diferentes dispositivos móveis compatíveis, decidiu-se desenvolver a aplicação para aparelhos com o sistema operativo *Android*, utilizando-se para tal o IDE *Android Studio*. Após a transferência e instalação do programa foi necessário descarregar um número significante de pacotes de dados de modo a garantir o correto funcionamento da aplicação e a compatibilidade com os diversos dispositivos e versões existentes. Além disso, era frequente a notificação de existência de atualizações recomendadas, demonstrando o constante desenvolvimento do *software* e do sistema operativo.

Seguindo a sugestão do *Android Developer*, desenvolveu-se uma pequena aplicação onde se exibia o texto "Olá Mundo!". Apesar desta ser uma aplicação bastante simples, facilitou o primeiro contacto com o *Android Studio*, identificando-se os diversos passos necessários à criação de um projeto. Verificou-se a distinção entre a componente do código fonte, onde através de linguagem *Java* se definia as ações a tomar, do componente *layout*, onde se inseria todas as ferramentas necessárias e se configurava a imagem a apresentar ao utilizador, através da visualização de *Design* ou de linguagem XML. Na Fig. 26 é visível a execução da aplicação "Olá Mundo!" num dispositivo móvel de 5 polegadas, suportado pelo API 18.

Fig. 26 – Execução da aplicação "Olá Mundo!"

De forma a colmatar a carência de conhecimentos de linguagem *Java* e para conhecer os inúmeros métodos e ferramentas disponíveis no *Android Studio*, foram desenvolvidas diversas pequenas aplicações de aprendizagem recorrendo a tutoriais, vídeos e livros. Assim, começando com aplicações mais gerais e simples, foi-se procurando conhecer ao detalhe o funcionamento de ferramentas e métodos que poderiam ser utilizadas na *app* a desenvolver. Destes exemplos criados, destaca-se a calculadora simples, que através do toque num botão de ação soma os valores de duas caixas de texto, apresentando o resultado através de mensagem, e uma bússola (Fig. 27).

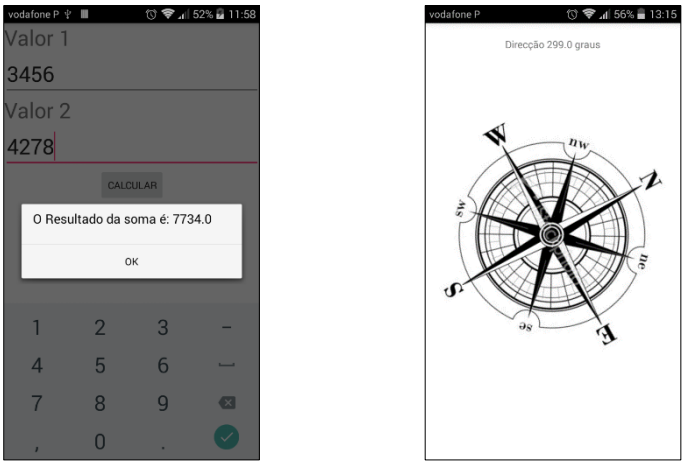


Fig. 27 – Exemplo de calculadora simples e bússola

Através do conhecimento adquirido pelo uso de diversas ferramentas e da sua manipulação, utilizando diferentes métodos, foi iniciado o desenvolvimento da aplicação proposta.

4.3.1. INÍCIO DO PROJETO

Com a criação de um novo projeto foi necessário definir qual a versão mínima do sistema operativo para o qual a aplicação podia ser executada, sendo esta escolha o resultado da conjugação entre vários fatores. Se por um lado uma versão do sistema operativo mais antiga significa um maior número de possíveis utilizadores (Tabela 18), traduz-se também na possibilidade de incompatibilidades na execução da aplicação. Estas incompatibilidades podem ser provocadas pela necessidade de execução em dispositivos com *hardware* mais evoluído ou pela aplicação utilizar ferramentas ou métodos não suportados por essas versões. Além disto, ao longo das atualizações do *Android*, as ferramentas existentes tendem a sofrer alterações na sua forma, apresentando um aspeto mais moderno, ou possibilitando novas funções, como alterar a cor predefinida da ferramenta. Assim, a mesma aplicação pode não reconhecer determinada função num dispositivo antigo ou apresentar um *layout* (parte gráfica visível ao utilizador) diferente num dispositivo mais recente.

Outro aspeto que necessita de análise, e que indiretamente se tornou determinante na escolha da versão mínima a utilizar, é a quantidade de informação em cada *layout*. Analisando o mercado de venda de dispositivos móveis, bem como a sua evolução, constata-se que atualmente a grande maioria dos dispositivos possui um ecrã igual ou superior a 5 polegadas, em detrimento dos mais antigos que, pelas suas funções mais básicas, não justificavam dimensões além das 4. Face a esta tendência e, prevendo que por vezes a quantidade de informação possa ser significativa e para que a sua apresentação seja clara e sem a ocupação integral do ecrã, definiu-se que a utilização ótima da aplicação seria num dispositivo de 7 polegadas, sendo que o limite mínimo aceitável seria de 5.

Tendo em conta os aspetos referidos, foi tomada a opção para que a aplicação estivesse disponível para dispositivos com, no mínimo, a versão 4.3 do *Android* instalada, correspondendo ao API 18 (Tabela 17), abrangendo 82,3 % (somatório da distribuição do API igual ou superior a 18) dos utilizadores com este sistema operativo. Apesar do nível API mínimo ser definido na criação de um novo projeto, este pode ser alterado no decorrer do desenvolvimento da aplicação através do acesso ao ficheiro *build.gradle* (Anexo I), local onde também se declara a versão da *app*.

Uma vez que o lançamento do *Android Nougat* está previsto para um futuro próximo e, de forma a que não existam incompatibilidades quando os dispositivos atualizarem o sistema operativo, definiu-se que a aplicação também estaria disponível para o API 24. Nos seguintes tópicos serão apresentadas as etapas e características mais relevantes da aplicação desenvolvida, pelo que a totalidade do código fonte se encontra em anexo (Anexo II – XIV).

4.3.2. ATIVIDADE PRINCIPAL – *ACT_MAIN*

Uma atividade é a componente principal de uma aplicação, sendo o local onde o código-fonte é introduzido para que esta seja executada. Deste modo, cada *layout* é associado a, no mínimo, uma atividade. Portanto, verifica-se que qualquer aplicação que apresente diferentes *layouts* na sua execução, é constituída por várias atividades.

Prevendo a necessidade de introduzir uma quantidade significativa de conteúdo na aplicação a desenvolver, optou-se pela utilização da ferramenta *TabHost*. Este componente cria uma barra de separadores no *layout* principal, permitindo uma rápida navegação entre *layouts* secundários. Assim, a organização da aplicação é facilitada, possibilitando a divisão de conteúdo em diferentes *layouts* e reduzindo o conteúdo apresentado em cada separador (*Tab*).

Face à existência deste recurso e, para dividir o cálculo da classificação SMR de uma forma lógica, decidiu-se inserir oito separadores, criando oito *layouts* e as respetivas atividades. Nos primeiros cinco separadores são caraterizados os parâmetros definidos por Bieniawski, sendo apresentado no sexto o resultado do RMR básico. No mesmo sentido, no sétimo separador é concluída a caraterização do talude, apresentando-se no último separador o resultado do índice SMR, bem como a classificação e o suporte recomendado.

Desta forma, inseriu-se no *layout* associado à atividade principal (*act_main*) a ferramenta *TabHost*. A inserção de ferramentas pode ser feita de diferentes formas, sendo que a mais simples consiste, com o *layout* aberto no modo de visualização de *Design*, na pesquisa na lista de ferramentas existentes o recurso pretendido e arrastá-lo para a parte gráfica, sendo-lhe atribuído um *id* (identidade). Na Fig. 28 é apresentada a visualização de *Design* para um dos separadores criados, evidenciando-se a lista de ferramentas disponíveis, a componente gráfica apresentada ao utilizador e as propriedades da ferramenta selecionada.

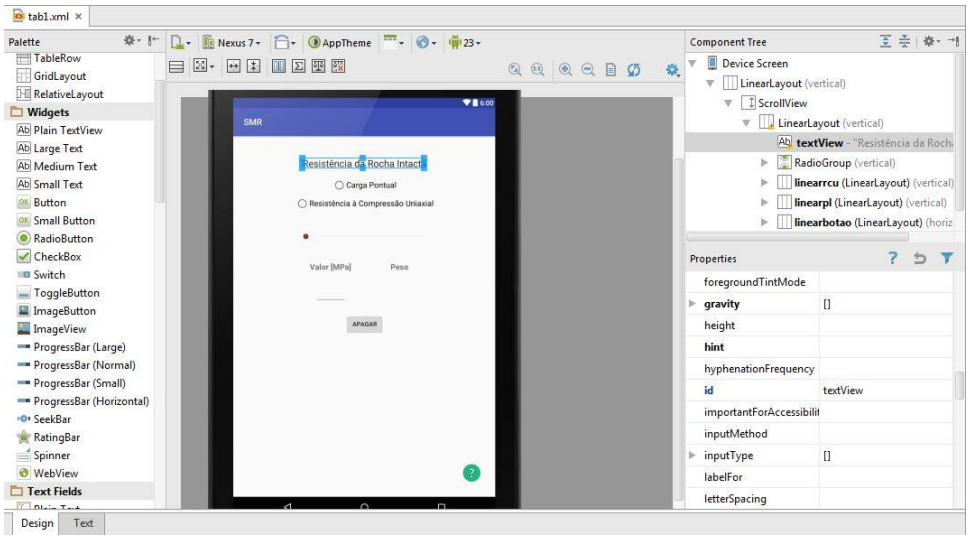


Fig. 28 – Modo *Design* do *layout*

Posteriormente foi necessário configurar a ferramenta inserida, quer quanto à sua apresentação, quer quanto à ação desempenhada por esta na aplicação. Assim, com o *layout* aberto no modo de visualização de *Design*, é possível configurar os aspetos estéticos através das propriedades, como tamanho, cor e posição no *layout*. Estes foram também definidos recorrendo a XML, uma linguagem estruturada e de fácil leitura, que permite uma clara análise do projeto. Na Fig. 29 é apresentado um pequeno excerto da configuração do *layout* em XML, para o separador exemplificado na figura anterior.

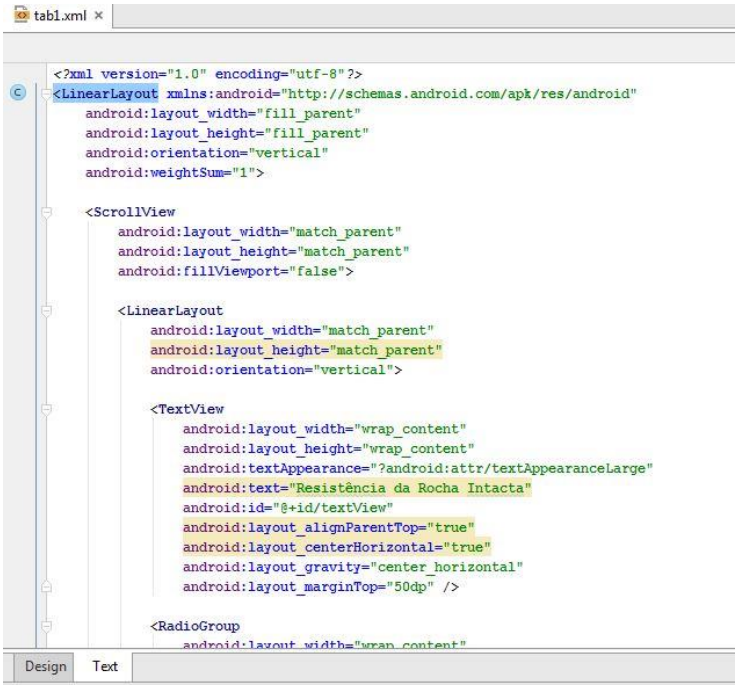


Fig. 29 – Modo *Text* do *layout*, usando linguagem XML

Após estar definido o *layout* a apresentar, foi necessário configurar a utilização da aplicação, em linguagem *Java*. Assim, começou-se por utilizar o método *findViewById*, invocando a ferramenta a manipular e atribuindo-lhe uma denominação, que normalmente é o próprio *id* ou uma abreviatura deste. Na Fig. 30 é implementado este método ao *TabHost* inserido.

```
final TabHost tabHost = (TabHost)findViewById(android.R.id.tabhost);
```

Fig. 30 – Método *findViewById*

Este é o método base para a implementação de qualquer código, uma vez que é pela denominação atribuída que a aplicação reconhece a ferramenta a manipular. Deste modo, procedeu-se à configuração do *TabHost*, criando os diferentes separadores e definindo o seu rótulo. Na Fig. 31 é apresentada a configuração necessária para cada separador, tendo-se exemplificado para o separador "Resistência", correspondente ao Tab1 da aplicação criada.

```
TabHost.TabSpec tab1 = tabHost.newTabSpec("Tab1");
tab1.setIndicator("Resistência");
tab1.setContent(new Intent(this, Tab1.class));
tabHost.addTab(tab1);
```

Fig. 31 – Configuração do separador "Resistência"

Na Fig. 32 é apresentado o *TabHost* criado, verificando-se que, devido ao elevado número de separadores criados e à dimensão dos seus rótulos, foi necessário inserir um *scroll* horizontal para o utilizador navegar ao longo de todo o *TabHost*.

Fig. 32 – *TabHost*

4.3.3. RESISTÊNCIA DA ROCHA INTACTA – *TAB1*

Neste primeiro separador é solicitada a introdução do parâmetro relativo à resistência da rocha intacta. Como tal, de forma a apresentar o parâmetro a introduzir, inseriu-se um *textView* com o texto "Resistência da Rocha Intacta", servindo assim de título do separador (Fig. 33). Este componente é uma caixa de texto não editável pelo utilizador, podendo ser constante ou variável ao longo da execução da aplicação, sendo, por norma, utilizada para a apresentação de títulos e legendas.

Fig. 33 – Exemplo de um *textView*

Sendo possível caracterizar a resistência da rocha intacta através da resistência à compressão uniaxial ou da carga pontual, é necessário diferenciar a introdução este parâmetro. Como tal, inseriu-se duas opções de *radioButton* onde o utilizador poderá selecionar como pretende caracterizar a resistência (Fig. 34).

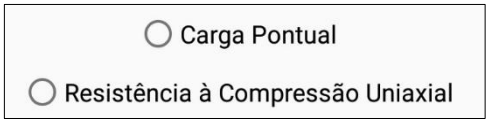


Fig. 34 – Exemplo de dois *radioButton*

Através da seleção de cada opção torna-se visível um *sublayout* que agrupa as ferramentas necessárias à caraterização do parâmetro. Na Fig. 35 apresenta-se a implementação do método *setOnClickListener* a um *radioButton*, verificando-se que através da seleção desta ferramenta, a visibilidade dos *sublayouts* criados é alterada.

```
radioButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        linearpl.setVisibility(View.VISIBLE);
        linearrcu.setVisibility(View.INVISIBLE);

    }
});
```

Fig. 35 – Método *setOnClickListener*

Uma vez que, independentemente da opção selecionada, a informação a introduzir é quantitativa, optou-se pela utilização de um *seekBar* em cada *sublayout* criado. Esta ferramenta é a mais comum na introdução de dados numéricos devido à sua simples e rápida manipulação, sendo necessário associá-la a uma caixa de texto para exibir o valor selecionado. Assim, inseriu-se um *editText*, componente

semelhante ao *textView* mas que permite ser editado pelo utilizador, possibilitando-lhe uma alternativa na introdução do valor pretendido. Optou-se por completar o *sublayout* com a inserção de um *textView* que exibe o peso correspondente ao valor da resistência inserida. Na Fig. 36 é apresentado o *sublayout* visível com a seleção da opção "Resistência Compressão Uniaxial".

Fig. 36 – Exemplo do *sublayout* criado

Posteriormente à criação dos *sublayouts* foi necessária a sua configuração, tendo-se implementado o método *setOnSeekBarChangeListener* a cada *seekBar*. Através deste, a sincronização entre a variação da posição do *seekBar* e os valores apresentados nas caixas de texto é automática. Deste modo, definiu-se que o *editText* apresenta o valor inserido no *seekBar (progress*), enquanto o *textView* exibe o peso calculado pela expressão (15). Para tal, no *sublayout* relativo à carga pontual, foi necessário iniciar o processo de cálculo na expressão (5), correlacionando o índice de carga pontual com a resistência à compressão uniaxial. Na Fig. 37 é exemplificado a configuração deste método para a *seekBar* relativa à resistência à compressão uniaxial.

```
sb1.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        double peso = -0.000163 * progress * progress + 0.102614 * progress;
        tv4.setText(String.valueOf(String.format("%2.0f", peso)));
        et1.setText(String.valueOf(progress));

    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});
```

Fig. 37 – Método *setOnSeekBarChangeListener*

De forma inversa, implementou-se o método *addTextChangedListener* a cada *editText*, criando uma sincronização automática entre o valor editado na caixa de texto e a posição do *seekBar*. Na Fig. 38 é apresentado a configuração deste método para as ferramentas referentes à introdução da resistência à compressão uniaxial.

```
et1.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
        try {
            sb1.setProgress(Integer.parseInt(charSequence.toString()));
            et1.setSelection(et1.getText().length());
        } catch (Exception ex) {}
    }

    @Override
    public void afterTextChanged(Editable editable) {

    }
});
```

Fig. 38 – Método *addTextChangedListener*

De forma a melhorar a utilização da aplicação, optou-se por inserir botões de ação, tornando-se necessário a implementação de dois métodos. O primeiro, o *View.OnClickListener*, informa a atividade da existência de ferramentas associadas ao toque, enquanto o segundo, o *setOnClickListener*, permite definir a ação a executar quando cada botão é selecionado. Assim, inseriu-se o botão "Apagar" que limpa os dados inseridos pelo utilizador e o botão ajuda que, através de mensagem, exibe as instruções para a correta utilização da aplicação. Na Fig. 39 é apresentada a função desempenhada por cada botão, quando solicitados.

```
@Override
public void onClick(final View v) {
    switch (v.getId()) {
        case R.id.button: {



            if (radioButton.isChecked()) {
                Double d = new Double(0);
                int i = d.intValue();
                sb2.setProgress(i);
                et2.setText("");
                tv7.setText("");
            }

            if (radioButton2.isChecked()) {
                Double d = new Double(0);
                int i = d.intValue();
                sb1.setProgress(i);
                et1.setText("");
                tv4.setText("");
            }
        }

        break;

        case R.id.imageButton:{
            AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
            dlg.setTitle("Resistência da Rocha Intacta");
            dlg.setMessage("Insira o valor da Carga Pontual ou da " +
                    "Resistência à Compressão Uniaxial." +
                    "\nMova a barra ou digite o valor pretentido.");
            dlg.show();
        }
        break;
    }
}
```

Fig. 39 – Configuração do botão "Limpar" e ajuda

Na Fig. 40 é apresentado o separador criado, tendo a aplicação sido executada num dispositivo de 7 polegadas, suportado pelo API 23 do *Android*. Apesar de não ser necessário para este dispositivo, inseriu-se o componente *scroll* para evitar incompatibilidades com ecrãs de menor dimensão.

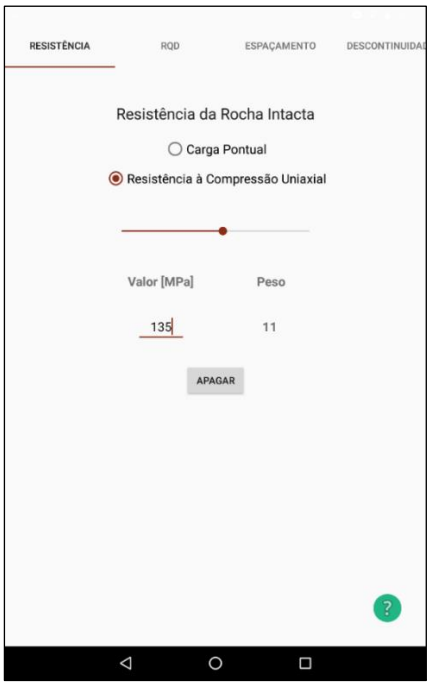


Fig. 40 – Separador "Resistência da Rocha Intacta"

4.3.4. *ROCK QUALITY DESIGNATION – TAB2*

Neste separador é avaliada, em percentagem, a fraturação do maciço rochoso através do índice RQD. Assim, optou-se novamente pela utilização de um *seekBar* onde, para uma uniformidade na aplicação, se utilizou o modelo de *sublayout* do separador anterior. A preferência pela utilização de funções contínuas no cálculo do peso manteve-se, tendo-se utilizado a expressão (16) no método *setOnSeekBarChangeListener*. Na Fig. 41 é apresentado o separador criado, para o qual se voltou a inserir os botões "Apagar" e ajuda.

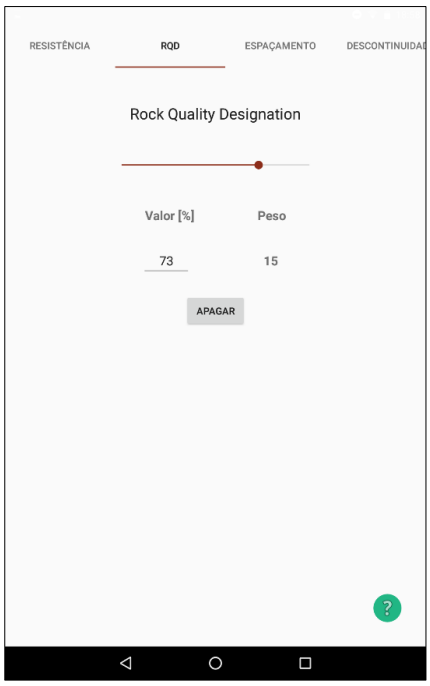


Fig. 41 – Separador "*Rock Quality Designation*"

4.3.5. ESPAÇAMENTO ENTRE DESCONTINUIDADES – *TAB3*

Neste terceiro separador é inserido o espaçamento entre descontinuidades onde, por ser uma informação quantitativa, se manteve a utilização do modelo de *sublayout* utilizado anteriormente, bem como os métodos associados. O cálculo do peso foi determinado através da expressão (17), sendo apresentado na Fig. 42 o resultado do separador.

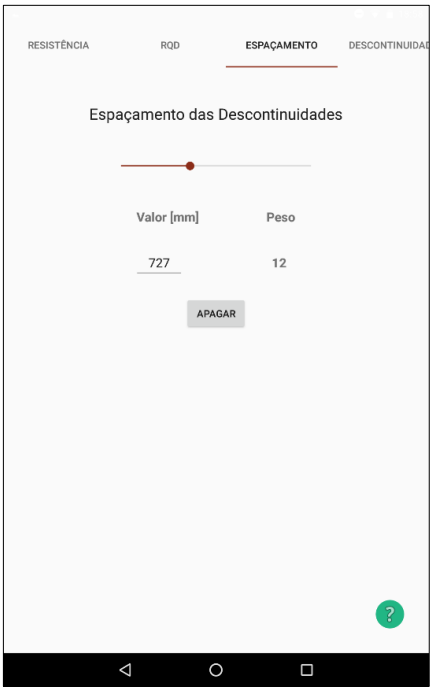


Fig. 42 – Separador "Espaçamento das Descontinuidades"

4.3.6. CARACTERÍSTICAS DAS DESCONTINUIDADES – *TAB4*

No quarto separador a descontinuidade é caraterizada quanto ao seu comprimento, abertura, rugosidade, enchimento e alteração. Como referido no Capítulo 2, Bieniawski definiu uma caraterização simples, criando diferentes descrições padrão (Tabela 6), e uma caraterização detalhada, permitindo uma melhor descrição de cada aspeto (Tabela 7). Desta forma, utilizaram-se dois *radioButton* (Fig. 43) que, analogamente ao separador Tab1, a sua seleção torna visível o *sublayout* associado, permitindo ao utilizador escolher como caraterizar a descontinuidade.

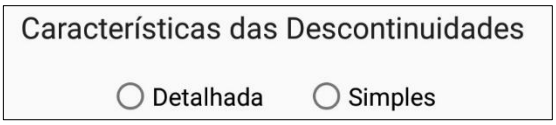


Fig. 43 – Inserção de *radioButtons* para caracterizar as descontinuidades

Relativamente à caraterização da descontinuidade de forma simples, decidiu-se utilizar novamente o *radioButton*, tendo-se inserido esta ferramenta associada a cada descrição padrão. Assim, invocou-se novamente o método *setOnClickListener* onde, em vez de alterar a visibilidade de *sublayouts* como nos exemplos anteriores, apresentava numa *textView* o peso correspondente à opção selecionada. Na Fig. 44 é apresentado a configuração do método mencionado, onde se verifica que através da seleção do *radioButton* é apresentado o respetivo peso na *textView*.

```
radioButton6.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        double peso = 25;
        tv35.setText(String.valueOf(String.format("%2.0f", peso)));

    }
});
```

Fig. 44 – Método *setOnClickListener*

O resultado deste *sublayout* é visível na Fig. 45, no qual são apresentadas as descrições padrão existentes e, através do método *setOnClickListener*, é atribuído o peso associado à descrição escolhida.

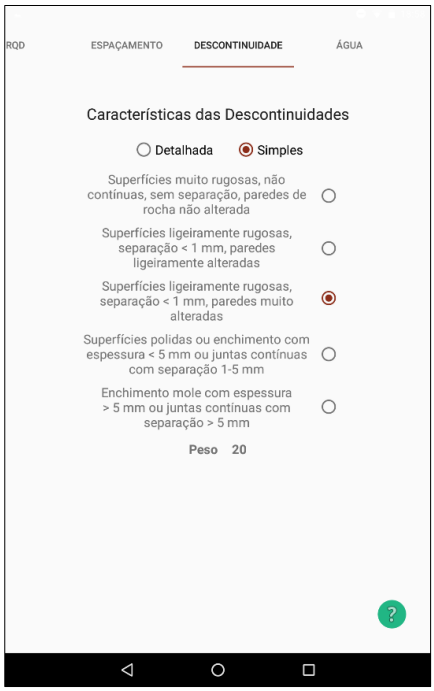


Fig. 45 – Separador "Características das Descontinuidades" – caraterização simples

Relativamente à caraterização da descontinuidade de forma detalhada verificou-se que cada caraterística pode ser definida de diferentes formas. Por isso, recorreu-se à utilização de *spinners*, componente que se assemelha a uma caixa de texto e que, através do toque, apresenta uma lista de opções disponíveis. Na Fig. 46 verifica-se a utilidade do componente, pois mantêm a lista de opções disponíveis minimizada, apresentando-a quando solicitada.

Fig. 46 – Ferramenta *spinner*

Para a configuração deste *sublayout* começou-se por definir *strings* (tipos de dados para representar sequências de caracteres) com a lista de opções disponíveis para cada característica, sendo exemplificado na Fig. 47 a criação das opções relativa à rugosidade. Estes dados são declarados no ficheiro *strings* (Anexo I).

```
<string-array name="rugosidade">
    <item>Escolha</item>
    <item>Muito Rugosa</item>
    <item>Rugosa</item>
    <item>Ligeiramente Rugosa</item>
    <item>Lisa</item>
    <item>Espelhada</item>
</string-array>
```

Fig. 47 – Lista de *string* da característica rugosidade

Posteriormente, associou-se a lista de *strings* ao *spinner* correspondente através da utilização do método *setAdapter*, permitindo a apresentação da lista quando a ferramenta era solicitada. De forma a que a aplicação interpretasse a opção escolhida e atribuísse o peso correspondente, utilizou-se o método *setOnItemSelectedListener*. Na Fig. 48 é visível, para a característica da rugosidade, a implementação do método *setAdapter* e *setOnItemSelectedListener*.

```
spinner3 = (Spinner) findViewById(R.id.spinner3);
adapter3 = ArrayAdapter.createFromResource(this, R.array.rugosidade, android.R.layout.simple_spinner_item);
adapter3.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner3.setAdapter(adapter3);
spinner3.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        String rugosidade = spinner3.getSelectedItem().toString();
        if (rugosidade.equals("Muito Rugosa")) {
            double peso = 6;
            tv24.setText(String.valueOf(String.format("%1.0f", peso)));
        } else if (rugosidade.equals("Rugosa")) {
            double peso = 5;
            tv24.setText(String.valueOf(String.format("%1.0f", peso)));
        } else if (rugosidade.equals("Ligeiramente Rugosa")) {
            double peso = 3;
            tv24.setText(String.valueOf(String.format("%1.0f", peso)));
        } else if (rugosidade.equals("Lisa")) {
            double peso = 1;
            tv24.setText(String.valueOf(String.format("%1.0f", peso)));
        } else if (rugosidade.equals("Espelhada")) {
            double peso = 0;
            tv24.setText(String.valueOf(String.format("%1.0f", peso)));
        } else if (rugosidade.equals("Escolha")) {
            double peso = 0;
            tv24.setText(String.valueOf(String.format("%1.0f", peso)));
        }

    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

Fig. 48 – Método *setAdapter* e *setOnItemSelectedListener*

Para facilitar a interpretação do utilizador quanto aos dados inseridos, além da exibição do peso parcial de cada aspeto em análise, é apresentado a soma dos mesmos, indicando o peso total do parâmetro relativo às características das descontinuidades. Ao contrário do *sublayout* anterior, optou-se por inserir o botão "Apagar" que, além de remover os pesos das diferentes análises, recoloca a opção "Escolha" nos diferentes *spinners*. A apresentação do separador é visível na Fig. 49.

Fig. 49 – Separador "Características das Descontinuidades" – caraterização detelhada

4.3.7. PRESENÇA DE ÁGUA – *TAB5*

Neste separador é caraterizada a influência da água, último parâmetro avaliado por Bieniawski para o cálculo do RMR básico. Uma vez que este pode ser caraterizado de forma qualitativa ou quantitativa, este último por intervalo de valores, optou-se pelo uso de *spinners*.

Assim, foram inseridos três *spinners*, permitindo caraterizar a existência de água segundo uma de três condições: pelas condições gerais, pelo caudal por 10 metros de túnel ou através do quociente entre a pressão e a tensão máxima, tendo-se criado a lista de *strings* correspondentes de forma análoga ao separador anterior. Na implementação do método s*etOnItemSelectedListener* definiu-se que a alteração num dos *spinners* implicava que os restantes retomassem a opção "Escolha". Com este pormenor evita-se que o utilizador caracterize a existência de água por diferentes métodos de avaliação, escolhendo opções associadas a pesos diferentes, em que o valor apresentado não seria válido. Na Fig. 50 é exemplificada a configuração do método para o *spinner* relativo ao caudal, verificando-se que a manipulação desta ferramenta implica a alteração das restantes para a opção inicial.

```
spinner6.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        String fluxo = spinner6.getSelectedItem().toString();
        if (fluxo.equals("Nulo")) {
            double peso = 15;
            tv41.setText(String.valueOf(String.format("%2.0f", peso)));
            spinner7.setSelection(adapter7.getPosition("Escolha"));
            spinner8.setSelection(adapter8.getPosition("Escolha"));
        } if (fluxo.equals("<10 l/min")) {
            double peso = 10;
            tv41.setText(String.valueOf(String.format("%2.0f", peso)));
            spinner7.setSelection(adapter7.getPosition("Escolha"));
            spinner8.setSelection(adapter8.getPosition("Escolha"));
        } if (fluxo.equals("10 - 25 l/min")) {
            double peso = 7;
            tv41.setText(String.valueOf(String.format("%1.0f", peso)));
            spinner7.setSelection(adapter7.getPosition("Escolha"));
            spinner8.setSelection(adapter8.getPosition("Escolha"));
        } if (fluxo.equals("25 - 125 l/min")) {
            double peso = 4;
            tv41.setText(String.valueOf(String.format("%1.0f", peso)));
            spinner7.setSelection(adapter7.getPosition("Escolha"));
            spinner8.setSelection(adapter8.getPosition("Escolha"));
        } if (fluxo.equals(">125 l/min")) {
            double peso = 0;
            tv41.setText(String.valueOf(String.format("%1.0f", peso)));
            spinner7.setSelection(adapter7.getPosition("Escolha"));
            spinner8.setSelection(adapter8.getPosition("Escolha"));
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

Fig. 50 – Método *setOnItemSelectedListener*

Concluída a configuração dos três *spinners*, inseriu-se o botão "Apagar" que, além de apagar o valor do peso, volta a apresentar a opção "Escolha" no *spinner* seleccionado. O separador criado é apresentado na Fig. 51.

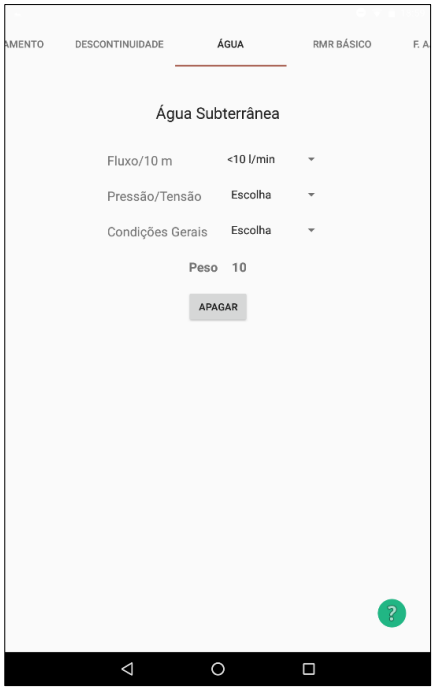


Fig. 51 – Separador Presença de Água

4.3.8. RMR BÁSICO – *TAB6*

Concluída a caraterização dos diferentes parâmetros propostos por Bieniawski, seguiu-se o cálculo do RMR básico, através da soma dos diferentes pesos atribuídos.

Ao longo das seis atividades desenvolvidas verificou-se que o método mais utilizado foi o *findViewById*, declarando as ferramentas introduzidas e permitindo a sua manipulação. É através deste que se inicia a configuração, por exemplo, da exibição do valor de uma *seekBar* numa *editText* (Fig. 36) ou da leitura do valor existente em várias *textView* e sua posterior soma (Fig. 49).

No entanto, a utilização deste método está limitada à manipulação de ferramentas existentes no *layout* associado à atividade a configurar. Ou seja, na configuração da atividade Tab6, este método apenas permite manipular ferramentas existentes no *layout* Tab6, não sendo possível invocar ferramentas pertencentes a outros separadores, como seria necessário para a recolha do valor dos diferentes pesos.

De forma a contornar esta limitação, utilizou-se o método *getSharedPreferences*. O funcionamento deste é mais complexo do que do *findViewById*, uma vez que é necessário definir na atividade de origem a partilha com a restante aplicação da variável pretendida, sendo posteriormente invocada na atividade de destino. Na Fig. 52 é exemplificado a implementação do método para o parâmetro RQD, verificando-se que a variável partilhada está associada ao *textView* onde o peso é exibido e, na Fig. 53, apresenta-se a receção da mesma variável na atividade Tab6.

```
double peso = 0.000616 * progress * progress + 0.113103 * progress + 3;
tv11.setText(String.valueOf(String.format("%6.0f", peso)));
SharedPreferences pesodois = getSharedPreferences("pesodois", 0);
SharedPreferences.Editor editor2 = pesodois.edit();
editor2.putString("rqd", tv11.getText().toString());
editor2.commit();
```

Fig. 52 – Método *SharedPreferences* na partilha da variável “rqd” com as restantes atividades

```
SharedPreferences pesodois = getSharedPreferences("pesodois", 0);
String rqd = pesodois.getString("rqd", "0");
```

Fig. 53 – Método *SharedPreferences* na receção da variável "rqd" na atividade Tab6

Este método foi implementado no cálculo do peso de cada parâmetro permitindo que a sincronização entre o peso exibido e a variável partilhada seja automática. Assim, se em componentes como o *seekBar* é suficiente inserir este método no *setOnSeekBarChangeListener* e associá-lo à *textView* pretendida, noutras, como o *radioButton*, é necessário implementá-lo em todas as opções disponíveis. Esta particularidade é apresentada na Fig. 54, onde se exemplificou para um *radioButton*, associado a uma caraterização padrão da descontinuidade, a aplicação deste método.

```
radioButton5.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        double peso = 30;
        tv35.setText(String.valueOf(String.format("%2.0f", peso)));
        SharedPreferences pesoquatro = getSharedPreferences("pesoquatro", 0);
        SharedPreferences.Editor editor4 = pesoquatro.edit();
        editor4.putString("des", tv35.getText().toString());
        editor4.commit();

    }
});
```

Fig. 54 – Método *SharedPreferences*

Posteriormente à configuração do método *getSharedPreferences* para os pesos dos cinco parâmetros existentes, procedeu-se ao cálculo do RMR básico onde, através do toque no botão "RMR básico", este é exibido numa *editText*. O uso de uma *editText* em detrimento de uma *textView* justifica-se com a possibilidade do utilizador conhecer previamente o valor do RMR básico, permitindo-lhe que inicie a utilização da aplicação no sexto separador. Na Fig. 55 verifica-se o cálculo do RMR básico através da soma do peso dos 5 parâmetros.

```
double a = Double.parseDouble(rcu);
double b = Double.parseDouble(rqd);
double c = Double.parseDouble(esp);
double d = Double.parseDouble(des);
double e = Double.parseDouble(agu);

double total = a + b + c + d + e;


et5.setText(String.valueOf(String.format("%2.0f", total)));
```

Fig. 55 – Cálculo do índice RMR básico

Uma particularidade deste método é manter a variável guardada na aplicação após o encerramento da mesma, ao contrário das ferramentas que não memorizam as opções introduzidas. Assim, de modo a evitar que o utilizador não caraterizasse um parâmetro e, no cálculo do RMR básico, fosse utilizado o peso relativo à última utilização, optou-se por reiniciar todas as variáveis quando a aplicação fosse iniciada. Deste modo, configurou-se também as diferentes variáveis na atividade principal, associando-as ao valor zero de cada vez que a aplicação fosse iniciada. Tal é verificado na Fig. 56, onde se exemplificou, com a variável do peso da resistência da rocha intacta, a atribuição do valor zero.

```
SharedPreferences pesoum = getSharedPreferences("pesoum", 0);
SharedPreferences.Editor editor1 = pesoum.edit();
editor1.putString("rcu", "0");
editor1.commit();
```

Fig. 56 – Exemplificação da atribuição do peso zero a uma variável no início da aplicação

Concluída a configuração do separador com os objetivos idealizados para o mesmo, verificou-se que o *layout* apresentado continha pouca informação. Deste modo, decidiu-se completá-lo de forma a apresentar as características do maciço rochoso, através da informação inserida nos separadores anteriores. Para tal foi necessário utilizar o método *getSharedPreferences* em dez novas variáveis, permitindo apresentar através de uma tabela as informações inseridas em cada parâmetro. Na Fig. 57 é exibido o separador criado.

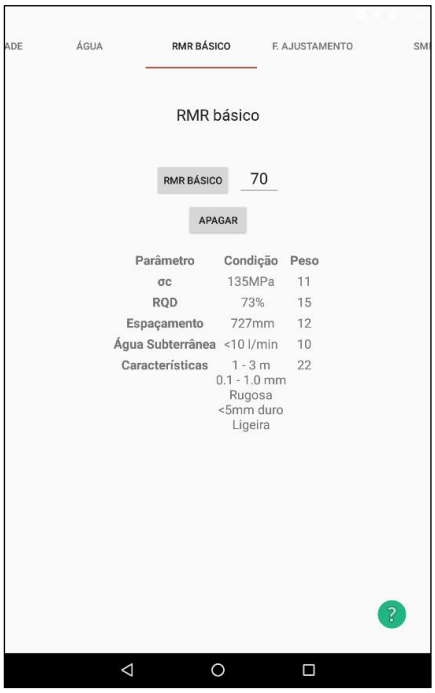


Fig. 57 – Separador RMR básico

4.3.9. FATORES DE AJUSTAMENTO – *TAB7*

Uma vez calculado o índice RMR básico, era necessário efetuar as correções propostas por Romana, adaptando a classificação de Bieniawski a taludes. Assim, para o cálculo dos diferentes fatores de ajustamento, foi necessário definir a orientação da face do talude e da descontinuidade, o método de escavação ocorrido no talude e o tipo de rotura em análise.

Deste modo, começou-se por inserir dois *radioButton* (Fig. 58) que, associados a diferentes *sublayouts*, possibilitam ao utilizador definir manualmente as orientações, com informação pré recolhida, ou de forma automática, recorrendo à bússola de geólogo.

Fig. 58 – Inserção de *radioButtons* para a caracterização das orientações

Relativamente ao *sublayout* de introdução manual das orientações optou-se pelo uso de *seekBars*, recorrendo-se a dois destes componentes para caraterizar cada orientação, o que permitiu definir a sua direção e inclinação. A legenda da orientação inserida é exibida numa *textView* sob a representação de *strike*/pendor da reta de maior declive e *dip direction*/*dip*, possibilitando ao utilizador diferentes formas de leitura. Na Fig. 59 é exemplificada a introdução da orientação do talude.

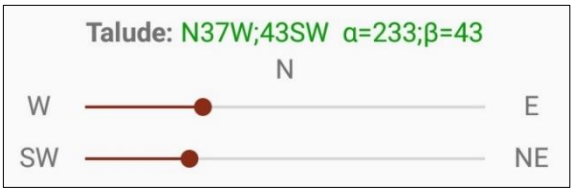


Fig. 59 – *Sublayout* de introdução manual da orientação do talude

Devido a cada orientação ser definida por dois componentes independentes, foi necessário implementar que cada *seekBar* interpretasse o valor da *seekBar* correspondente, apresentando a orientação de forma correta. Assim, comparando a Fig. 60 com a figura anterior é possível verificar que, alterando apenas a direção do *strike*, as possíveis direções da reta de maior declive são atualizadas.

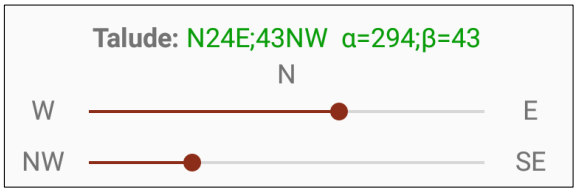


Fig. 60 – *Sublayout* de introdução manual da orientação do talude

Relativamente à seleção do *radioButton* "Automático", torna-se visível um *sublayout* que contém o botão "Bússola". Através do toque neste botão, a aplicação inicia um ciclo de quatro novas atividades, recolhendo a orientação do talude e, posteriormente, da descontinuidade.

Deste modo, na primeira e terceira atividade (Tab9 e Tab11) é apresentado uma bússola que indica a direção do talude e da descontinuidade, respetivamente. Como tal, foi criada a imagem de uma bússola (Fig. 61) e utilizado o conjunto de métodos fornecidos pela *Android Developer* para a recolha de orientações (Android Developers, 2016c), recorrendo às capacidades do acelerómetro. A utilização deste sensor não cria qualquer incompatibilidade na aplicação, uma vez que está presente na totalidade dos dispositivos móveis.

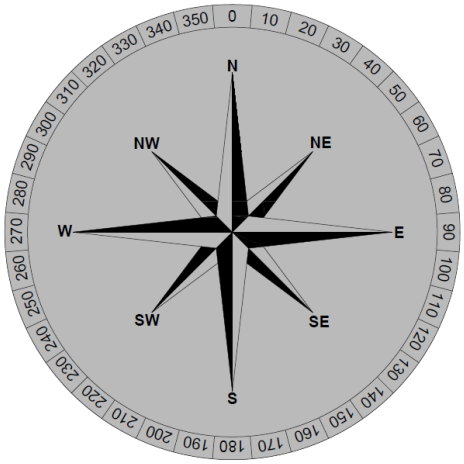


Fig. 61 – Imagem da bússola

No entanto foi necessário modificar o conjunto de métodos importados, adaptando-se à função pretendida. Assim, inseriu-se uma animação que rodava a imagem da bússola com o ângulo medido, apresentando este valor numa *textView* com a representação de *strike*. Na Fig. 62 é exibido o *layout* da atividade criada para a medição da direção do talude, verificando-se o funcionamento da bússola.

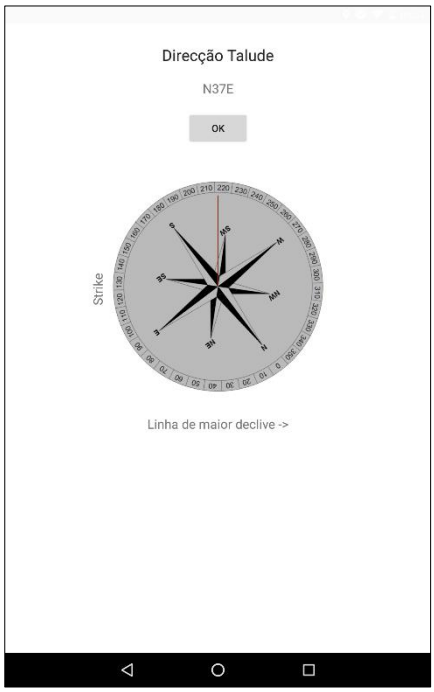


Fig. 62 – Funcionamento da bússola

Dado que a utilização desta capacidade procura simular a medição de direções através de uma bússola, procurou-se impor um processo de medição semelhante. Deste modo, a correta direção é aferida colocando o dispositivo na horizontal, tendo uma das arestas encostada ao *strike*. Para além disso, e de forma a que fosse reconhecida a direção da linha de maior declive, impôs-se que a dita aresta fosse a esquerda, de modo a que a *app* assumisse que a direção do pendor tenderia para o lado direito do aparelho. Na Fig. 63 é demonstrada a correta utilização desta capacidade na medição da direção de um talude rochoso.

Fig. 63 – Medição da orientação de um talude rochoso

De forma a concluir a caraterização das orientações procedeu-se à medição da inclinação do talude e da descontinuidade, utilizando-se a segunda e quarta atividades criadas (Tab10 e Tab12). Deste modo, criou-se a imagem de um transferidor (Fig. 64) e importou-se os métodos sugeridos pela *Android Developer* para a medição da inclinação (Android Developers, 2016d). No entanto, este conjunto baseia-se na utilização do giroscópio, um sensor normalmente presente em dispositivos recentes, mas pouco frequente nos antigos, tornando esta função indisponível para alguns aparelhos.

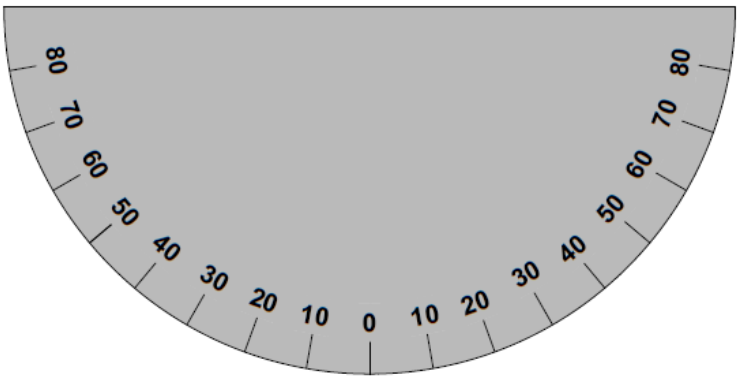


Fig. 64 – Imagem do transferidor

De forma análoga à configuração anterior, foi necessário adaptar o método importado à função pretendida. Devido às dimensões da figura, esta ajusta-se de melhor forma ao *layout* se for inserida na vertical, permitindo uma melhor análise ao utilizador. Assim, e uma vez que o método mede o ângulo relativamente à vertical, implementou-se que a inclinação seria nula quando o dispositivo se encontrasse na horizontal. Além disto, inseriu-se a imagem de uma agulha no centro do transferidor que simula a ação de um fio de prumo, sofrendo uma rotação de acordo com a inclinação medida. Na Fig. 65 é exemplificado o funcionamento do clinómetro, medindo a inclinação do talude.

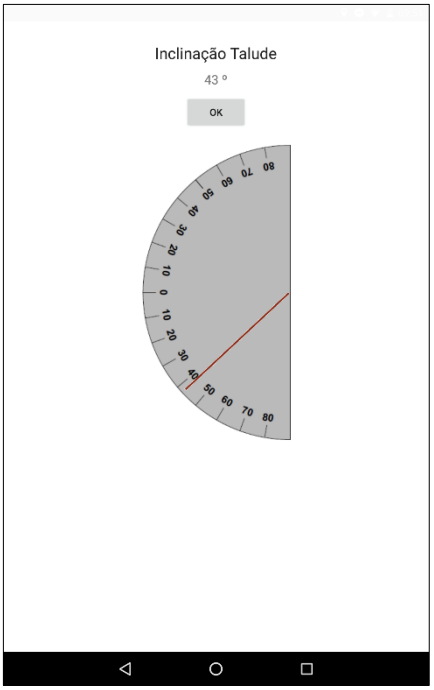


Fig. 65 – Funcionamento do clinómetro

A utilização desta função procura, mais uma vez, simular o uso de uma bússola de geólogo. Assim, a inclinação é medida enconstando a aresta esquerda do dispositivo no plano em análise, procurando que este contacto seja perpendicular e que o aparelho esteja paralelo à linha de maior declive. Na Fig. 66 é demonstrada a utilização da função clinómetro na medição da inclinação de um talude rochoso.

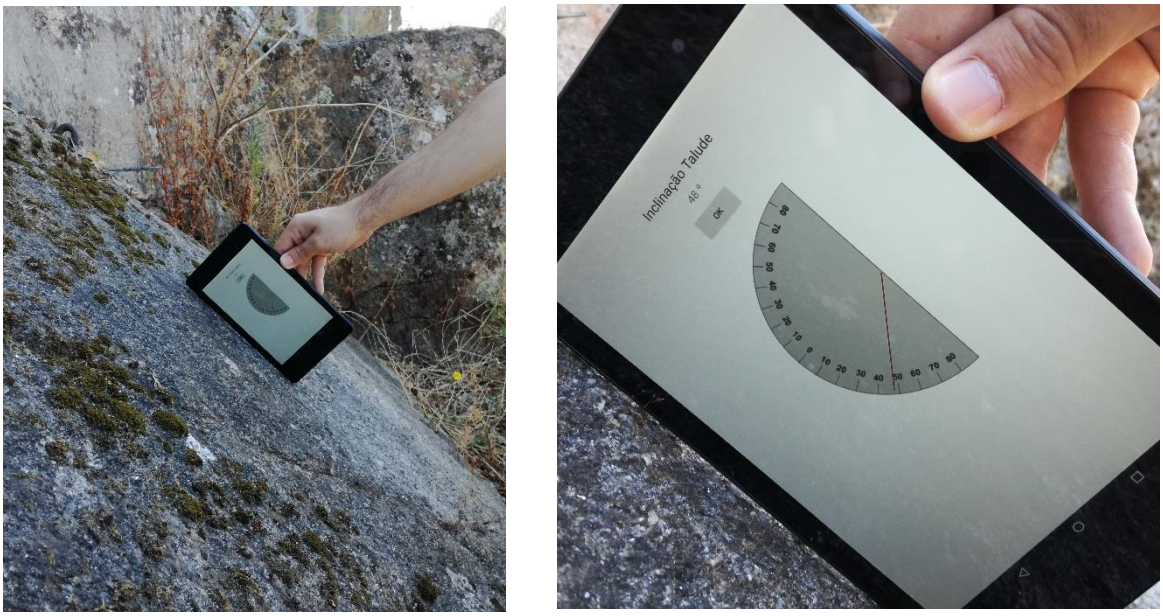


Fig. 66 – Medição da inclinação de um talude rochoso

Uma vez configuradas as quatro atividades para a medição das orientações, implementou-se o método *getSharedPreferences* de forma a importar para a atividade Tab7 os ângulos medidos, apresentando as orientações através das duas representações já mencionadas. Na Fig. 67 é apresentado o *sublayout* de introdução automática das orientações, após a recolha das mesmas.

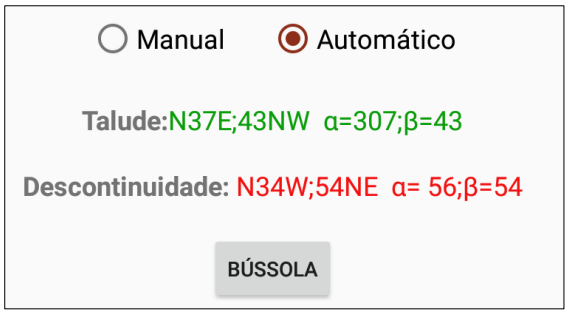


Fig. 67 – *Sublayout* de introdução automática das orientações

Concluída a configuração dos dois *sublayout* para a introdução das orientações decidiu-se representar a projeção hemisférica, permitindo uma melhor análise ao utilizador. No entanto, uma vez que o *software* não possui ferramentas para esta função, foi necessário instalar o *GraphView* (Graph View, 2016), uma extensão para o *Android Studio* que permite a criação de gráficos. Assim, começou-se por configurar o gráfico apresentado, definindo-se os seus eixos e inserindo-se a circunferência limite da projeção (Fig. 68).

```
final GraphView graph = (GraphView) findViewById(R.id.graph);

graph.getViewport().setXAxisBoundsManual(true);
graph.getViewport().setMaxX(90);
graph.getViewport().setMinX(-90);

graph.getViewport().setYAxisBoundsManual(true);
graph.getViewport().setMinY(-90);
graph.getViewport().setMaxY(90);

graph.getGridLabelRenderer().setNumHorizontalLabels(3);
graph.getGridLabelRenderer().setNumVerticalLabels(3);

int count = 2000;
DataPoint[] values = new DataPoint[count];
for (int i=0; i<count; i++) {
    double x = 90 * Math.cos(i);
    double y = 90 * Math.sin(i);
    DataPoint v = new DataPoint(x, y);
    values[i] = v;
}
PointsGraphSeries<DataPoint> circulo = new PointsGraphSeries<>(values);
circulo.setSize(1);
circulo.setColor(Color.BLACK);
graph.addSeries(circulo);

graph.getGridLabelRenderer().setVerticalLabelsVisible(false);
graph.getGridLabelRenderer().setHorizontalLabelsVisible(false);
```

Fig. 68 – Configuração do gráfico

Devido à elevada complexidade da definição matemática da projeção do círculo maior do plano, foi adotada uma simplificação da representação, desenhando a linha de *strike* e o ponto correspondente à extremidade da linha de maior declive. Inseriu-se ainda o polo da normal, onde, para que não haja equívocos quanto ao outro ponto representado, se indica com um pequeno traço a direção do pendor. Uma vez que não era possível distinguir as representações efetuadas, optou-se pela sua diferenciação ao apresentarem cores diferentes, correspondendo às utilizadas nas legendas das orientações. Esta representação gráfica é apresentada na Fig. 69.

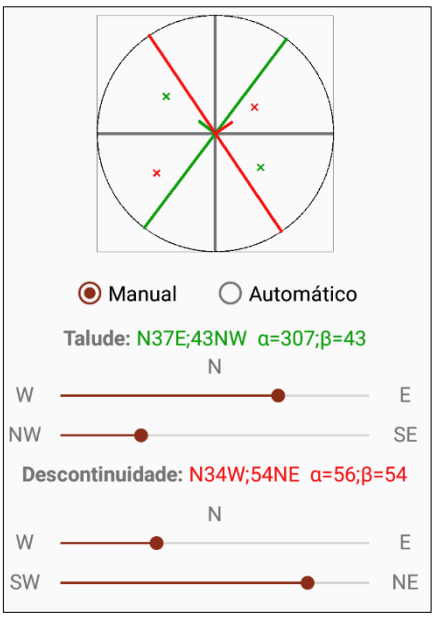


Fig. 69 – Representação gráfica do talude e da descontinuidade

Posteriormente à caraterização das orientações foi necessário definir o método de escavação e o tipo de rotura em análise, utilizando-se a ferramenta *spinner*. De forma análoga às configurações anteriores desta ferramenta, implementou-se o método *setAdapter* para a exibição da lista de *strings* associada a cada *spinner*. Na Fig. 70 é apresentado o modelo escolhido para a introdução do método de escavação e do tipo de rotura.

Fig. 70 – *Spinners* para a escolha do método de escavação e do tipo de rotura

Concluída a introdução da informação necessária, procedeu-se à configuração do cálculo dos fatores de ajustamento. Uma vez que o método de cálculo varia com o tipo de rotura em análise, começou-se por implementar blocos de condições lógicas (*if*), permitindo à aplicação diferenciar as tarefas a realizar com a opção de *spinner* selecionada. Deste modo, para cada tipo de rotura, aplicou-se esta estratégia e definiu-se as expressões para o cálculo dos parâmetros *A*, *B* e *C*, necessários para a determinação dos fatores de ajustamento F_1, F_2 e F_3, respetivamente. Visto que o fator de ajustamento F_4 é independente do tipo de rotura, não foi necessário associá-lo a nenhuma condição *if*. Assim, através do toque no botão "Calcular" a aplicação executa o processo de cálculo implementado, apresentando os diferentes fatores de ajustamento em *textViews*. Na Fig. 71 é exemplificado a configuração do bloco de condição para o tipo de rotura planar, verificando-se a utilização do processo de cálculo apresentado no Capítulo 2.

```
if (esc.equals("Planar")){
    double A = Math.abs(aj-as);
    double B = bj;
    double C = bj-bs;
    if (A<=90){
        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A - 17)) * 180/Math.PI;
        tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
    }
    if (A>90 && A<=180){
        double A1 = 180 - A;
        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 - 17)) * 180/Math.PI;
        tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
    }
    if (A>180 && A<=270){
        double A1 = A - 180;
        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 - 17)) * 180/Math.PI;
        tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
    }
    if (A>270 && A<=360){
        double A1 = 360 - A;
        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 - 17)) * 180/Math.PI;
        tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
    }

    double f2i = 0.5625 + 0.00512821 * Math.atan(0.17 * bj - 5) * 180/Math.PI;
    double f3i = - 30 + 0.333333 * Math.atan(bj-bs) * 180/Math.PI;
    tv88.setText(String.valueOf(String.format("%2.3f",f2i)));
    tv91.setText(String.valueOf(String.format("%2.3f",f3i)));
}
```

Fig. 71 – Cálculo dos fatores de ajustamento para a rotura planar, usando o método "*if*"

Uma vez definido o processo de cálculo dos fatores de ajustamento para cada tipo de rotura, concluiu-se a configuração deste separador, sendo apresentado na Fig. 72.

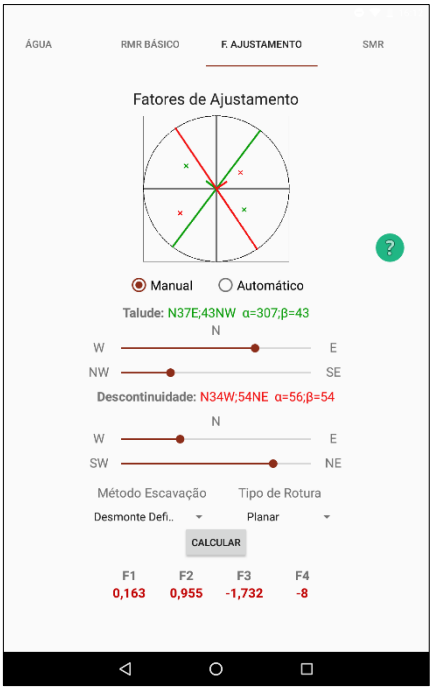


Fig. 72 – Separador "Fatores de Ajustamento"

4.3.10. SMR – *TAB8*

Concluído o cálculo do valor do RMR básico e dos fatores de ajustamento, torna-se possível determinar o índice SMR. Deste modo, aplicou-se novamente o método *getSharedPreferences*, importando para a atividade Tab8 as cinco variáveis necessárias e recorreu-se à expressão (8) para calcular o índice (Fig. 73).

```
double a = Double.parseDouble(rmr);
double b = Double.parseDouble(fumf);
double c = Double.parseDouble(fdoisf);
double d = Double.parseDouble(ftresf);
double e = Double.parseDouble(fquatrof);
double f = (b * c * d) + e;
double smr = a + f;

tv114.setText(String.valueOf(String.format("%2.0f", smr)));
```

Fig. 73 – Cálculo do índice SMR

Através do toque no botão "Carregar" é apresentado o resultado obtido, bem como a classificação do talude e o suporte recomentado. Para tal, recorreu-se novamente a um bloco de condições *if* de forma a identificar o valor do índice e para o caracterizar de acordo com a informação descrita na Tabela 15 e na Tabela 16. Na Fig. 74 verifica-se a utilização deste método para apresentar as informações retiradas da Tabela 15.

```
if (smr <= 20) {
    tv116.setText("Muito má");
    tv117.setText("Completamente instável");
    tv118.setText("Planar ou circular");
}
if (smr > 20 && smr <= 40) {
    tv116.setText("Má");
    tv117.setText("Instável");
    tv118.setText("Planar ou em cunha");
}
if (smr > 40 && smr <= 60) {
    tv116.setText("Normal");
    tv117.setText("Parcialmente estável");
    tv118.setText("Algumas juntas ou" +
            "\nmuitas cunhas");
}
if (smr > 60 && smr <= 80) {
    tv116.setText("Boa");
    tv117.setText("Estável");
    tv118.setText("Alguns blocos");
}
if (smr > 80 && smr <= 100) {
    tv116.setText("Muito boa");
    tv117.setText("Completamente estável");
    tv118.setText("Nenhuma");
}
```

Fig. 74 – Utilização de condições "*if*" para apresentação da classificação do talude

De forma a completar a informação apresentada ao utilizador, importou-se um conjunto de métodos que identifica a localização do dispositivo (GeoCoder, 2016), exibindo-a em coordenadas geográficas. Além disso, mediante a ligação à internet, é ainda identificada a morada do local. Para tal foi necessário declarar no ficheiro *AndroidManifest* (Anexo I) a permissão para a aplicação aceder à localização do dispostivo e à internet (Anexo I). Na Fig. 75 é apresentado o separador criado, verificando-se a completa caraterização do talude, bem como o suporte recomendado a aplicar.

Fig. 75 – Separador "SMR"

4.4. ANÁLISE COMPARATIVA DA APLICAÇÃO

De forma a verificar a correta implementação do código mencionado, procedeu-se à utilização da aplicação desenvolvida para a classificação do maciço rochoso definido na Tabela 17, permitindo a comparação do resultado obtido com os das aplicações analisadas. Na Fig. 76 é apresentado o índice RMR básico do maciço rochoso em estudo, verificando-se a correta utilização da classificação de Bieniawski, uma vez que a ligeira diferença observada se deve à utilização de funções contínuas no cálculo do peso dos parâmetros da resistência à compressão uniaxial, do índice RQD e do espaçamento entre descontinuidades. Comparando as cinco aplicações utilizadas para o cálculo da classificação RMR, a aplicação desenvolvida parece ser a mais intuitiva, facilitando a interpretação do utilizador quanto aos parâmetros a introduzir, sendo obtido um índice de qualidade do maciço rochoso semelhante ao esperado.

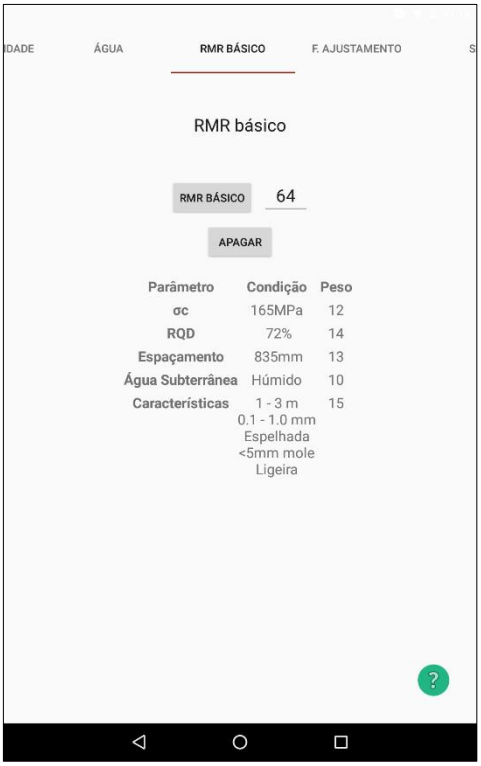


Fig. 76 – Cálculo do índice RMR básico

Uma vez que não existem aplicações *Android* destinadas ao cálculo da classificação SMR, comparou-se a utilização da aplicação desenvolvida com a ferramenta *SMRTool*. Para tal, considerou-se que o maciço mencionado era um talude natural com orientação N62E;34SE, analisando-se a possibilidade de rotura planar para a existência de uma descontinuidade com a orientação N13W;53NE. Dado que tanto a aplicação desenvolvida como a *SMRTool* recorrem à utilização das funções contínuas propostas por Tomás para o cálculo dos fatores de ajustamento, era esperado que o índice SMR obtido fosse o mesmo. Na Fig. 77 e na Fig. 78 demonstra-se a utilização da *app* desenvolvida e da *SMRTool* no cálculo da classificação SMR, verificando-se a semelhança nos resultados obtidos para os fatores de ajustamento e índice SMR.

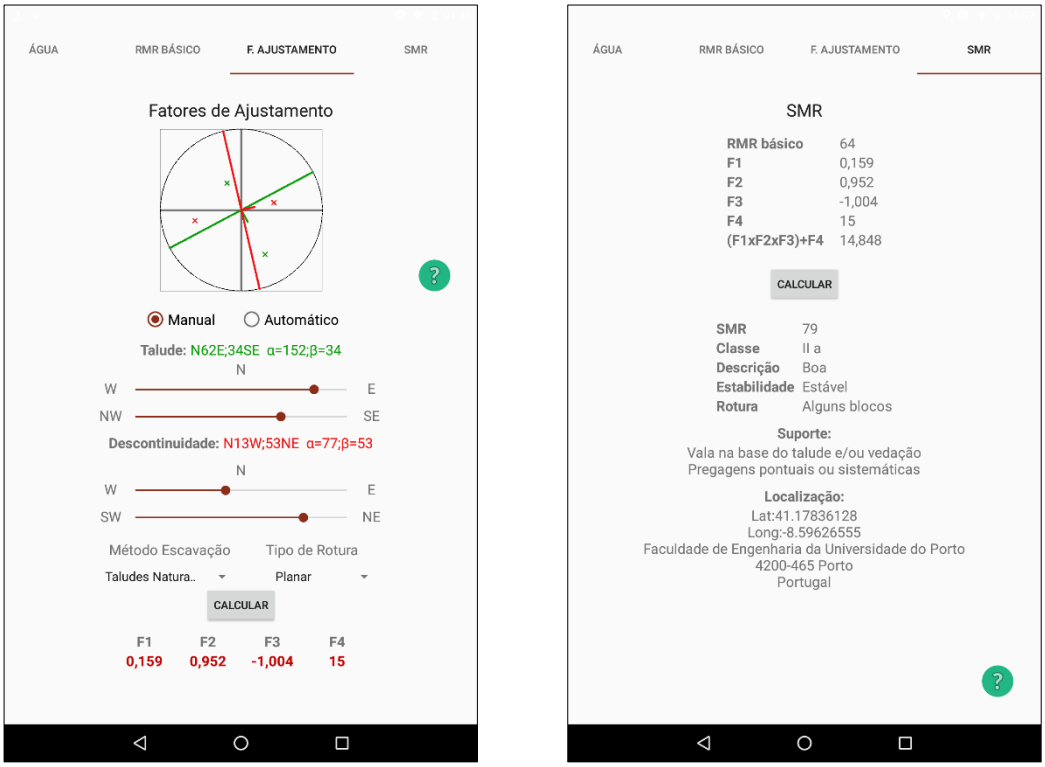


Fig. 77 – Cálculo do índice SMR através da *app* desenvolvida

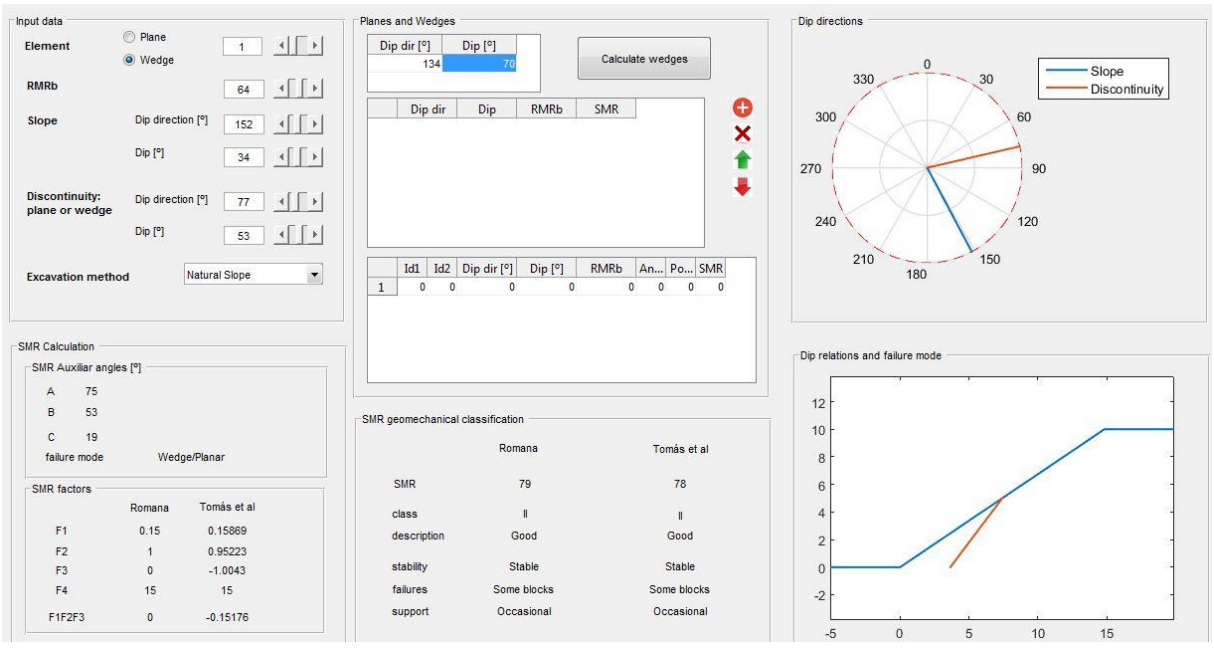


Fig. 78 – Cálculo do índice SMR através da *SMRTool*

4.5. PUBLICAÇÃO DA APLICAÇÃO

Concluído o desenvolvimento da aplicação proposta, decidiu-se partilhá-la aos utilizadores interessados, através da *Google Play Store*. Assim, começou-se por criar o ícone da aplicação (Fig. 79), sobrepondo a sigla da classificação calculada à imagem de um talude. Posteriormente foi criado o ficheiro apk, correspondendo ao ficheiro executável para dispositivos *Android*.

Fig. 79 – Ícone da aplicação

Devido à falta de experiência no desenvolvimento de aplicações e prevendo a existência de possíveis falhas na sua utilização, optou-se por iniciar a partilha do apk numa versão beta. Deste modo, o utilizador era informado que a aplicação ainda se encontrava em desenvolvimento, não sendo possível classifica-la. A partir dos comentários recebidos alterou-se significativamente a aplicação, tornando-a mais simples e de fácil utilização, correspondendo à aplicação descrita neste capítulo.

A versão atual da aplicação está disponível, de forma gratuita, no seguinte *link*:

https://play.google.com/store/apps/details?id=com.projecto.diogoassis.smr

5

CONSIDERAÇÕES FINAIS E PROPOSTAS PARA TRABALHOS FUTUROS

Com a realização deste trabalho foi abordada a análise de uma classificação geomecânica destinada a taludes rochosos, culminando com desenvolvimento de uma aplicação *Android* que permita, em obra, aplicar esta ferramenta de cálculo de forma expedita.

De entre as várias classificações de taludes existentes, optou-se pela utilização do sistema SMR uma vez que é a classificação que representa maior adoção internacional. Além disso, o facto de ser mencionada em livros sobre a estabilidade de taludes rochosos, no plano de estudos de cursos de engenharia ou em regulamentos técnicos de diferentes países sobre a classificação de taludes rochosos, demonstra a relevância deste sistema. No entanto, apesar da grande utilidade desta classificação, verificou-se que esta não tinha sido adaptada para aplicações móveis, demonstrando a singularidade da *app* desenvolvida. Assim, a aplicação criada permite ao utilizador, de uma forma simples e intuitiva, avaliar a estabilidade do talude rochoso. Para tal, começa por caraterizar o maciço rochoso, calculando o RMR básico, e, posteriormente, aplica as correções destinadas a taludes, determinando o índice SMR. Esta *app* é complementada com o recurso a capacidades do dispositivo móvel, nomeadamente na identificação da localização e na utilização de uma bússola de geólogo, permitindo definir as orientações do talude e da descontinuidade.

No entanto, dado o tempo necessário para a aprendizagem de bases em linguagem *Java* e para conhecer o funcionamento do programa *Android Studio* ser considerável, ocupando significativamente o tempo atribuído à realização da tese, não foi possível completar a aplicação com todas as funcionalidades idealizadas. Deste modo, identificam-se algumas limitações da *app* desenvolvida, podendo estas serem consideradas como propostas para desenvolvimento futuro. Nomeadamente, o facto de só ser possível fazer a análise individual das descontinuidades implica que o utilizador tenha que repetir o processo em número igual ao de descontinuidades existentes, sendo esta a principal limitação. Assim, deveria ser possível caraterizar várias descontinuidades simultaneamente, identificando qual a que provocaria maior instabilidade no talude. Para além disso, outra limitação detetada é a impossibilidade de utilizar de forma completa a bússola de geólogo, em dispositivos que não contenham o sensor giroscópio.

Apesar da aplicação criada possibilitar a caraterização de cada parâmetro em estudo, esta podia ser melhorada recorrendo a outras capacidades do dispositivo móvel, nomeadamente na medição de distâncias através da câmara. Para tal, seria necessário sobrepor a imagem apresentada na câmara com uma régua, em que a sua escala estaria dependente da distância entre o dispositivo e o parâmetro a medir.

Uma vez que a classificação SMR avalia pontualmente a estabilidade do talude rochoso, seria interessante que a aplicação avaliasse a estabilidade do talude ao longo do seu desenvolvimento. Com a

implementação de uma base de dados, as avaliações efetuadas seriam guardadas e, associando a aplicação a um mapa, verificar-se-ia a evolução da qualidade do talude rochoso. Deste modo, o utilizador teria uma melhor perceção da estabilidade global do talude, permitindo definir diferentes medidas de estabilização a aplicar em cada troço.

Sendo o *Android* um sistema operativo recente, o mercado de aplicações no *Google Play Store* ainda se encontra em expansão, criando a oportunidade de desenvolvimento de *apps* inovadoras. Como tal, face à disponibilidade gratuita do programa *Android Studio* e da necessidade recorrente de processos de cálculo, é possível desenvolver ferramentas que auxiliem a prática de engenharia, traduzindo-se numa elevada procura e aplicabilidade em obra.

REFERÊNCIAS BIBLIOGRÁFICAS

Android Developers (https://developer.android.com/about/dashboards/index.html). 2016a.

Android Developers (https://developer.android.com/studio/index.html). 2016b.

Android Developers (https://developer.android.com/guide/topics/sensors/sensors_motion.html). 2016c.

Android Developers (https://developer.android.com/reference/android/hardware/SensorManager.html). 2016d.

Android Statistics (http://www.appbrain.com/stats/free-and-paid-android-applications). 2016.

Barton, N., Lien, R., Lunde, J. (1974). *Engineering classification of rock masses for the design of rock support*. Rock Mechanics.

Bieniawski, Z. T. (1973). *Engineering Classification of Jointed Rock Masses*. Trans. S. African Inst. Civil Engrs.

Bieniawski, Z. T. (1974). *Estimating the strength of rock materials*. J. S. Afr. Inst. Min. Metall, 312-20.

Bieniawski, Z. T. (1989). *Engineering rock mass classifications*. John Wiley & Sons. New York.

Chen, Z. (1995). *Recent developments in slope stability analysis* 8th Int. Cong. Rock Mech (FUJII, T.). 1041-1048.

Deere, D. U., Deere, D. W. (1988). *The rock quality designation (RQD) index in practice*. American Society for Testing and Materials. Philadelphia.

GeoCoder (http://www.java2s.com/Code/Android/Core-Class/UsingGeocoder.htm). 2016.

GeoToolbox (https://play.google.com/store/apps/details?id=appinventor.ai_marco_filipponi.GeoToolbox). 2016.

González de Vallejo, L. I. (2002). *Ingeniería geológica*. Prentice Hall. Madrid.

Graph View (http://www.android-graphview.org/). 2016.

Hack, H. R. (1998). *Slope Stability Probability Classification*. ITC Delf Publication, 273.

Hoek, E (2007). *Pratical Rock Engineering*. Rocscience. Toronto.

Lindsay, P., Campbell, R. N., Fergusson, D. A., Gillard, G. R., Moore, T. A. (2001). *Slope Stability Probability Classification* International Journal of Coal Geology, 127-145.

Palmstrom, A. (1982). *The volumetric joint count - a useful and simple measure of the degree of rock mass jointing*. IV Congress International Association of Engineering Geology New Dehli.

Queiróz, R. (2016). *Android: Desenvolvimento de Aplicações com Android Studio*. MyTI.

RMR & GRC (https://play.google.com/store/apps/details?id=com.turgutsaricam.basarir). 2016.

RMR Calc Free (https://play.google.com/store/apps/details?id=es.terrasolum.rockmassratingcalc). 2016.

Robertson, A. M (1988). *Estimating weak rock strength*. Society of Mining Engineering Annual Meeting (SASTRY, K. V. S.). Phoenix. 1-5.

Rock Mass Classification (https://play.google.com/store/apps/details?id=com.rakeshsarangi.rockmassclassification). 2016.

Romana, M. R. (1993). *A Geomechanical Classification for Slopes: Slope Mass Rating*. Pergamon Press. Oxford.

Romana, M., Tomás, R., Serón, J. B. (2015). *Slope Mass Rating (SMR) geomechanics classification: thirty years review*. ISRM Congress 2015 Proceedings -InternationalSymposium on Rock Mechanics Quebec, Canada. 10.

Shuk, T. (1994). *Key elements and applications of the natural slope methodology (NSM) with some emphasis on slope stability aspects*. 4th South American Congress on Rock Mechanics 955-960.

Simple Slope (https://play.google.com/store/apps/details?id=es.terrasolum.slide_bishop&hl=pt-PT). 2016.

SMRTool (http://personal.ua.es/en/ariquelme/smrtool.html). 2016.

Tomás, R., Delgado, J., Serón, J. B. (2007). *Modification of slope mass rating (SMR) by continuous functions*. International Journal of Rock Mechanics and Mining Sciences, 1062-1069.

Wyllie, D. C., Mah, C. (2005). *Rock slope engineering*. Spon Spress.

ANEXOS

ANEXO I
Android Studio

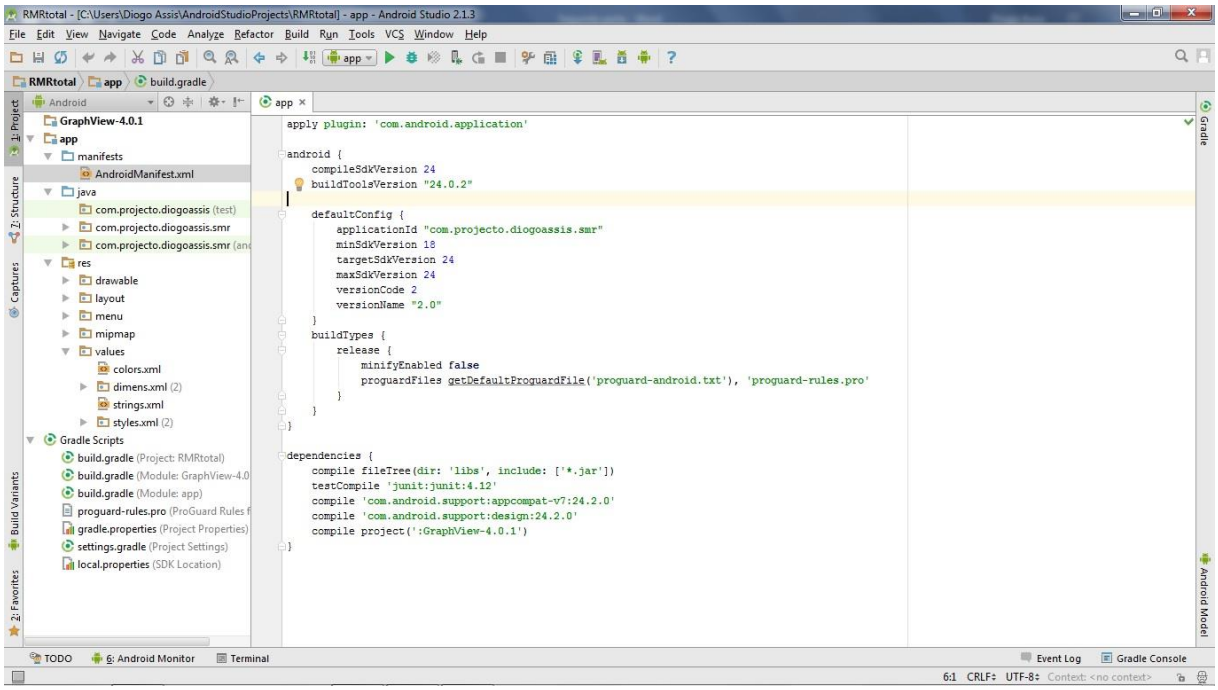


Fig. I. 1 – Ficheiro *build.gradle*

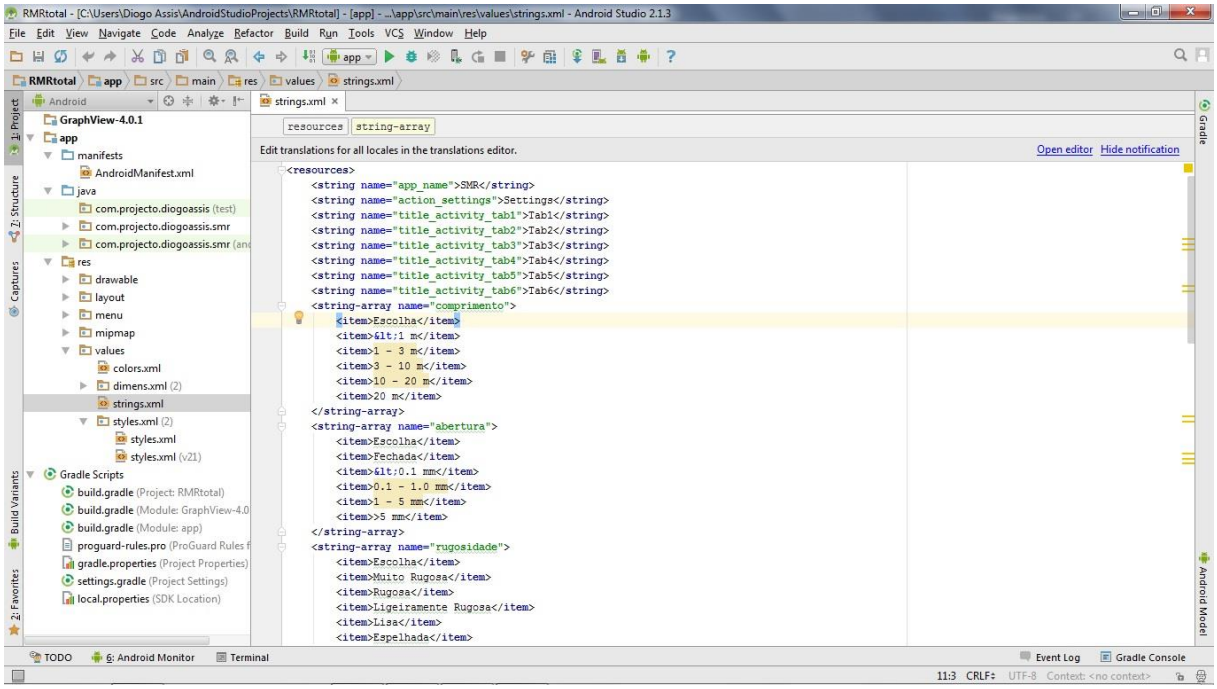


Fig. I. 2 – Ficheiro *strings*

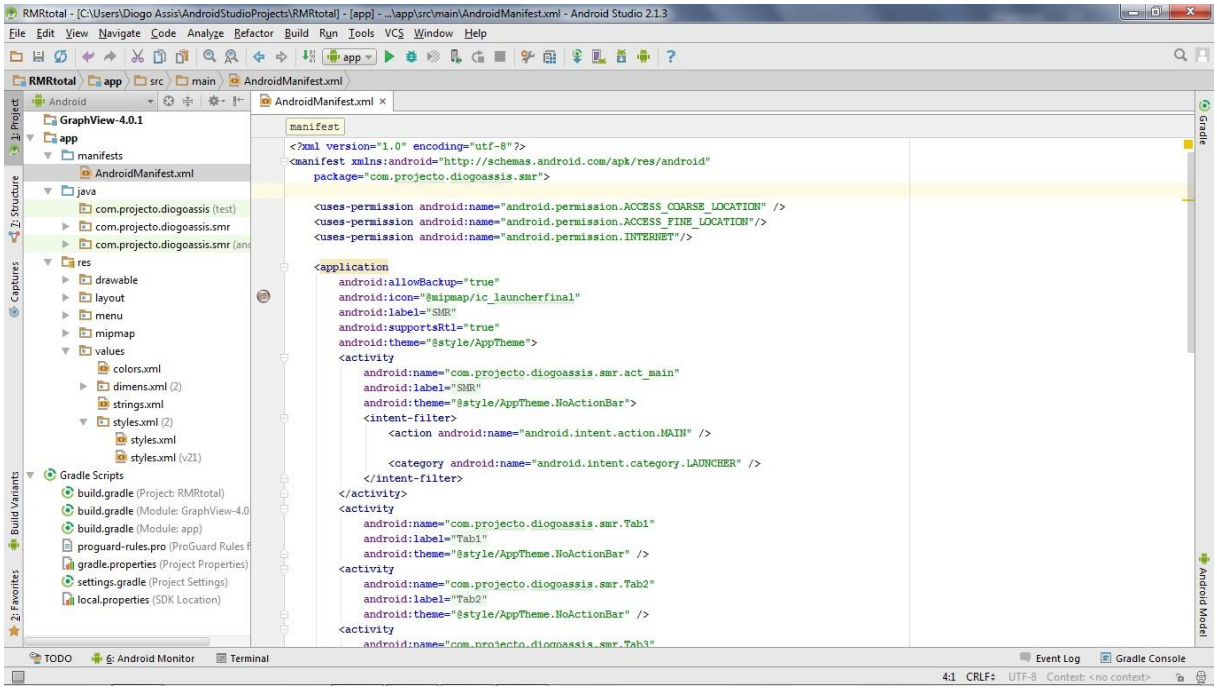


Fig. I. 3 – Ficheiro *AndroidManifest*

ANEXO II
Atividade Principal – *Act_main*

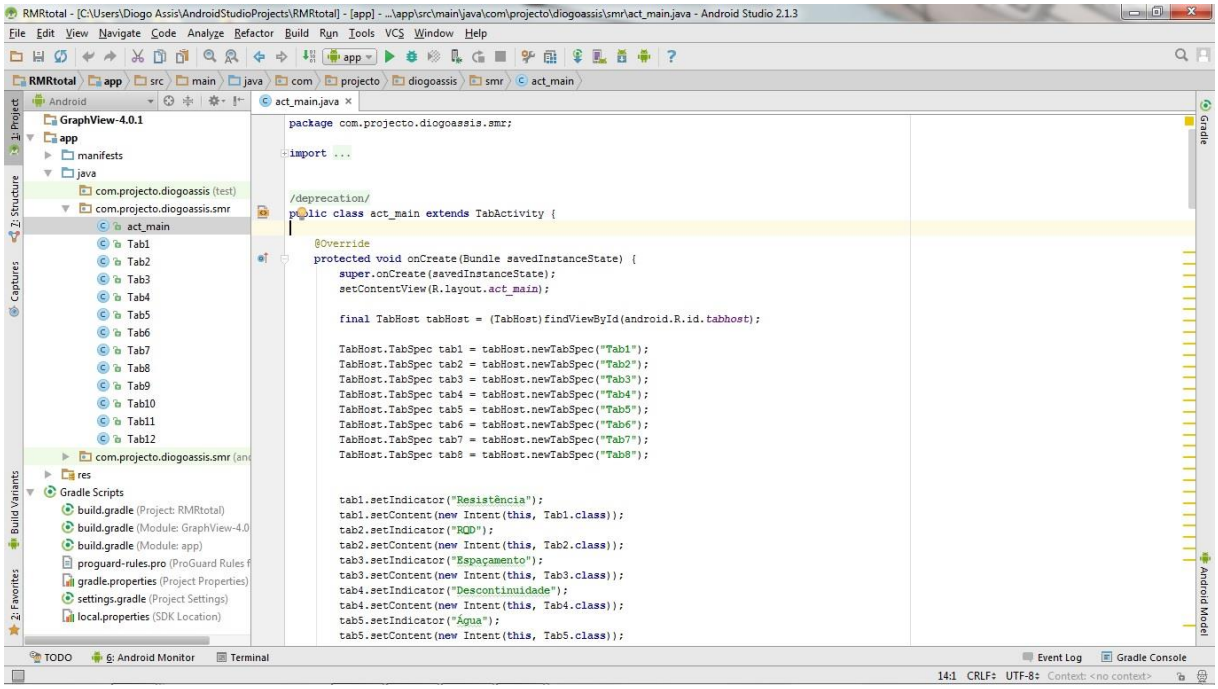


Fig. II. 1 – Código fonte da atividade *act_main*

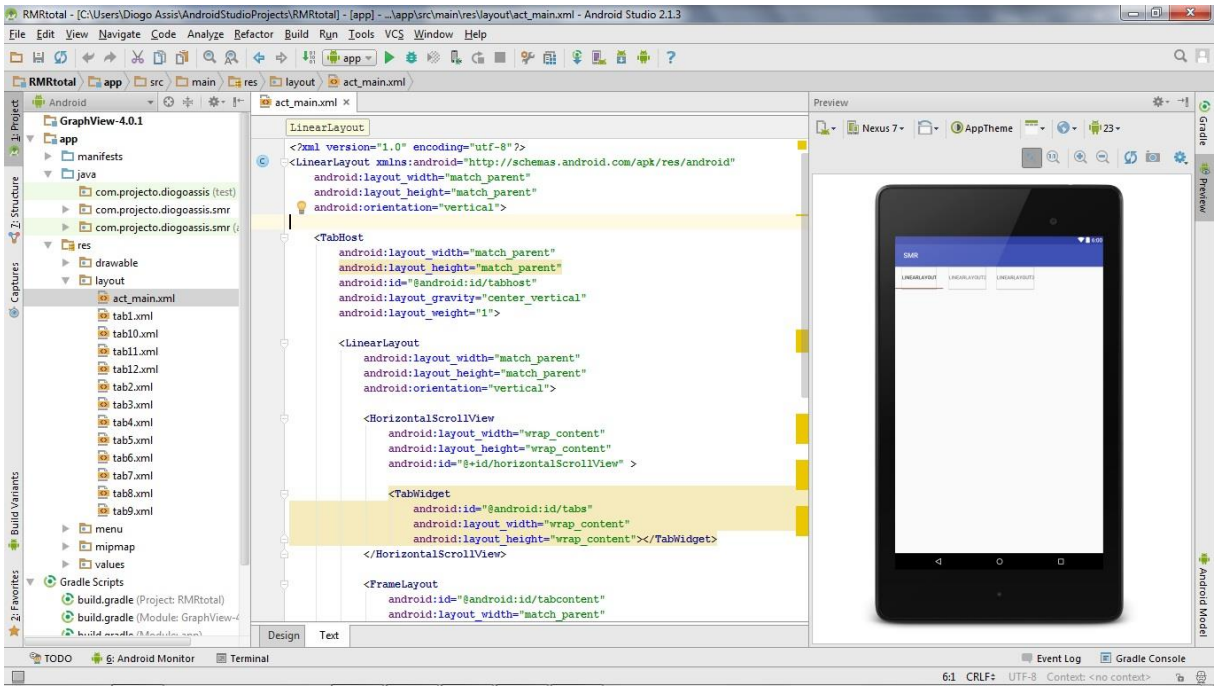


Fig. II. 2 – Código XML do layou *act_main*

```
public class act_main extends TabActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.act_main);

        final TabHost tabHost =
(TabHost)findViewById(android.R.id.tabhost);

        TabHost.TabSpec tab1 = tabHost.newTabSpec("Tab1");
        TabHost.TabSpec tab2 = tabHost.newTabSpec("Tab2");
        TabHost.TabSpec tab3 = tabHost.newTabSpec("Tab3");
        TabHost.TabSpec tab4 = tabHost.newTabSpec("Tab4");
        TabHost.TabSpec tab5 = tabHost.newTabSpec("Tab5");
        TabHost.TabSpec tab6 = tabHost.newTabSpec("Tab6");
        TabHost.TabSpec tab7 = tabHost.newTabSpec("Tab7");
        TabHost.TabSpec tab8 = tabHost.newTabSpec("Tab8");


        tab1.setIndicator("Resistência");
        tab1.setContent(new Intent(this, Tab1.class));
        tab2.setIndicator("RQD");
        tab2.setContent(new Intent(this, Tab2.class));
        tab3.setIndicator("Espaçamento");
        tab3.setContent(new Intent(this, Tab3.class));
        tab4.setIndicator("Descontinuidade");
        tab4.setContent(new Intent(this, Tab4.class));
        tab5.setIndicator("Água");
        tab5.setContent(new Intent(this, Tab5.class));
        tab6.setIndicator("RMR básico");
        tab6.setContent(new Intent(this, Tab6.class));
        tab7.setIndicator("F. Ajustamento");
        tab7.setContent(new Intent(this, Tab7.class));
        tab8.setIndicator("SMR");
        tab8.setContent(new Intent(this, Tab8.class));

        tabHost.addTab(tab1);
        tabHost.addTab(tab2);
        tabHost.addTab(tab3);
        tabHost.addTab(tab4);
        tabHost.addTab(tab5);
        tabHost.addTab(tab6);
        tabHost.addTab(tab7);
        tabHost.addTab(tab8);

        SharedPreferences pesoum = getSharedPreferences("pesoum", 0);
        SharedPreferences.Editor editor1 = pesoum.edit();
        editor1.putString("rcu", "0");
        editor1.commit();

        SharedPreferences pesodois = getSharedPreferences("pesodois", 0);
        SharedPreferences.Editor editor2 = pesodois.edit();
        editor2.putString("rqd", "0");
        editor2.commit();

        SharedPreferences pesotres = getSharedPreferences("pesotres", 0);
        SharedPreferences.Editor editor3 = pesotres.edit();
        editor3.putString("esp", "0");
```

```
        editor3.commit();

        SharedPreferences pesoquatro = getSharedPreferences("pesoquatro",
0);
        SharedPreferences.Editor editor4 = pesoquatro.edit();
        editor4.putString("des", "0");
        editor4.commit();

        SharedPreferences pesocinco = getSharedPreferences("pesocinco", 0);
        SharedPreferences.Editor editor5 = pesocinco.edit();
        editor5.putString("agu", "0");
        editor5.commit();

        SharedPreferences rmrb = getSharedPreferences("rmrb", 0);
        SharedPreferences.Editor editor6 = rmrb.edit();
        editor6.putString("rmr", "0");
        editor6.commit();

        SharedPreferences f1 = getSharedPreferences("f1", 0);
        SharedPreferences.Editor editor7 = f1.edit();
        editor7.putString("fumf", "0");
        editor7.commit();

        SharedPreferences f2 = getSharedPreferences("f2", 0);
        SharedPreferences.Editor editor8 = f2.edit();
        editor8.putString("fdoisf", "0");
        editor8.commit();

        SharedPreferences f3 = getSharedPreferences("f3", 0);
        SharedPreferences.Editor editor9 = f3.edit();
        editor9.putString("ftresf", "0");
        editor9.commit();

        SharedPreferences f4 = getSharedPreferences("f4", 0);
        SharedPreferences.Editor editor10 = f4.edit();
        editor10.putString("fquatrof", "0");
        editor10.commit();

        SharedPreferences az1 = getSharedPreferences("az1", 0);
        SharedPreferences.Editor editor11 = az1.edit();
        editor11.putString("az1f", "0");
        editor11.commit();

        SharedPreferences grau10 = getSharedPreferences("grau10", 0);
        SharedPreferences.Editor editor12 = grau10.edit();
        editor12.putString("grau1f", "0");
        editor12.commit();

        SharedPreferences queda1 = getSharedPreferences("queda1", 0);
        SharedPreferences.Editor editor13 = queda1.edit();
        editor13.putString("queda1f", "0");
        editor13.commit();

        SharedPreferences pendor1 = getSharedPreferences("pendor1", 0);
        SharedPreferences.Editor editor14 = pendor1.edit();
        editor14.putString("pendor1f", "0");
        editor14.commit();

        SharedPreferences az2 = getSharedPreferences("az2", 0);
```

```
SharedPreferences.Editor editor15 = az2.edit();
editor15.putString("az2f", "0");
editor15.commit();

SharedPreferences grau20 = getSharedPreferences("grau20", 0);
SharedPreferences.Editor editor16 = grau20.edit();
editor16.putString("grau2f", "0");
editor16.commit();

SharedPreferences queda2 = getSharedPreferences("queda2", 0);
SharedPreferences.Editor editor17 = queda2.edit();
editor17.putString("queda2f", "0");
editor17.commit();

SharedPreferences pendor2 = getSharedPreferences("pendor2", 0);
SharedPreferences.Editor editor18 = pendor2.edit();
editor18.putString("pendor2f", "0");
editor18.commit();

SharedPreferences resist = getSharedPreferences("resist", 0);
SharedPreferences.Editor editor19 = resist.edit();
editor19.putString("resist1", "");
editor19.commit();

SharedPreferences valor1 = getSharedPreferences("valor1", 0);
SharedPreferences.Editor editor20 = valor1.edit();
editor20.putString("valor11", "0");
editor20.commit();

SharedPreferences valor2 = getSharedPreferences("valor2", 0);
SharedPreferences.Editor editor21 = valor2.edit();
editor21.putString("valor21", "0");
editor21.commit();

SharedPreferences valor3 = getSharedPreferences("valor3", 0);
SharedPreferences.Editor editor22 = valor3.edit();
editor22.putString("valor31", "0");
editor22.commit();

SharedPreferences valor5 = getSharedPreferences("valor5", 0);
SharedPreferences.Editor editor24 = valor5.edit();
editor24.putString("valor51", "");
editor24.commit();

SharedPreferences valor6 = getSharedPreferences("valor6", 0);
SharedPreferences.Editor editor25 = valor6.edit();
editor25.putString("valor61", "");
editor25.commit();

SharedPreferences valor7 = getSharedPreferences("valor7", 0);
SharedPreferences.Editor editor26 = valor7.edit();
editor26.putString("valor71", "");
editor26.commit();

SharedPreferences valor8 = getSharedPreferences("valor8", 0);
SharedPreferences.Editor editor27 = valor8.edit();
editor27.putString("valor81", "");
editor27.commit();
```

```
        SharedPreferences valor9 = getSharedPreferences("valor9", 0);
        SharedPreferences.Editor editor28 = valor9.edit();
        editor28.putString("valor91", "");
        editor28.commit();

        SharedPreferences valor10 = getSharedPreferences("valor10", 0);
        SharedPreferences.Editor editor29 = valor10.edit();
        editor29.putString("valor101", "");
        editor29.commit();



    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_act_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }


    public void updateMain() {
    }
}
```

ANEXO III
Resistência da Rocha Intacta – *Tab1*

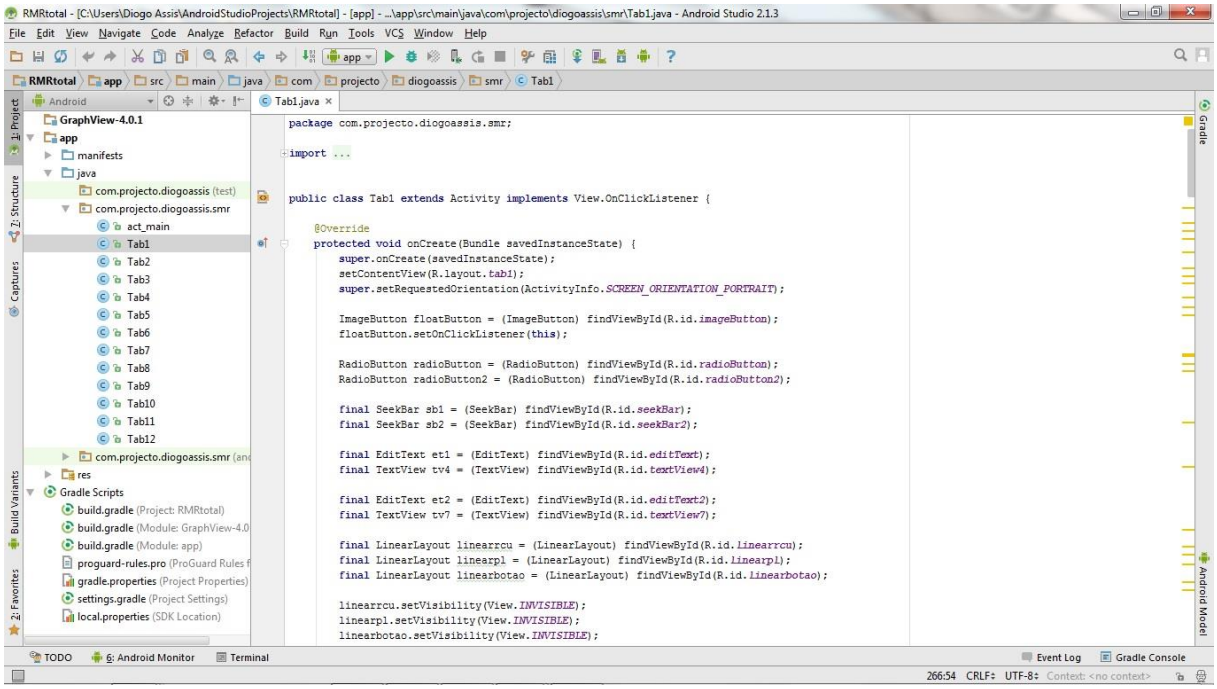


Fig. III. 1 – Código fonte da atividade *Tab1*

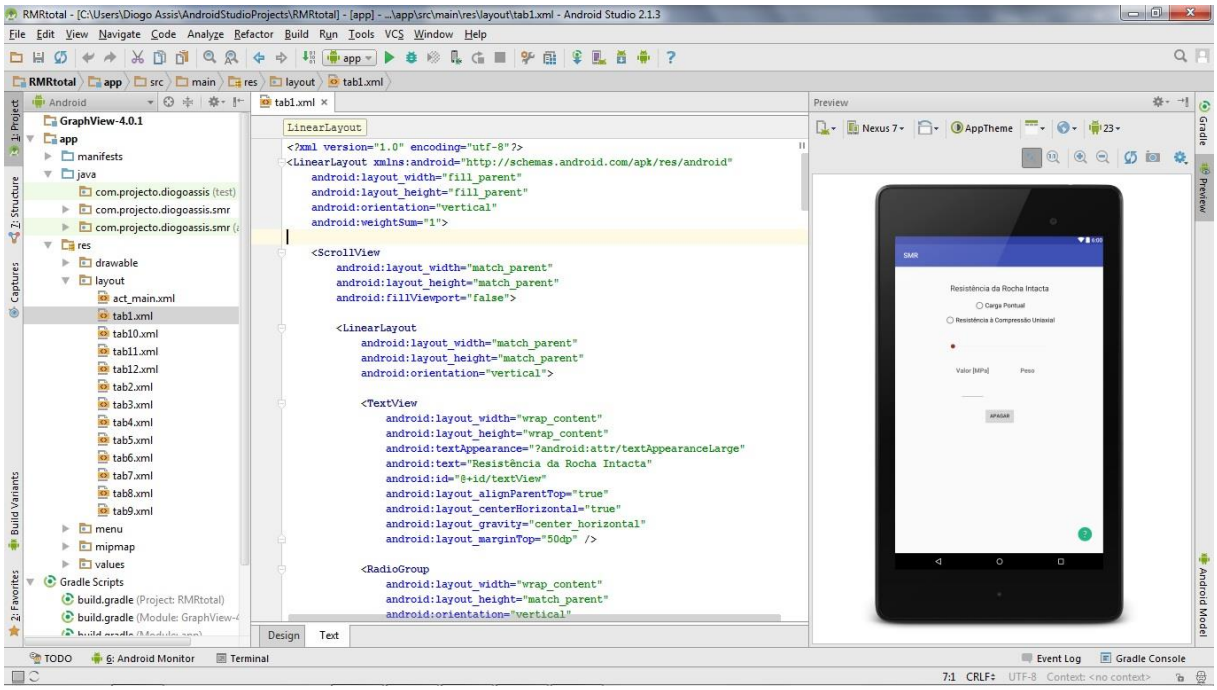


Fig. III. 2 – Código XML do *layout Tab1*

```
public class Tab1 extends Activity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab1);

super.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

        ImageButton floatButton = (ImageButton)
findViewById(R.id.imageButton);
        floatButton.setOnClickListener(this);

        RadioButton radioButton = (RadioButton)
findViewById(R.id.radioButton);
        RadioButton radioButton2 = (RadioButton)
findViewById(R.id.radioButton2);

        final SeekBar sb1 = (SeekBar) findViewById(R.id.seekBar);
        final SeekBar sb2 = (SeekBar) findViewById(R.id.seekBar2);

        final EditText et1 = (EditText) findViewById(R.id.editText);
        final TextView tv4 = (TextView) findViewById(R.id.textView4);

        final EditText et2 = (EditText) findViewById(R.id.editText2);
        final TextView tv7 = (TextView) findViewById(R.id.textView7);

        final LinearLayout linearrcu = (LinearLayout)
findViewById(R.id.linearrcu);
        final LinearLayout linearpl = (LinearLayout)
findViewById(R.id.linearpl);
        final LinearLayout linearbotao = (LinearLayout)
findViewById(R.id.linearbotao);

        linearrcu.setVisibility(View.INVISIBLE);
        linearpl.setVisibility(View.INVISIBLE);
        linearbotao.setVisibility(View.INVISIBLE);

        radioButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                linearpl.setVisibility(View.VISIBLE);
                linearrcu.setVisibility(View.INVISIBLE);
                linearbotao.setVisibility(View.VISIBLE);
                Double d = new Double(0);
                int i = d.intValue();
                sb2.setProgress(i);
                et2.setText("");
                tv7.setText("");
                SharedPreferences pesoum = getSharedPreferences("pesoum",
0);
                SharedPreferences.Editor editor1 = pesoum.edit();
                editor1.putString("rcu", String.valueOf(0));
                editor1.commit();
                SharedPreferences resist = getSharedPreferences("resist",
0);
                SharedPreferences.Editor editor19 = resist.edit();
                editor19.putString("resist1", "Carga Pontual");
                editor19.commit();
```

```
                SharedPreferences valor1 = getSharedPreferences("valor1",
0);
                SharedPreferences.Editor editor20 = valor1.edit();
                editor20.putString("valor11", String.valueOf(0));
                editor20.commit();
            }
        });

        radioButton2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                linearpl.setVisibility(View.INVISIBLE);
                linearrcu.setVisibility(View.VISIBLE);
                linearbotao.setVisibility(View.VISIBLE);
                Double d = new Double(0);
                int i = d.intValue();
                sb1.setProgress(i);
                et1.setText("");
                tv4.setText("");
                SharedPreferences pesoum = getSharedPreferences("pesoum",
0);
                SharedPreferences.Editor editor1 = pesoum.edit();
                editor1.putString("rcu", String.valueOf(0));
                editor1.commit();
                SharedPreferences resist = getSharedPreferences("resist",
0);
                SharedPreferences.Editor editor19 = resist.edit();
                editor19.putString("resist1", "σc");
                editor19.commit();
                SharedPreferences valor1 = getSharedPreferences("valor1",
0);
                SharedPreferences.Editor editor20 = valor1.edit();
                editor20.putString("valor11", String.valueOf(0));
                editor20.commit();
            }
        });

        sb1.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                double peso = -0.000145 * progress * progress + 0.095521 *
progress + 0.5;
                tv4.setText(String.valueOf(String.format("%2.0f", peso)));
                et1.setText(String.valueOf(progress));
                SharedPreferences pesoum = getSharedPreferences("pesoum",
0);
                SharedPreferences.Editor editor1 = pesoum.edit();
                editor1.putString("rcu", String.valueOf(peso));
                editor1.commit();
                SharedPreferences valor1 = getSharedPreferences("valor1",
0);
                SharedPreferences.Editor editor20 = valor1.edit();
                editor20.putString("valor11", String.valueOf(progress));
                editor20.commit();
            }

            @Override
```

```
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        sb2.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                double cp = progress + 1 ;
                double rcu = (cp) * 24;
                double peso = -0.000145 * rcu * rcu + 0.095521 * rcu + 0.5;
                tv7.setText(String.valueOf(String.format("%6.0f", peso)));
                et2.setText(String.valueOf(String.format("%2.0f", cp)));
                SharedPreferences pesoum = getSharedPreferences("pesoum",
0);
                SharedPreferences.Editor editor1 = pesoum.edit();
                editor1.putString("rcu", String.valueOf(peso));
                editor1.commit();
                SharedPreferences valor1 = getSharedPreferences("valor1",
0);
                SharedPreferences.Editor editor20 = valor1.edit();
                editor20.putString("valor11", String.valueOf(cp));
                editor20.commit();

            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        et1.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i,
int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int
i1, int i2) {
                try {

sb1.setProgress(Integer.parseInt(charSequence.toString()));
                    et1.setSelection(et1.getText().length());
                } catch (Exception ex) {}
```

```
            }

            @Override
            public void afterTextChanged(Editable editable) {

            }
        });

        et2.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i,
int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int
i1, int i2) {
                try {
                    double a =
Double.parseDouble(et2.getText().toString());
                    double b = (a - 1);
                    int c = (int)b;
                    sb2.setProgress(c);
                    et2.setSelection(et2.getText().length());
                } catch (Exception ex) {}
            }

            @Override
            public void afterTextChanged(Editable editable) {

            }
        });

        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(this);

    }


    @Override
    public void onClick(final View v) {
        switch (v.getId()) {
            case R.id.button: {
                RadioButton radioButton = (RadioButton)
findViewById(R.id.radioButton);
                RadioButton radioButton2 = (RadioButton)
findViewById(R.id.radioButton2);

                final SeekBar sb1 = (SeekBar) findViewById(R.id.seekBar);
                final EditText et1 = (EditText)
findViewById(R.id.editText);
                final TextView tv4 = (TextView)
findViewById(R.id.textView4);

                final SeekBar sb2 = (SeekBar) findViewById(R.id.seekBar2);
                final EditText et2 = (EditText)
findViewById(R.id.editText2);
                final TextView tv7 = (TextView)
```

```
findViewById(R.id.textView7);


                if (radioButton.isChecked()) {
                    Double d = new Double(0);
                    int i = d.intValue();
                    sb2.setProgress(i);
                    et2.setText("");
                    tv7.setText("");

                    SharedPreferences pesoum =
getSharedPreferences("pesoum", 0);
                    SharedPreferences.Editor editor1 = pesoum.edit();
                    editor1.putString("rcu", String.valueOf(0));
                    editor1.commit();
                    SharedPreferences valor1 =
getSharedPreferences("valor1", 0);
                    SharedPreferences.Editor editor20 = valor1.edit();
                    editor20.putString("valor11", String.valueOf(0));
                    editor20.commit();

                }

                if (radioButton2.isChecked()) {
                    Double d = new Double(0);
                    int i = d.intValue();
                    sb1.setProgress(i);
                    et1.setText("");
                    tv4.setText("");

                    SharedPreferences pesoum =
getSharedPreferences("pesoum", 0);
                    SharedPreferences.Editor editor1 = pesoum.edit();
                    editor1.putString("rcu", String.valueOf(0));
                    editor1.commit();
                    SharedPreferences valor1 =
getSharedPreferences("valor1", 0);
                    SharedPreferences.Editor editor20 = valor1.edit();
                    editor20.putString("valor11", String.valueOf(0));
                    editor20.commit();
                }
            }


            break;

            case R.id.imageButton:{
                AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
                dlg.setTitle("Resistência da Rocha Intacta");
                dlg.setMessage("Insira o valor da Carga Pontual ou da
Resistência à Compressão Uniaxial." +
                        "\nMova a barra ou digite o valor pretentido.");
                dlg.show();
            }
            break;
        }
    }
}
```

ANEXO IV

Rock Quality Designation – Tab2

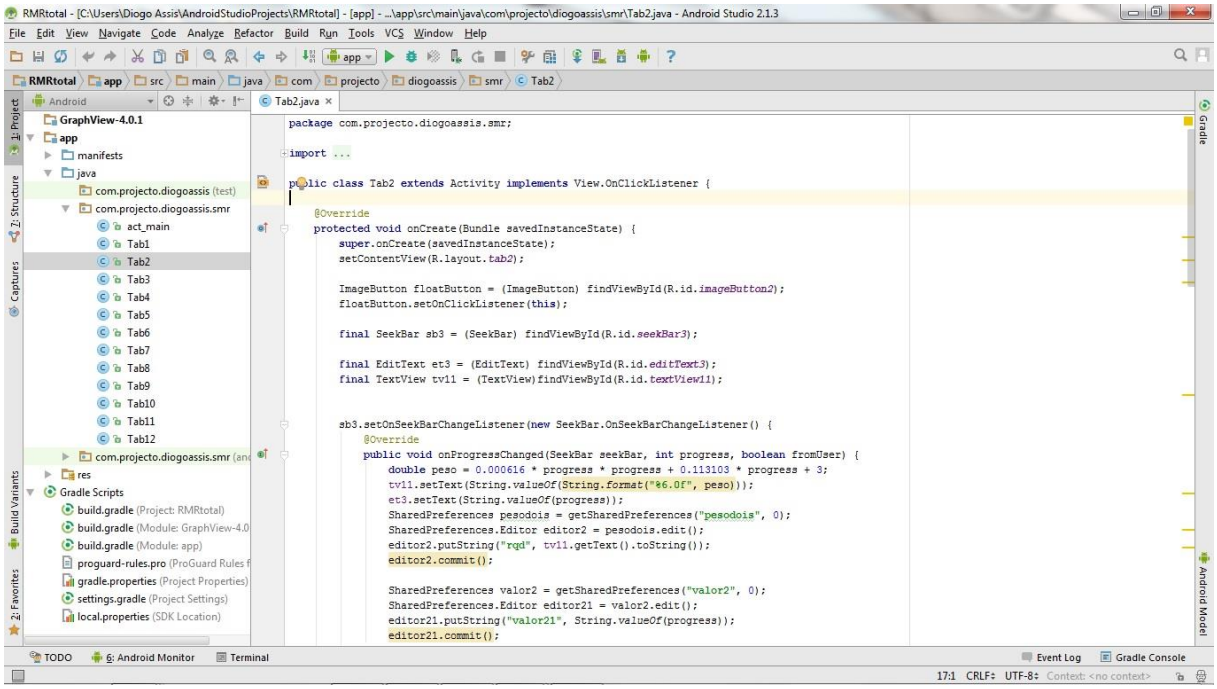


Fig. IV. 1 – Código fonte da atividade *Tab2*

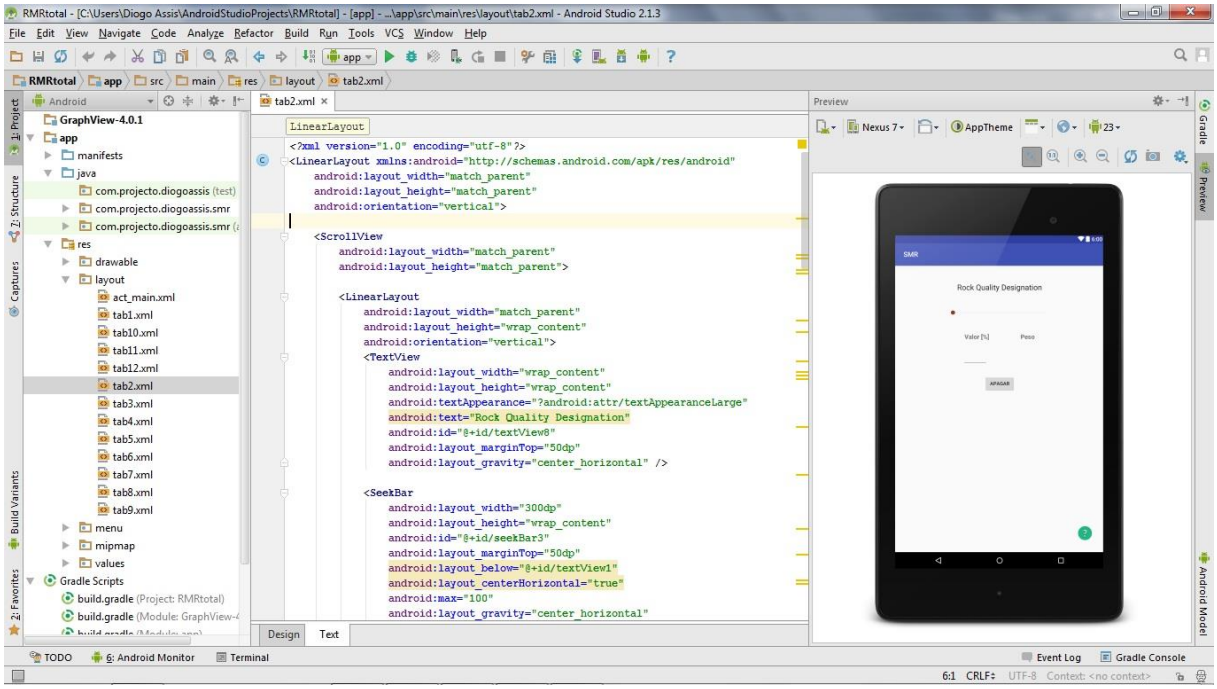


Fig. IV. 2 – Código XML do *layout Tab2*

```
public class Tab2 extends Activity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab2);

        ImageButton floatButton = (ImageButton)
findViewById(R.id.imageButton2);
        floatButton.setOnClickListener(this);

        final SeekBar sb3 = (SeekBar) findViewById(R.id.seekBar3);

        final EditText et3 = (EditText) findViewById(R.id.editText3);
        final TextView tv11 = (TextView)findViewById(R.id.textView11);


        sb3.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                double peso = 0.000616 * progress * progress + 0.113103 *
progress + 3;
                tv11.setText(String.valueOf(String.format("%6.0f", peso)));
                et3.setText(String.valueOf(progress));
                SharedPreferences pesodois =
getSharedPreferences("pesodois", 0);
                SharedPreferences.Editor editor2 = pesodois.edit();
                editor2.putString("rqd", tv11.getText().toString());
                editor2.commit();

                SharedPreferences valor2 = getSharedPreferences("valor2",
0);
                SharedPreferences.Editor editor21 = valor2.edit();
                editor21.putString("valor21", String.valueOf(progress));
                editor21.commit();
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        et3.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i,
int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int
```

```
i1, int i2) {
                try {

sb3.setProgress(Integer.parseInt(charSequence.toString()));
                    et3.setSelection(et3.getText().length());
                } catch (Exception ex) {}
            }

            @Override
            public void afterTextChanged(Editable editable) {

            }
        });

        Button button2 = (Button)findViewById(R.id.button2);
        button2.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button2: {
                final SeekBar sb3 = (SeekBar) findViewById(R.id.seekBar3);
                final EditText et3 = (EditText)
findViewById(R.id.editText3);
                final TextView tv11 =
(TextView)findViewById(R.id.textView11);

                Double d = new Double(0);
                int i = d.intValue();
                sb3.setProgress(i);
                tv11.setText("");
                et3.setText("");
                SharedPreferences pesodois =
getSharedPreferences("pesodois", 0);
                SharedPreferences.Editor editor2 = pesodois.edit();
                editor2.putString("rqd", String.valueOf(0));
                editor2.commit();
                SharedPreferences valor2 = getSharedPreferences("valor2",
0);
                SharedPreferences.Editor editor21 = valor2.edit();
                editor21.putString("valor21", String.valueOf(0));
                editor21.commit();

            }
            break;

            case R.id.imageButton2:{
                AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
                dlg.setTitle("Rock Quality Designation");
                dlg.setMessage("Insira o valor do RQD." +
                        "\nMova a barra ou digite o valor pretentido.");
                dlg.show();
            }
            break;
        }
    }
}
```

ANEXO V

Espaçamento entre Descontinuidades – *Tab3*

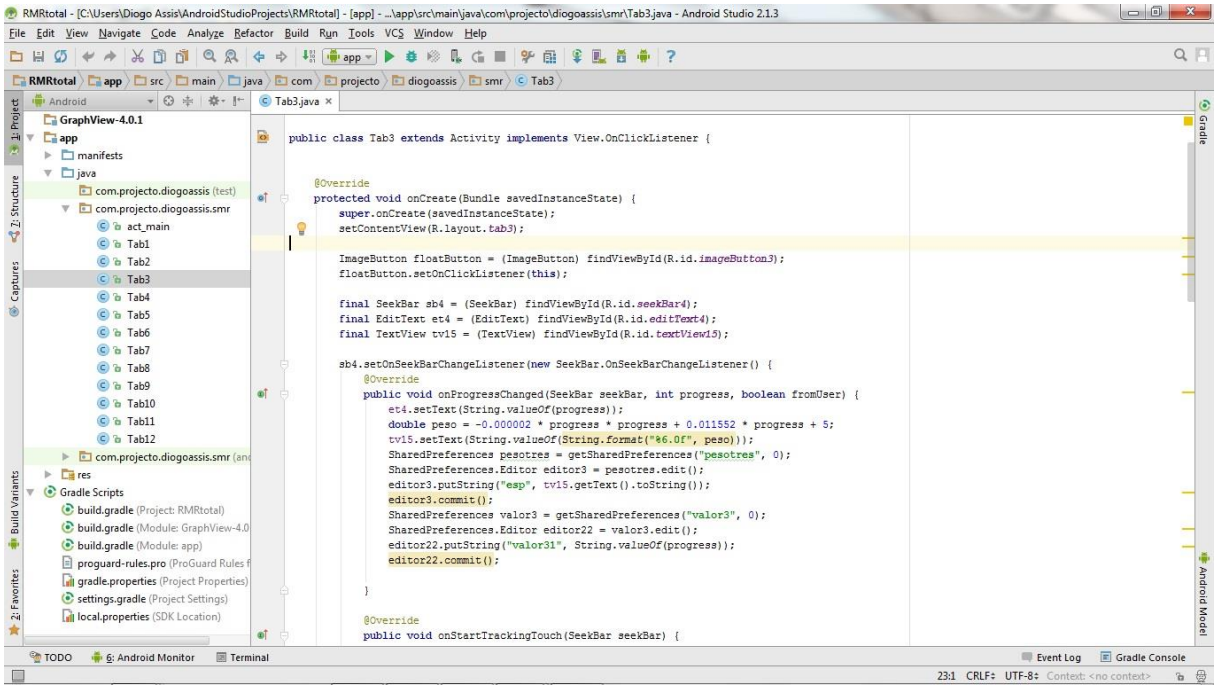


Fig. V. 1 – Código fonte da atividade *Tab3*

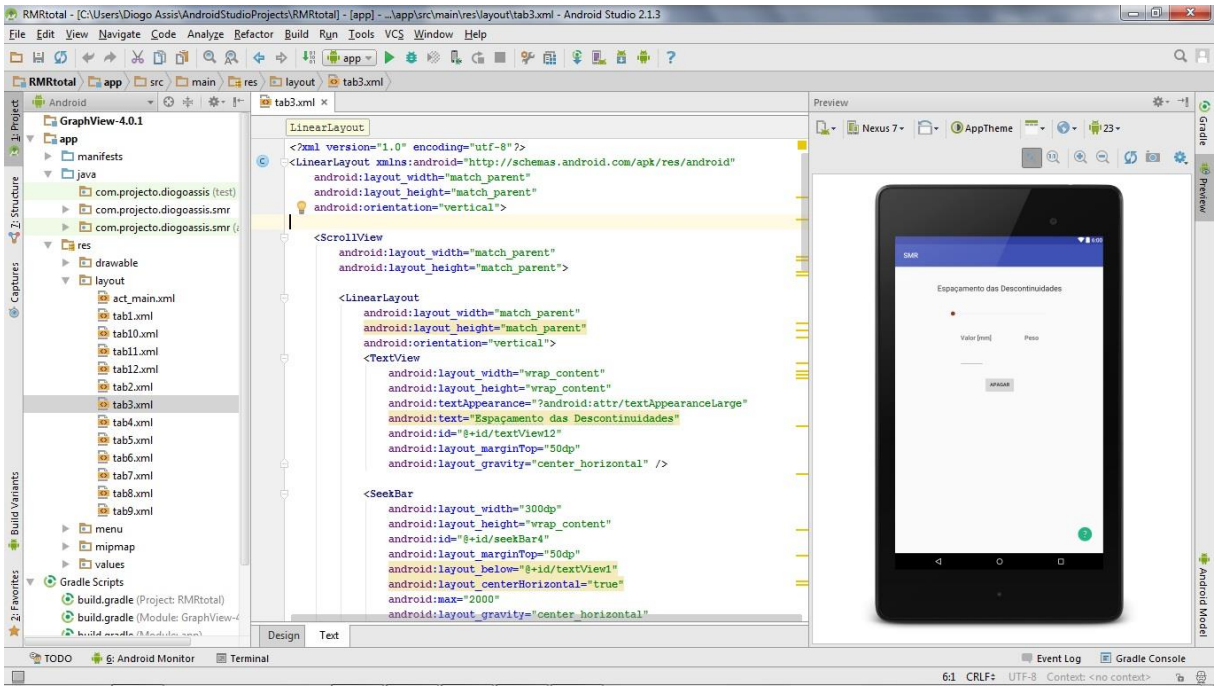


Fig. V. 2 – Código XML do *layout Tab3*

```
public class Tab3 extends Activity implements View.OnClickListener {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab3);

        ImageButton floatButton = (ImageButton)
findViewById(R.id.imageButton3);
        floatButton.setOnClickListener(this);

        final SeekBar sb4 = (SeekBar) findViewById(R.id.seekBar4);
        final EditText et4 = (EditText) findViewById(R.id.editText4);
        final TextView tv15 = (TextView) findViewById(R.id.textView15);

        sb4.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                et4.setText(String.valueOf(progress));
                double peso = -0.000002 * progress * progress + 0.011552 *
progress + 5;
                tv15.setText(String.valueOf(String.format("%6.0f", peso)));
                SharedPreferences pesotres =
getSharedPreferences("pesotres", 0);
                SharedPreferences.Editor editor3 = pesotres.edit();
                editor3.putString("esp", tv15.getText().toString());
                editor3.commit();
                SharedPreferences valor3 = getSharedPreferences("valor3",
0);
                SharedPreferences.Editor editor22 = valor3.edit();
                editor22.putString("valor31", String.valueOf(progress));
                editor22.commit();

            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        et4.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i,
int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int
i1, int i2) {
```

```
                try {

sb4.setProgress(Integer.parseInt(charSequence.toString()));
                    et4.setSelection(et4.getText().length());
                } catch (Exception ex) {}
            }

            @Override
            public void afterTextChanged(Editable editable) {

            }
        });

        Button button3 = (Button)findViewById(R.id.button3);
        button3.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button3: {
                final SeekBar sb4 = (SeekBar) findViewById(R.id.seekBar4);
                final EditText et4 = (EditText)
findViewById(R.id.editText4);
                final TextView tv15 = (TextView)
findViewById(R.id.textView15);

                Double d = new Double(0);
                int i = d.intValue();
                sb4.setProgress(i);
                tv15.setText("");
                et4.setText("");
                SharedPreferences pesotres =
getSharedPreferences("pesotres", 0);
                SharedPreferences.Editor editor3 = pesotres.edit();
                editor3.putString("esp", String.valueOf(0));
                editor3.commit();
                SharedPreferences valor3 = getSharedPreferences("valor3",
0);
                SharedPreferences.Editor editor22 = valor3.edit();
                editor22.putString("valor31", String.valueOf(0));
                editor22.commit();

            }


            break;
            case R.id.imageButton3:{
                AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
                dlg.setTitle("Espaçamento das Descontinuidades");
                dlg.setMessage("Insira o valor do espaçamento das
descontinuidades." +
                        "\nMova a barra ou digite o valor pretendido.");
                dlg.show();
            }
            break;
        }
```

ANEXO VI

Características das Descontinuidades – *Tab4*

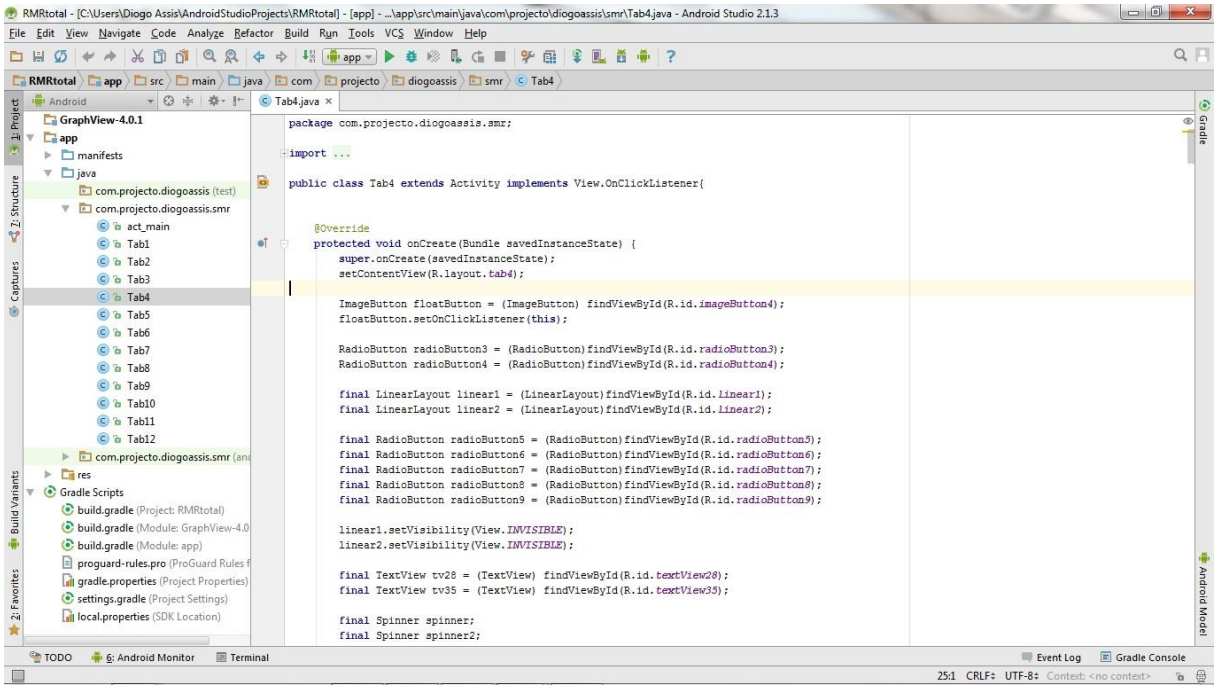


Fig. VI. 1– Código fonte da atividade *Tab4*

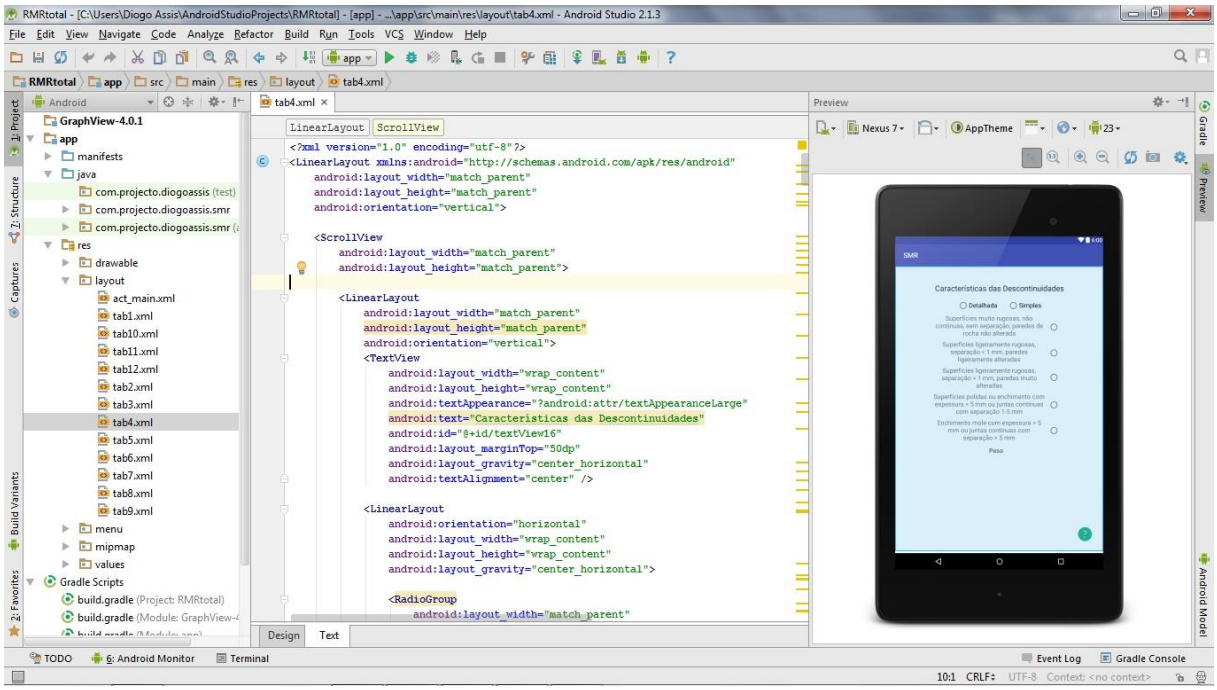


Fig. VI. 2 – Código XML do *layout Tab4*

```
public class Tab4 extends Activity implements View.OnClickListener{


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab4);

        ImageButton floatButton = (ImageButton)
findViewById(R.id.imageButton4);
        floatButton.setOnClickListener(this);

        RadioButton radioButton3 =
(RadioButton)findViewById(R.id.radioButton3);
        RadioButton radioButton4 =
(RadioButton)findViewById(R.id.radioButton4);

        final LinearLayout linear1 =
(LinearLayout)findViewById(R.id.linear1);
        final LinearLayout linear2 =
(LinearLayout)findViewById(R.id.linear2);

        final RadioButton radioButton5 =
(RadioButton)findViewById(R.id.radioButton5);
        final RadioButton radioButton6 =
(RadioButton)findViewById(R.id.radioButton6);
        final RadioButton radioButton7 =
(RadioButton)findViewById(R.id.radioButton7);
        final RadioButton radioButton8 =
(RadioButton)findViewById(R.id.radioButton8);
        final RadioButton radioButton9 =
(RadioButton)findViewById(R.id.radioButton9);

        linear1.setVisibility(View.INVISIBLE);
        linear2.setVisibility(View.INVISIBLE);

        final TextView tv28 = (TextView) findViewById(R.id.textView28);
        final TextView tv35 = (TextView) findViewById(R.id.textView35);

        final Spinner spinner;
        final Spinner spinner2;
        final Spinner spinner3;
        final Spinner spinner4;
        final Spinner spinner5;

        final ArrayAdapter<CharSequence> adapter;
        final ArrayAdapter<CharSequence> adapter2;
        final ArrayAdapter<CharSequence> adapter3;
        final ArrayAdapter<CharSequence> adapter4;
        final ArrayAdapter<CharSequence> adapter5;

        final TextView tv22 = (TextView) findViewById(R.id.textView22);
        final TextView tv23 = (TextView) findViewById(R.id.textView23);
        final TextView tv24 = (TextView) findViewById(R.id.textView24);
        final TextView tv25 = (TextView) findViewById(R.id.textView25);
        final TextView tv26 = (TextView) findViewById(R.id.textView26);

        spinner = (Spinner) findViewById(R.id.spinner);
        adapter = ArrayAdapter.createFromResource(this,
```

```
R.array.comprimento, android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
        spinner.setAdapter(adapter);

        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                String comprimento = spinner.getSelectedItem().toString();
                if (comprimento.equals("<1 m")){
                    double peso = 6;
                    tv22.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (comprimento.equals("1 - 3 m")) {
                    double peso = 4;
                    tv22.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (comprimento.equals("3 - 10 m")) {
                    double peso = 2;
                    tv22.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (comprimento.equals("10 - 20 m")) {
                    double peso = 1;
                    tv22.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (comprimento.equals("20 m")) {
                    double peso = 0;
                    tv22.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (comprimento.equals("Escolha")) {
                    double peso = 0;
                    tv22.setText(String.valueOf(String.format("%1.0f",
peso)));
                }
                SharedPreferences valor6 = getSharedPreferences("valor6",
0);
                SharedPreferences.Editor editor25 = valor6.edit();
                editor25.putString("valor61", String.valueOf(comprimento));
                editor25.commit();
                double peso1 =
Double.parseDouble(tv22.getText().toString());
                double peso2 =
Double.parseDouble(tv23.getText().toString());
                double peso3 =
Double.parseDouble(tv24.getText().toString());
                double peso4 =
Double.parseDouble(tv25.getText().toString());
                double peso5 =
Double.parseDouble(tv26.getText().toString());

                double pesototal = peso1 + peso2 + peso3 + peso4 + peso5;

                final TextView tv28 = (TextView)
findViewById(R.id.textView28);
                tv28.setText(String.valueOf(String.format("%2.0f",
pesototal)));
```

```
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv28.getText().toString());
                editor4.commit();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        spinner2 = (Spinner) findViewById(R.id.spinner2);
        adapter2 = ArrayAdapter.createFromResource(this, R.array.abertura,
android.R.layout.simple_spinner_item);

adapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        spinner2.setAdapter(adapter2);

        spinner2.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                String abertura = spinner2.getSelectedItem().toString();
                if (abertura.equals("Fechada")){
                    double peso = 6;
                    tv23.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (abertura.equals("<0.1 mm")) {
                    double peso = 5;
                    tv23.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (abertura.equals("0.1 - 1.0 mm")) {
                    double peso = 4;
                    tv23.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (abertura.equals("1 - 5 mm")) {
                    double peso = 1;
                    tv23.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (abertura.equals(">5 mm")) {
                    double peso = 0;
                    tv23.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (abertura.equals("Escolha")) {
                    double peso = 0;
                    tv23.setText(String.valueOf(String.format("%1.0f",
peso)));
                }
                SharedPreferences valor7 = getSharedPreferences("valor7",
0);
                SharedPreferences.Editor editor26 = valor7.edit();
                editor26.putString("valor71", String.valueOf(abertura));
                editor26.commit();
                double peso1 =
Double.parseDouble(tv22.getText().toString());
```

```
                double peso2 =
Double.parseDouble(tv23.getText().toString());
                double peso3 =
Double.parseDouble(tv24.getText().toString());
                double peso4 =
Double.parseDouble(tv25.getText().toString());
                double peso5 =
Double.parseDouble(tv26.getText().toString());

                double pesototal = peso1 + peso2 + peso3 + peso4 + peso5;

                final TextView tv28 = (TextView)
findViewById(R.id.textView28);
                tv28.setText(String.valueOf(String.format("%2.0f",
pesototal)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv28.getText().toString());
                editor4.commit();

            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });


        spinner3 = (Spinner) findViewById(R.id.spinner3);
        adapter3 = ArrayAdapter.createFromResource(this,
R.array.rugosidade, android.R.layout.simple_spinner_item);

adapter3.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        spinner3.setAdapter(adapter3);

        spinner3.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                String rugosidade = spinner3.getSelectedItem().toString();
                if (rugosidade.equals("Muito Rugosa")) {
                    double peso = 6;
                    tv24.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (rugosidade.equals("Rugosa")) {
                    double peso = 5;
                    tv24.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (rugosidade.equals("Ligeiramente Rugosa")) {
                    double peso = 3;
                    tv24.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (rugosidade.equals("Lisa")) {
                    double peso = 1;
                    tv24.setText(String.valueOf(String.format("%1.0f",
```

```
peso)));
                } else if (rugosidade.equals("Espelhada")) {
                    double peso = 0;
                    tv24.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (rugosidade.equals("Escolha")) {
                    double peso = 0;
                    tv24.setText(String.valueOf(String.format("%1.0f",
peso)));
                }
                SharedPreferences valor8 = getSharedPreferences("valor8",
0);
                SharedPreferences.Editor editor27 = valor8.edit();
                editor27.putString("valor81", String.valueOf(rugosidade));
                editor27.commit();
                double peso1 =
Double.parseDouble(tv22.getText().toString());
                double peso2 =
Double.parseDouble(tv23.getText().toString());
                double peso3 =
Double.parseDouble(tv24.getText().toString());
                double peso4 =
Double.parseDouble(tv25.getText().toString());
                double peso5 =
Double.parseDouble(tv26.getText().toString());

                double pesototal = peso1 + peso2 + peso3 + peso4 + peso5;

                final TextView tv28 = (TextView)
findViewById(R.id.textView28);
                tv28.setText(String.valueOf(String.format("%2.0f",
pesototal)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv28.getText().toString());
                editor4.commit();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });


        spinner4 = (Spinner) findViewById(R.id.spinner4);
        adapter4 = ArrayAdapter.createFromResource(this,
R.array.enchimento, android.R.layout.simple_spinner_item);

adapter4.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        spinner4.setAdapter(adapter4);

        spinner4.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
```

```
                String enchimento = spinner4.getSelectedItem().toString();
                if (enchimento.equals("Nenhum")) {
                    double peso = 6;
                    tv25.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (enchimento.equals("<5mm duro")) {
                    double peso = 4;
                    tv25.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (enchimento.equals(">5mm duro")) {
                    double peso = 2;
                    tv25.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (enchimento.equals("<5mm mole")) {
                    double peso = 2;
                    tv25.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (enchimento.equals(">5mm mole")) {
                    double peso = 0;
                    tv25.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (enchimento.equals("Escolha")) {
                    double peso = 0;
                    tv25.setText(String.valueOf(String.format("%1.0f",
peso)));
                }
                SharedPreferences valor9 = getSharedPreferences("valor9",
0);
                SharedPreferences.Editor editor28 = valor9.edit();
                editor28.putString("valor91", String.valueOf(enchimento));
                editor28.commit();
                double peso1 =
Double.parseDouble(tv22.getText().toString());
                double peso2 =
Double.parseDouble(tv23.getText().toString());
                double peso3 =
Double.parseDouble(tv24.getText().toString());
                double peso4 =
Double.parseDouble(tv25.getText().toString());
                double peso5 =
Double.parseDouble(tv26.getText().toString());

                double pesototal = peso1 + peso2 + peso3 + peso4 + peso5;

                final TextView tv28 = (TextView)
findViewById(R.id.textView28);
                tv28.setText(String.valueOf(String.format("%2.0f",
pesototal)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv28.getText().toString());
                editor4.commit();

            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
```

```
            }
        });

        spinner5 = (Spinner) findViewById(R.id.spinner5);
        adapter5 = ArrayAdapter.createFromResource(this, R.array.alteracao,
android.R.layout.simple_spinner_item);

adapter5.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        spinner5.setAdapter(adapter5);

        spinner5.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                String alteracao = spinner5.getSelectedItem().toString();
                if (alteracao.equals("Não Alterada")) {
                    double peso = 6;
                    tv26.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (alteracao.equals("Ligeira")) {
                    double peso = 5;
                    tv26.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (alteracao.equals("Moderada")) {
                    double peso = 3;
                    tv26.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (alteracao.equals("Muito Alterada")) {
                    double peso = 1;
                    tv26.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (alteracao.equals("Decomposta")) {
                    double peso = 0;
                    tv26.setText(String.valueOf(String.format("%1.0f",
peso)));
                } else if (alteracao.equals("Escolha")) {
                    double peso = 0;
                    tv26.setText(String.valueOf(String.format("%1.0f",
peso)));
                }
                SharedPreferences valor10 = getSharedPreferences("valor10",
0);
                SharedPreferences.Editor editor29 = valor10.edit();
                editor29.putString("valor101", String.valueOf(alteracao));
                editor29.commit();

                double peso1 =
Double.parseDouble(tv22.getText().toString());
                double peso2 =
Double.parseDouble(tv23.getText().toString());
                double peso3 =
Double.parseDouble(tv24.getText().toString());
                double peso4 =
Double.parseDouble(tv25.getText().toString());
                double peso5 =
Double.parseDouble(tv26.getText().toString());
```

```
                double pesototal = peso1 + peso2 + peso3 + peso4 + peso5;

                final TextView tv28 = (TextView)
findViewById(R.id.textView28);
                tv28.setText(String.valueOf(String.format("%2.0f",
pesototal)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv28.getText().toString());
                editor4.commit();

            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        radioButton3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                linear1.setVisibility(View.VISIBLE);
                linear2.setVisibility(View.INVISIBLE);
                tv28.setText("");
                spinner.setSelection(adapter.getPosition("Escolha"));
                spinner2.setSelection(adapter2.getPosition("Escolha"));
                spinner3.setSelection(adapter3.getPosition("Escolha"));
                spinner4.setSelection(adapter4.getPosition("Escolha"));
                spinner5.setSelection(adapter5.getPosition("Escolha"));
                double peso = 0;
                tv35.setText(String.valueOf(String.format("%1.0f", peso)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", String.valueOf(0));
                editor4.commit();

            }
        });

        radioButton4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                linear1.setVisibility(View.INVISIBLE);
                linear2.setVisibility(View.VISIBLE);
                radioButton5.setChecked(false);
                radioButton6.setChecked(false);
                radioButton7.setChecked(false);
                radioButton8.setChecked(false);
                radioButton9.setChecked(false);
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", String.valueOf(0));
                editor4.commit();

                SharedPreferences valor6 = getSharedPreferences("valor6",
```

```
0);
                SharedPreferences.Editor editor25 = valor6.edit();
                editor25.putString("valor61", "");
                editor25.commit();

                SharedPreferences valor7 = getSharedPreferences("valor7",
0);
                SharedPreferences.Editor editor26 = valor7.edit();
                editor26.putString("valor71", "");
                editor26.commit();

                SharedPreferences valor8 = getSharedPreferences("valor8",
0);
                SharedPreferences.Editor editor27 = valor8.edit();
                editor27.putString("valor81", "");
                editor27.commit();

                SharedPreferences valor9 = getSharedPreferences("valor9",
0);
                SharedPreferences.Editor editor28 = valor9.edit();
                editor28.putString("valor91", "");
                editor28.commit();

                SharedPreferences valor10 = getSharedPreferences("valor10",
0);
                SharedPreferences.Editor editor29 = valor10.edit();
                editor29.putString("valor101", "");
                editor29.commit();
                double peso = 0;
                tv35.setText(String.valueOf(String.format("%1.0f", peso)));

            }
        });

        radioButton5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                double peso = 30;
                tv35.setText(String.valueOf(String.format("%2.0f", peso)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv35.getText().toString());
                editor4.commit();

            }
        });

        radioButton6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                double peso = 25;
                tv35.setText(String.valueOf(String.format("%2.0f", peso)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv35.getText().toString());
                editor4.commit();
```

```
            }
        });


        radioButton7.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                double peso = 20;
                tv35.setText(String.valueOf(String.format("%2.0f", peso)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv35.getText().toString());
                editor4.commit();

            }
        });

        radioButton8.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                double peso = 10;
                tv35.setText(String.valueOf(String.format("%2.0f", peso)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv35.getText().toString());
                editor4.commit();

            }
        });

        radioButton9.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                double peso = 0;
                tv35.setText(String.valueOf(String.format("%1.0f", peso)));
                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", tv35.getText().toString());
                editor4.commit();

            }
        });


        Button button4 = (Button)findViewById(R.id.button4);
        button4.setOnClickListener(this);


    }


    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button4: {
                final TextView tv35 = (TextView)
```

```
findViewById(R.id.textView35);
                tv35.setText("");

                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                SharedPreferences.Editor editor4 = pesoquatro.edit();
                editor4.putString("des", String.valueOf(0));
                editor4.commit();

                final Spinner spinner;
                final Spinner spinner2;
                final Spinner spinner3;
                final Spinner spinner4;
                final Spinner spinner5;

                final ArrayAdapter<CharSequence> adapter;
                final ArrayAdapter<CharSequence> adapter2;
                final ArrayAdapter<CharSequence> adapter3;
                final ArrayAdapter<CharSequence> adapter4;
                final ArrayAdapter<CharSequence> adapter5;

                spinner = (Spinner) findViewById(R.id.spinner);
                adapter = ArrayAdapter.createFromResource(this,
R.array.comprimento, android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
                spinner.setAdapter(adapter);

                spinner2 = (Spinner) findViewById(R.id.spinner2);
                adapter2 = ArrayAdapter.createFromResource(this,
R.array.abertura, android.R.layout.simple_spinner_item);

adapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
                spinner2.setAdapter(adapter2);

                spinner3 = (Spinner) findViewById(R.id.spinner3);
                adapter3 = ArrayAdapter.createFromResource(this,
R.array.rugosidade, android.R.layout.simple_spinner_item);

adapter3.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
                spinner3.setAdapter(adapter3);

                spinner4 = (Spinner) findViewById(R.id.spinner4);
                adapter4 = ArrayAdapter.createFromResource(this,
R.array.enchimento, android.R.layout.simple_spinner_item);

adapter4.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
                spinner4.setAdapter(adapter4);

                spinner5 = (Spinner) findViewById(R.id.spinner5);
                adapter5 = ArrayAdapter.createFromResource(this,
R.array.alteracao, android.R.layout.simple_spinner_item);

adapter5.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
```

```
                spinner5.setAdapter(adapter5);

                spinner.setSelection(adapter.getPosition("Escolha"));
                spinner2.setSelection(adapter2.getPosition("Escolha"));
                spinner3.setSelection(adapter3.getPosition("Escolha"));
                spinner4.setSelection(adapter4.getPosition("Escolha"));
                spinner5.setSelection(adapter5.getPosition("Escolha"));

            }

            break;

            case R.id.imageButton4:{
                AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
                dlg.setTitle("Características das Descontinuidades");
                dlg.setMessage("Caracterize as descontinuidades de forma
simples ou detalhada." +
                        "\nEscolha uma caraterização padrão ou avalie cada
aspeto.");
                dlg.show();
            }
            break;
        }
    }

    }
```

ANEXO VII
Presença de Água – *Tab5*

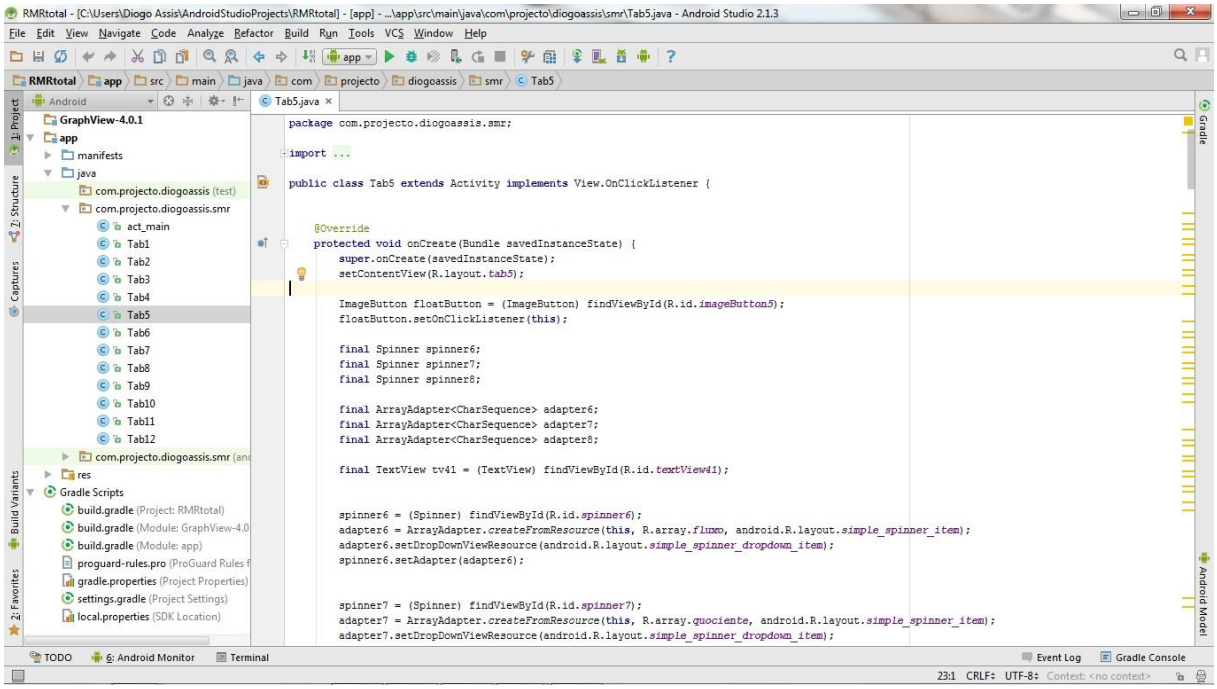


Fig. VII. 1 – Código fonte da atividade *Tab5*

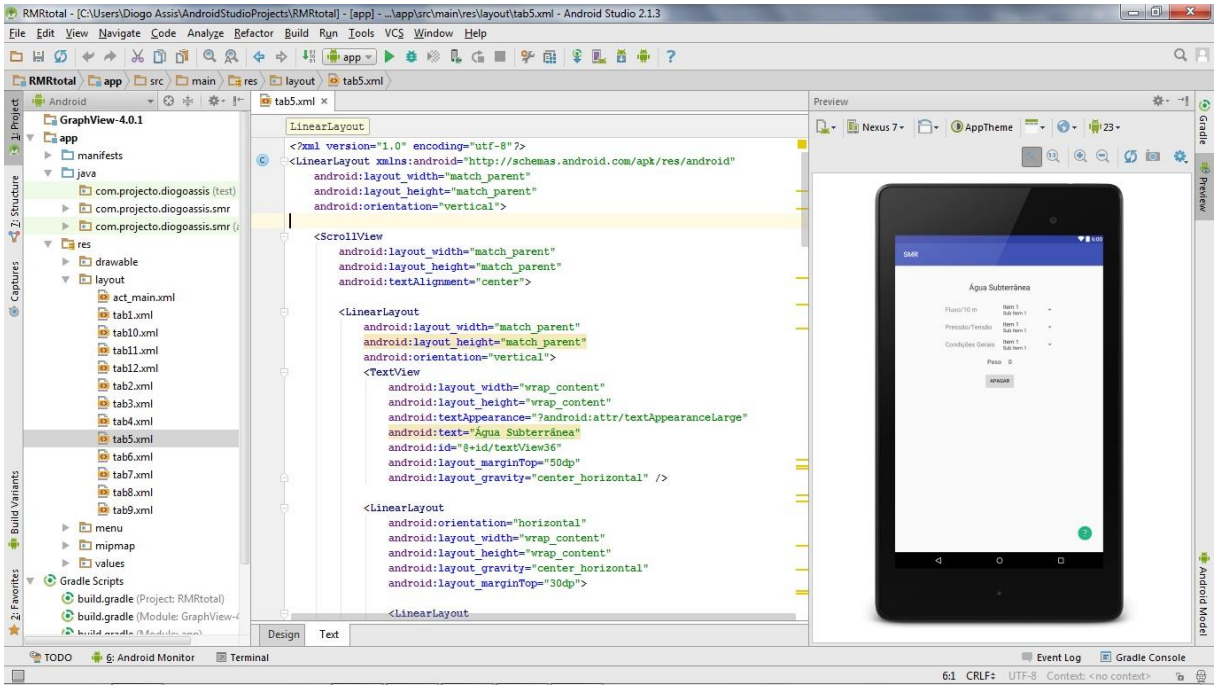


Fig. VII. 2 – Código XML do *layout Tab5*

```
public class Tab5 extends Activity implements View.OnClickListener {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab5);

        ImageButton floatButton = (ImageButton)
findViewById(R.id.imageButton5);
        floatButton.setOnClickListener(this);

        final Spinner spinner6;
        final Spinner spinner7;
        final Spinner spinner8;

        final ArrayAdapter<CharSequence> adapter6;
        final ArrayAdapter<CharSequence> adapter7;
        final ArrayAdapter<CharSequence> adapter8;

        final TextView tv41 = (TextView) findViewById(R.id.textView41);


        spinner6 = (Spinner) findViewById(R.id.spinner6);
        adapter6 = ArrayAdapter.createFromResource(this, R.array.fluxo,
android.R.layout.simple_spinner_item);

adapter6.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        spinner6.setAdapter(adapter6);


        spinner7 = (Spinner) findViewById(R.id.spinner7);
        adapter7 = ArrayAdapter.createFromResource(this, R.array.quociente,
android.R.layout.simple_spinner_item);

adapter7.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        spinner7.setAdapter(adapter7);

        spinner8 = (Spinner) findViewById(R.id.spinner8);
        adapter8 = ArrayAdapter.createFromResource(this, R.array.condicao,
android.R.layout.simple_spinner_item);

adapter8.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        spinner8.setAdapter(adapter8);


        spinner6.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                String fluxo = spinner6.getSelectedItem().toString();
                if (fluxo.equals("Nulo")) {
                    double peso = 15;
                    tv41.setText(String.valueOf(String.format("%2.0f",
peso)));
```

```
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51", String.valueOf(fluxo));
                    editor24.commit();
                } if (fluxo.equals("<10 l/min")) {
                    double peso = 10;
                    tv41.setText(String.valueOf(String.format("%2.0f",
peso)));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51", String.valueOf(fluxo));
                    editor24.commit();
                } if (fluxo.equals("10 - 25 l/min")) {
                    double peso = 7;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51", String.valueOf(fluxo));
                    editor24.commit();
                } if (fluxo.equals("25 - 125 l/min")) {
                    double peso = 4;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51", String.valueOf(fluxo));
                    editor24.commit();
                } if (fluxo.equals(">125 l/min")) {
                    double peso = 0;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51", String.valueOf(fluxo));
                    editor24.commit();
                }
                SharedPreferences pesocinco =
getSharedPreferences("pesocinco", 0);
                SharedPreferences.Editor editor5 = pesocinco.edit();
                editor5.putString("agu", tv41.getText().toString());
                editor5.commit();
            }
```

```
            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });


        spinner7.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                String quociente = spinner7.getSelectedItem().toString();
                if (quociente.equals("0")) {
                    double peso = 15;
                    tv41.setText(String.valueOf(String.format("%2.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(quociente));
                    editor24.commit();
                } if (quociente.equals("0.0 - 0.1")) {
                    double peso = 10;
                    tv41.setText(String.valueOf(String.format("%2.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(quociente));
                    editor24.commit();
                } if (quociente.equals("0.1 - 0.2")) {
                    double peso = 7;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(quociente));
                    editor24.commit();
                } if (quociente.equals("0.2 - 0.5")) {
                    double peso = 4;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
```

```
String.valueOf(quociente));
                    editor24.commit();
                } if (quociente.equals(">0.5")) {
                    double peso = 0;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner8.setSelection(adapter8.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(quociente));
                    editor24.commit();
                }
                SharedPreferences pesocinco =
getSharedPreferences("pesocinco", 0);
                SharedPreferences.Editor editor5 = pesocinco.edit();
                editor5.putString("agu", tv41.getText().toString());
                editor5.commit();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });


        spinner8.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                String condicao = spinner8.getSelectedItem().toString();
                if (condicao.equals("Seco")) {
                    double peso = 15;
                    tv41.setText(String.valueOf(String.format("%2.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(condicao));
                    editor24.commit();
                } if (condicao.equals("Húmido")) {
                    double peso = 10;
                    tv41.setText(String.valueOf(String.format("%2.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(condicao));
                    editor24.commit();
```

```
                } if (condicao.equals("Saturado")) {
                    double peso = 7;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(condicao));
                    editor24.commit();
                } if (condicao.equals("Gotejante")) {
                    double peso = 4;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(condicao));
                    editor24.commit();
                } if (condicao.equals("Escorrência")) {
                    double peso = 0;
                    tv41.setText(String.valueOf(String.format("%1.0f",
peso)));
                    spinner6.setSelection(adapter6.getPosition("Escolha"));
                    spinner7.setSelection(adapter7.getPosition("Escolha"));
                    SharedPreferences valor5 =
getSharedPreferences("valor5", 0);
                    SharedPreferences.Editor editor24 = valor5.edit();
                    editor24.putString("valor51",
String.valueOf(condicao));
                    editor24.commit();
                }
                SharedPreferences pesocinco =
getSharedPreferences("pesocinco", 0);
                SharedPreferences.Editor editor5 = pesocinco.edit();
                editor5.putString("agu", tv41.getText().toString());
                editor5.commit();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        Button button5 = (Button) findViewById(R.id.button5);
        button5.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button5: {
```

```
                final Spinner spinner6;
                final Spinner spinner7;
                final Spinner spinner8;

                final ArrayAdapter<CharSequence> adapter6;
                final ArrayAdapter<CharSequence> adapter7;
                final ArrayAdapter<CharSequence> adapter8;

                spinner6 = (Spinner) findViewById(R.id.spinner6);
                adapter6 = ArrayAdapter.createFromResource(this,
R.array.fluxo, android.R.layout.simple_spinner_item);

adapter6.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
                spinner6.setAdapter(adapter6);

                spinner7 = (Spinner) findViewById(R.id.spinner7);
                adapter7 = ArrayAdapter.createFromResource(this,
R.array.quociente, android.R.layout.simple_spinner_item);

adapter7.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
                spinner7.setAdapter(adapter7);

                spinner8 = (Spinner) findViewById(R.id.spinner8);
                adapter8 = ArrayAdapter.createFromResource(this,
R.array.condicao, android.R.layout.simple_spinner_item);

adapter8.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
                spinner8.setAdapter(adapter8);

                spinner6.setSelection(adapter6.getPosition("Escolha"));
                spinner7.setSelection(adapter7.getPosition("Escolha"));
                spinner8.setSelection(adapter8.getPosition("Escolha"));

                final TextView tv41 = (TextView)
findViewById(R.id.textView41);
                tv41.setText(String.valueOf(0));

                SharedPreferences pesocinco =
getSharedPreferences("pesocinco", 0);
                SharedPreferences.Editor editor5 = pesocinco.edit();
                editor5.putString("agu", String.valueOf(0));
                editor5.commit();

                SharedPreferences valor5 = getSharedPreferences("valor5",
0);
                SharedPreferences.Editor editor24 = valor5.edit();
                editor24.putString("valor51", "Escolha");
                editor24.commit();

            }

            break;


            case R.id.imageButton5:{
                AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
```

```
                dlg.setTitle("Água Subterrânea");
                dlg.setMessage("Escolha uma das três formas de caracterizar
a existência de água, indicando o seu valor.");
                dlg.show();
            }
            break;

        }
    }
}
```

ANEXO VIII
RMR Básico – *Tab6*

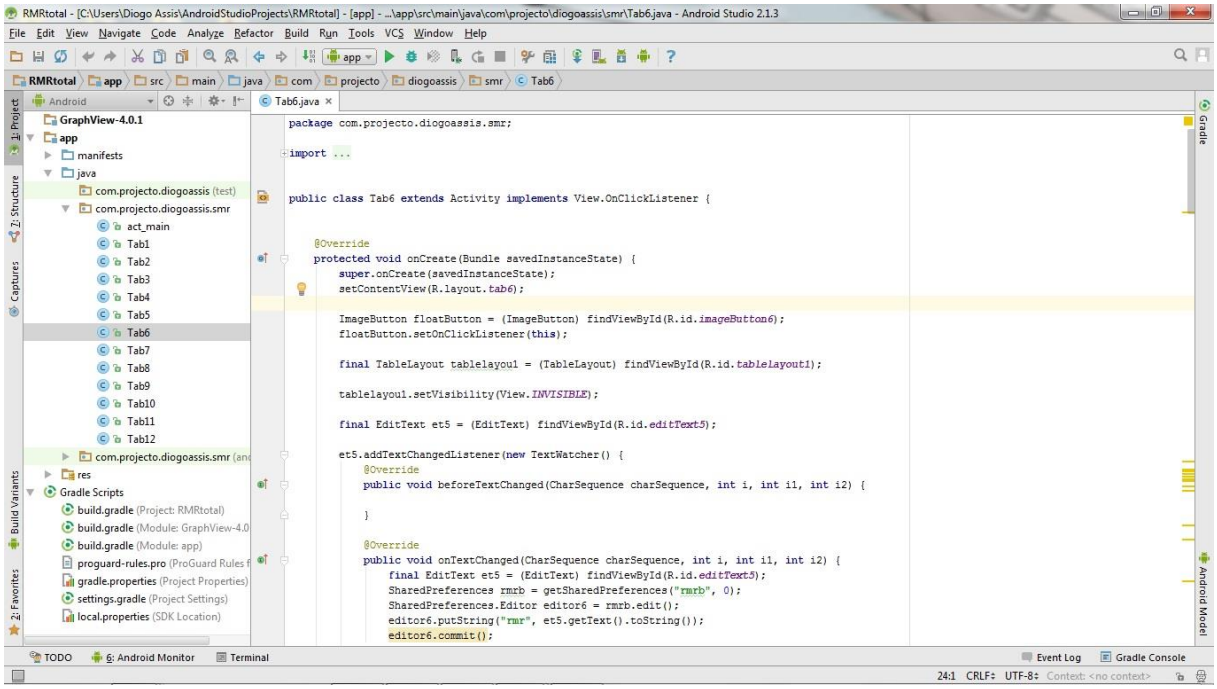


Fig. VIII. 1 – Código fonte da atividade *Tab6*

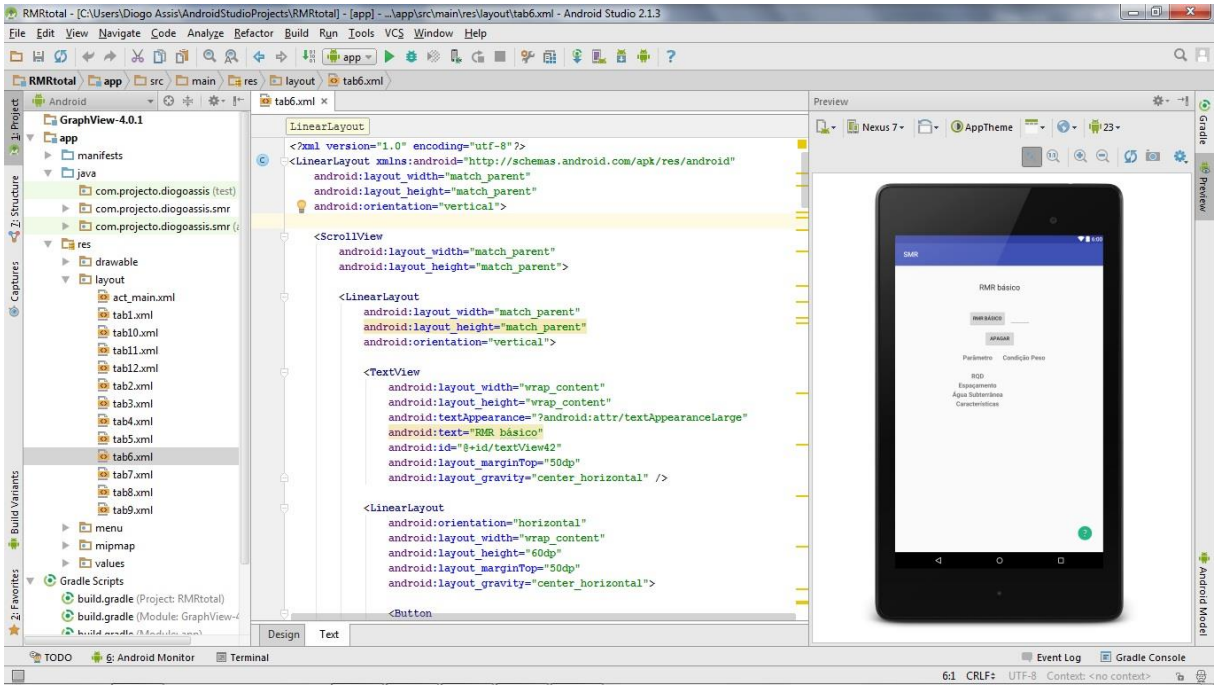


Fig. VIII. 2 – Código XML do *layout Tab6*

```
public class Tab6 extends Activity implements View.OnClickListener {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab6);

        ImageButton floatButton = (ImageButton)
findViewById(R.id.imageButton6);
        floatButton.setOnClickListener(this);

        final TableLayout tablelayou1 = (TableLayout)
findViewById(R.id.tablelayout1);

        tablelayou1.setVisibility(View.INVISIBLE);

        final EditText et5 = (EditText) findViewById(R.id.editText5);

        et5.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i,
int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int
i1, int i2) {
                final EditText et5 = (EditText)
findViewById(R.id.editText5);
                SharedPreferences rmrb = getSharedPreferences("rmrb", 0);
                SharedPreferences.Editor editor6 = rmrb.edit();
                editor6.putString("rmr", et5.getText().toString());
                editor6.commit();
            }

            @Override
            public void afterTextChanged(Editable editable) {

            }
        });


        Button button6 = (Button)findViewById(R.id.button6);
        button6.setOnClickListener(this);

        Button button7 = (Button)findViewById(R.id.button7);
        button7.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button6: {
                TableLayout tablelayou1 = (TableLayout)
findViewById(R.id.tablelayout1);
```

```
                tablelayou1.setVisibility(View.VISIBLE);


                final TextView tv46 = (TextView)
findViewById(R.id.textView46);
                final TextView tv47 = (TextView)
findViewById(R.id.textView47);
                final TextView tv48 = (TextView)
findViewById(R.id.textView48);

                final TextView tv50 = (TextView)
findViewById(R.id.textView50);
                final TextView tv51 = (TextView)
findViewById(R.id.textView51);

                final TextView tv53 = (TextView)
findViewById(R.id.textView53);
                final TextView tv54 = (TextView)
findViewById(R.id.textView54);

                final TextView tv56 = (TextView)
findViewById(R.id.textView56);
                final TextView tv57 = (TextView)
findViewById(R.id.textView57);

                final TextView tv59 = (TextView)
findViewById(R.id.textView59);
                final TextView tv60 = (TextView)
findViewById(R.id.textView60);

                final EditText et5 = (EditText)
findViewById(R.id.editText5);

                SharedPreferences pesoum = getSharedPreferences("pesoum",
0);
                String rcu = pesoum.getString("rcu", "0");

                SharedPreferences resist = getSharedPreferences("resist",
0);
                String resist1 = resist.getString("resist1", "");

                SharedPreferences valor1 = getSharedPreferences("valor1",
0);
                String valor11 = valor1.getString("valor11", "0");

                SharedPreferences pesodois =
getSharedPreferences("pesodois", 0);
                String rqd = pesodois.getString("rqd", "0");

                SharedPreferences valor2 = getSharedPreferences("valor2",
0);
                String valor21 = valor2.getString("valor21", "0");

                SharedPreferences pesotres =
getSharedPreferences("pesotres", 0);
                String esp = pesotres.getString("esp", "0");

                SharedPreferences valor3 = getSharedPreferences("valor3",
0);
```

```
                String valor31 = valor3.getString("valor31", "0");

                SharedPreferences pesoquatro =
getSharedPreferences("pesoquatro", 0);
                String des = pesoquatro.getString("des", "0");

                SharedPreferences pesocinco =
getSharedPreferences("pesocinco", 0);
                String agu = pesocinco.getString("agu", "0");

                SharedPreferences valor5 = getSharedPreferences("valor5",
0);
                String valor51 = valor5.getString("valor51", "");

                SharedPreferences valor6 = getSharedPreferences("valor6",
0);
                String valor61 = valor6.getString("valor61", "");

                SharedPreferences valor7 = getSharedPreferences("valor7",
0);
                String valor71 = valor7.getString("valor71", "");

                SharedPreferences valor8 = getSharedPreferences("valor8",
0);
                String valor81 = valor8.getString("valor81", "");

                SharedPreferences valor9 = getSharedPreferences("valor9",
0);
                String valor91 = valor9.getString("valor91", "");

                SharedPreferences valor10 = getSharedPreferences("valor10",
0);
                String valor101 = valor10.getString("valor101", "");


                double a = Double.parseDouble(rcu);
                double a1 = Double.parseDouble(valor11);
                double b = Double.parseDouble(rqd);
                double b1 = Double.parseDouble(valor21);
                double c = Double.parseDouble(esp);
                double c1 = Double.parseDouble(valor31);
                double d = Double.parseDouble(des);
                double e = Double.parseDouble(agu);

                double total = a + b + c + d + e;


                et5.setText(String.valueOf(String.format("%2.0f", total)));

                tv46.setText(resist1);
                tv47.setText(String.format("%3.0f", a1) + "MPa");
                tv48.setText(String.valueOf(String.format("%2.0f", a)));
                tv50.setText(String.format("%3.0f", b1) + "%");
                tv51.setText(String.valueOf(String.format("%2.0f", b)));
                tv53.setText(String.format("%4.0f", c1) + "mm");
                tv54.setText(String.valueOf(String.format("%2.0f", c)));
                tv56.setText(valor51);
                tv57.setText(String.valueOf(String.format("%2.0f", e)));
```

```
                tv60.setText(String.valueOf(String.format("%2.0f", d)));




                if (String.valueOf(valor61).trim().length() == 0 &
                        String.valueOf(valor71).trim().length() == 0 &
                        String.valueOf(valor81).trim().length() == 0 &
                        String.valueOf(valor91).trim().length() == 0 &
                        String.valueOf(valor101).trim().length() == 0){
                    tv59.setText("");
                }
                else {

tv59.setText(valor61+"\n"+valor71+"\n"+valor81+"\n"+valor91+"\n"+valor101);
                }

                SharedPreferences rmrb = getSharedPreferences("rmrb", 0);
                SharedPreferences.Editor editor6 = rmrb.edit();
                editor6.putString("rmr", et5.getText().toString());
                editor6.commit();


            }

            break;

            case R.id.button7: {

                TableLayout tablelayou1 = (TableLayout)
findViewById(R.id.tablelayout1);
                tablelayou1.setVisibility(View.INVISIBLE);

                final EditText et5 = (EditText)
findViewById(R.id.editText5);
                et5.setText("");
                SharedPreferences rmrb = getSharedPreferences("rmrb", 0);
                SharedPreferences.Editor editor6 = rmrb.edit();
                editor6.putString("rmr", "0");
                editor6.commit();

                break;
            }

            case R.id.imageButton6:{
                AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
                dlg.setTitle("RMR básico");
                dlg.setMessage("Pressione o botão RMR básico para o cálculo
deste índice " +
                        "com base nos parâmetros definidos anteriormente."
+
                        "\nCaso o queira alterar, digite o valor
pretendido.");
                dlg.show();
            }
            break;
        }
```

ANEXO IX
Fatores de Ajustamento – *Tab7*

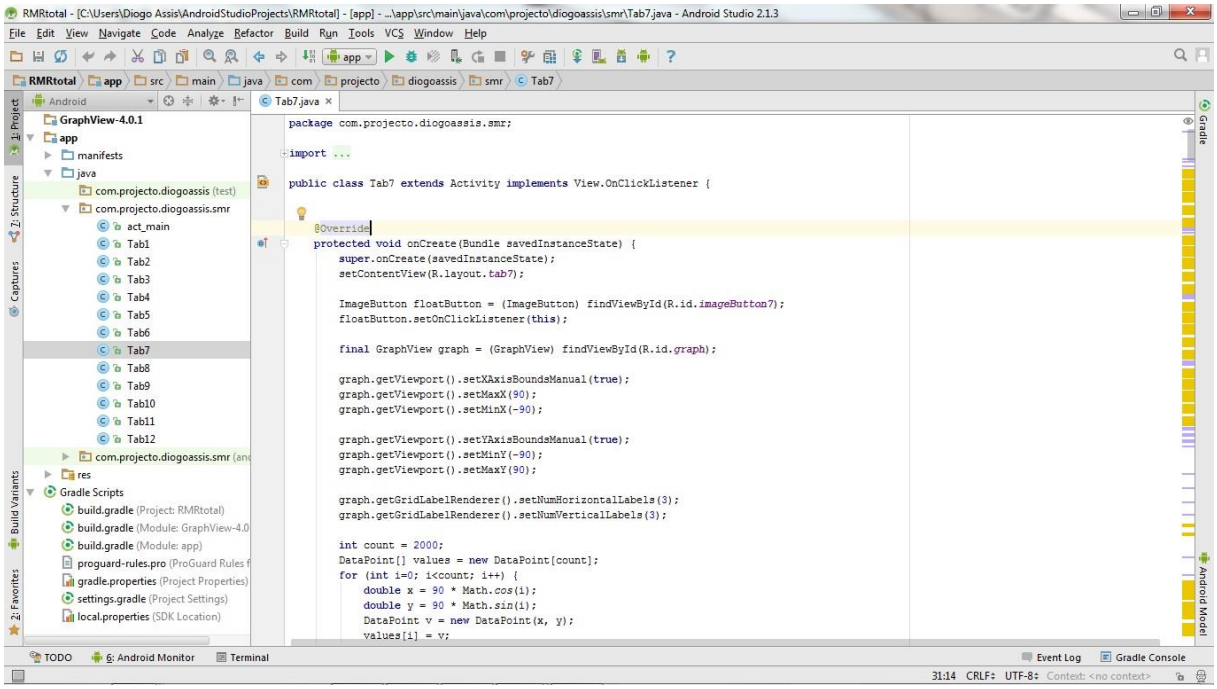


Fig. IX. 1 – Código fonte da atividade *Tab7*

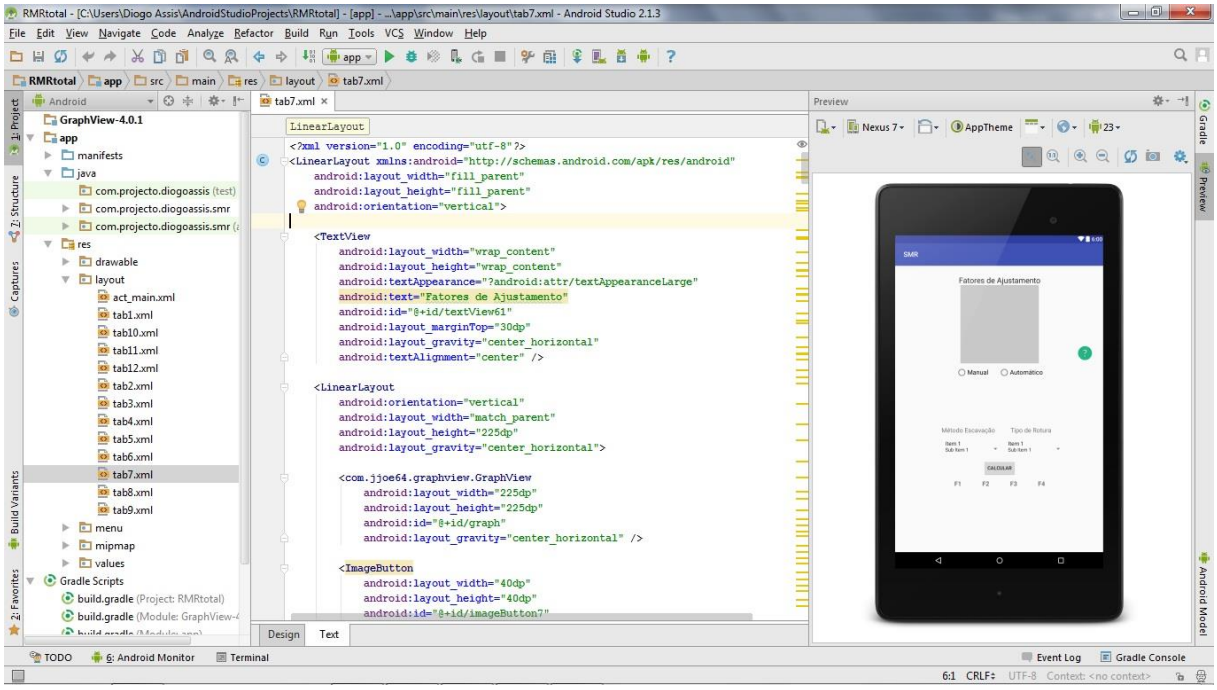


Fig. IX. 2 – Código XML do *layout Tab7*

```
public class Tab7 extends Activity implements View.OnClickListener {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab7);

        ImageButton floatButton = (ImageButton)
findViewById(R.id.imageButton7);
        floatButton.setOnClickListener(this);

        final GraphView graph = (GraphView) findViewById(R.id.graph);

        graph.getViewport().setXAxisBoundsManual(true);
        graph.getViewport().setMaxX(90);
        graph.getViewport().setMinX(-90);

        graph.getViewport().setYAxisBoundsManual(true);
        graph.getViewport().setMinY(-90);
        graph.getViewport().setMaxY(90);

        graph.getGridLabelRenderer().setNumHorizontalLabels(3);
        graph.getGridLabelRenderer().setNumVerticalLabels(3);

        int count = 2000;
        DataPoint[] values = new DataPoint[count];
        for (int i=0; i<count; i++) {
            double x = 90 * Math.cos(i);
            double y = 90 * Math.sin(i);
            DataPoint v = new DataPoint(x, y);
            values[i] = v;
        }
        PointsGraphSeries<DataPoint> circulo = new
PointsGraphSeries<>(values);
        circulo.setSize(1);
        circulo.setColor(Color.BLACK);
        graph.addSeries(circulo);

        graph.getGridLabelRenderer().setVerticalLabelsVisible(false);
        graph.getGridLabelRenderer().setHorizontalLabelsVisible(false);

        final SeekBar sb4 = (SeekBar) findViewById(R.id.seekBar4);
        final SeekBar sb5 = (SeekBar) findViewById(R.id.seekBar5);
        final SeekBar sb6 = (SeekBar) findViewById(R.id.seekBar6);
        final SeekBar sb7 = (SeekBar) findViewById(R.id.seekBar7);

        final TextView tv63 = (TextView) findViewById(R.id.textView63);
        final TextView tv64 = (TextView) findViewById(R.id.textView64);
        final TextView tv68 = (TextView) findViewById(R.id.textView68);
        final TextView tv69 = (TextView) findViewById(R.id.textView69);

        final TextView tv71 = (TextView) findViewById(R.id.textView71);
        final TextView tv72 = (TextView) findViewById(R.id.textView72);
        final TextView tv76 = (TextView) findViewById(R.id.textView76);
        final TextView tv77 = (TextView) findViewById(R.id.textView77);

        final TextView tv86 = (TextView) findViewById(R.id.textView86);
        final TextView tv89 = (TextView) findViewById(R.id.textView89);
```

```
        final TextView tv92 = (TextView) findViewById(R.id.textView92);
        final TextView tv95 = (TextView) findViewById(R.id.textView95);

        final Spinner spinner10;
        ArrayAdapter<CharSequence> adapter10;
        final Spinner spinner9;
        ArrayAdapter<CharSequence> adapter9;

        final RadioButton radioButton10 = (RadioButton)
findViewById(R.id.radioButton10);
        final RadioButton radioButton11 = (RadioButton)
findViewById(R.id.radioButton11);

        final LinearLayout input1 = (LinearLayout)
findViewById(R.id.input1);
        final LinearLayout input2 = (LinearLayout)
findViewById(R.id.input2);
        final LinearLayout input3 = (LinearLayout)
findViewById(R.id.input3);

        input1.setVisibility(View.INVISIBLE);
        input2.setVisibility(View.INVISIBLE);
        input3.setVisibility(View.INVISIBLE);


        radioButton10.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                input1.setVisibility(View.VISIBLE);
                input2.setVisibility(View.INVISIBLE);
                input3.setVisibility(View.VISIBLE);
                LinearLayout.LayoutParams params =
(LinearLayout.LayoutParams)input3.getLayoutParams();
                params.setMargins(0, 200, 0, 0);
                input3.setLayoutParams(params);
            }
        });

        radioButton11.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                input1.setVisibility(View.INVISIBLE);
                input2.setVisibility(View.VISIBLE);
                input3.setVisibility(View.VISIBLE);
                LinearLayout.LayoutParams params =
(LinearLayout.LayoutParams)input3.getLayoutParams();
                params.setMargins(0, 0, 0, 0);
                input3.setLayoutParams(params);

            }
        });

        sb4.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                double dir = sb4.getProgress()-90;
                double inc = sb5.getProgress()-90;
```

```
                if(dir == 0){
                    String a = "N";
                    String b = "S";
                    tv68.setText("W");
                    tv69.setText("E");
                    if (inc < 0) {
                        String c = "W";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double beta = Math.abs(inc);
                        double alfa = dir + 270;
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                    ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "E";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 90;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                    ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(b));
                        tv64.setText("");
                    }
                }
                if (dir > 0){
                    String a = "N";
                    String b = "E";
                    tv68.setText("NW");
                    tv69.setText("SE");
                    if (inc < 0) {
                        String c = "NW";
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir)))
                                    + String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 270;
                        double beta = Math.abs(inc);
```

```
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";"
                                + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "SE";
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 90;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) + String.valueOf(b));
                        tv64.setText("");
                    }
                }
                if(dir < 0) {
                    String a = "N";
                    String b = "W";
                    tv68.setText("SW");
                    tv69.setText("NE");
                    if (inc < 0) {
                        String c = "SW";
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 270;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "NE";
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
```

```
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 90 - Math.abs(dir);
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) + String.valueOf(b));
                        tv64.setText("");
                    }
                }
                if (dir == 90)
                {
                    String a = "E";
                    String b = "W";
                    tv68.setText("N");
                    tv69.setText("S");
                    if (inc < 0) {
                        String c = "N";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 0;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                    ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "S";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 180;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                    ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
```

```
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(b));
                        tv64.setText("");
                    }
                }
                if (dir == -90)
                {
                    String a = "E";
                    String b = "W";
                    tv68.setText("S");
                    tv69.setText("N");
                    if (inc < 0) {
                        String c = "S";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 180;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "N";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 0;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(b));
                        tv64.setText("");
                    }
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
```

```
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });

        sb5.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                double dir = sb4.getProgress()-90;
                double inc = sb5.getProgress()-90;
                if(dir == 0){
                    String a = "N";
                    String b = "S";
                    tv68.setText("W");
                    tv69.setText("E");
                    if (inc < 0) {
                        String c = "W";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double beta = Math.abs(inc);
                        double alfa = dir + 270;
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "E";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 90;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(b));
                        tv64.setText("");
                    }
```

```
                }
                if (dir > 0){
                    String a = "N";
                    String b = "E";
                    tv68.setText("NW");
                    tv69.setText("SE");
                    if (inc < 0) {
                        String c = "NW";
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir)))
                                + String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 270;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β="
                                +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "SE";
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 90;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) + String.valueOf(b));
                        tv64.setText("");
                    }
                }
                if(dir < 0) {
                    String a = "N";
                    String b = "W";
                    tv68.setText("SW");
                    tv69.setText("NE");
                    if (inc < 0) {
                        String c = "SW";
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) + String.valueOf(b)
                                + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 270;
```

```
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "NE";
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 90 - Math.abs(dir);
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";"
                                + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) + String.valueOf(b));
                        tv64.setText("");
                    }
                }
                if (dir == 90)
                {
                    String a = "E";
                    String b = "W";
                    tv68.setText("N");
                    tv69.setText("S");
                    if (inc < 0) {
                        String c = "N";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 0;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "S";
```

```
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 180;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                     "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(b));
                        tv64.setText("");
                    }
                }
                if (dir == -90)
                {
                    String a = "E";
                    String b = "W";
                    tv68.setText("S");
                    tv69.setText("N");
                    if (inc < 0) {
                        String c = "S";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 180;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                     "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "N";
                        tv63.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 0;
                        double beta = Math.abs(inc);
                        tv64.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                     ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv86.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));
```

```
tv89.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv63.setText(String.valueOf(a) +
String.valueOf(b));
                        tv64.setText("");
                    }
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });

        sb6.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                double dir = sb6.getProgress()-90;
                double inc = sb7.getProgress()-90;
                if(dir == 0){
                    String a = "N";
                    String b = "S";
                    tv76.setText("W");
                    tv77.setText("E");
                    if (inc < 0) {
                        String c = "W";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double beta = Math.abs(inc);
                        double alfa = dir + 270;
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) +
                                ";" + "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "E";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 90;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
```

```
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv71.setText(String.valueOf(a) +
String.valueOf(b));
                        tv72.setText("");
                    }
                }
                if (dir > 0){
                    String a = "N";
                    String b = "E";
                    tv76.setText("NW");
                    tv77.setText("SE");
                    if (inc < 0) {
                        String c = "NW";
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
                                    String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 270;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β="
                                    +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "SE";
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
                                    String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 90;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) + String.valueOf(b));
                        tv72.setText("");
```

```
                    }
                }
                if(dir < 0) {
                    String a = "N";
                    String b = "W";
                    tv76.setText("SW");
                    tv77.setText("NE");
                    if (inc < 0) {
                        String c = "SW";
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f", Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f", Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 270;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "NE";
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f", Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f", Math.abs(inc))) + String.valueOf(c));
                        double alfa = 90 - Math.abs(dir);
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0) {
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f", Math.abs(dir))) + String.valueOf(b));
                        tv72.setText("");
                    }
                }
                if (dir == 90)
                {
                    String a = "E";
                    String b = "W";
                    tv76.setText("N");
                    tv77.setText("S");
                    if (inc < 0) {
                        String c = "N";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
```

```
                        double alfa = 0;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                     "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "S";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 180;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β="
                                     +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv71.setText(String.valueOf(a) +
String.valueOf(b));
                        tv72.setText("");
                    }
                }
                if (dir == -90)
                {
                    String a = "E";
                    String b = "W";
                    tv76.setText("S");
                    tv77.setText("N");
                    if (inc < 0) {
                        String c = "S";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 180;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                     "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
```

```
                        String c = "N";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 0;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv71.setText(String.valueOf(a) +
String.valueOf(b));
                        tv72.setText("");
                    }
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        sb7.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                double dir = sb6.getProgress()-90;
                double inc = sb7.getProgress()-90;
                if(dir == 0){
                    String a = "N";
                    String b = "S";
                    tv76.setText("W");
                    tv77.setText("E");
                    if (inc < 0) {
                        String c = "W";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double beta = Math.abs(inc);
                        double alfa = dir + 270;
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                "β=" +
```

```
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "E";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 90;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" +
                                "β=" +
String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv71.setText(String.valueOf(a) +
String.valueOf(b));
                        tv72.setText("");
                    }
                }
                if (dir > 0){
                    String a = "N";
                    String b = "E";
                    tv76.setText("NW");
                    tv77.setText("SE");
                    if (inc < 0) {
                        String c = "NW";
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 270;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "SE";
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 90;
```

```
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f",Math.abs(dir))) + String.valueOf(b));
                        tv72.setText("");
                    }
                }
                if(dir < 0) {
                    String a = "N";
                    String b = "W";
                    tv76.setText("SW");
                    tv77.setText("NE");
                    if (inc < 0) {
                        String c = "SW";
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f", Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f", Math.abs(inc))) + String.valueOf(c));
                        double alfa = dir + 270;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "NE";
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f", Math.abs(dir))) +
                                String.valueOf(b) + ";" +
String.valueOf(String.format("%2.0f", Math.abs(inc))) + String.valueOf(c));
                        double alfa = 90 - Math.abs(dir);
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0) {
                        tv71.setText(String.valueOf(a) +
String.valueOf(String.format("%2.0f", Math.abs(dir))) + String.valueOf(b));
```

```
                        tv72.setText("");
                    }
                }
                if (dir == 90)
                {
                    String a = "E";
                    String b = "W";
                    tv76.setText("N");
                    tv77.setText("S");
                    if (inc < 0) {
                        String c = "N";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 0;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "S";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 180;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv71.setText(String.valueOf(a) +
String.valueOf(b));
                        tv72.setText("");
                    }
                }
                if (dir == -90)
                {
                    String a = "E";
                    String b = "W";
                    tv76.setText("S");
                    tv77.setText("N");
                    if (inc < 0) {
                        String c = "S";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
```

```
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 180;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc > 0) {
                        String c = "N";
                        tv71.setText(String.valueOf(a) + String.valueOf(b)
+ ";" +

String.valueOf(String.format("%2.0f",Math.abs(inc))) + String.valueOf(c));
                        double alfa = 0;
                        double beta = Math.abs(inc);
                        tv72.setText("  α=" +
String.valueOf(String.format("%2.0f",Math.abs(alfa))) + ";" + "β=" +

String.valueOf(String.format("%2.0f",Math.abs(beta))));

tv92.setText(String.valueOf(String.format("%2.0f",Math.abs(alfa))));

tv95.setText(String.valueOf(String.format("%2.0f",Math.abs(beta))));
                    }
                    if (inc == 0)
                    {
                        tv71.setText(String.valueOf(a) +
String.valueOf(b));
                        tv72.setText("");
                    }
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        spinner9 = (Spinner) findViewById(R.id.spinner9);
        adapter9 = ArrayAdapter.createFromResource(this, R.array.metodo,
android.R.layout.simple_spinner_item);

adapter9.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem);
        spinner9.setAdapter(adapter9);
```

```
        spinner9.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {

            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        spinner10 = (Spinner) findViewById(R.id.spinner10);
        adapter10 = ArrayAdapter.createFromResource(this, R.array.esc,
android.R.layout.simple_spinner_item);

adapter10.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
        spinner10.setAdapter(adapter10);

        spinner10.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {

            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });

        Button button8 = (Button) findViewById(R.id.button8);
        button8.setOnClickListener(this);
        Button button9 = (Button) findViewById(R.id.button9);
        button9.setOnClickListener(this);
    }


    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button9: {
                final RadioButton radioButton10 = (RadioButton)
findViewById(R.id.radioButton10);
                final RadioButton radioButton11 = (RadioButton)
findViewById(R.id.radioButton11);

                final TextView tv85 = (TextView)
findViewById(R.id.textView85);
                final TextView tv88 = (TextView)
findViewById(R.id.textView88);
                final TextView tv91 = (TextView)
findViewById(R.id.textView91);
                final TextView tv94 = (TextView)
findViewById(R.id.textView94);
```

```
                final TextView tv86 = (TextView)
findViewById(R.id.textView86);
                final TextView tv89 = (TextView)
findViewById(R.id.textView89);
                final TextView tv92 = (TextView)
findViewById(R.id.textView92);
                final TextView tv95 = (TextView)
findViewById(R.id.textView95);

                final TextView tv79 =
(TextView)findViewById(R.id.textView79);
                final TextView tv81 =
(TextView)findViewById(R.id.textView81);

                final SeekBar sb4 = (SeekBar) findViewById(R.id.seekBar4);
                final SeekBar sb5 = (SeekBar) findViewById(R.id.seekBar5);
                final SeekBar sb6 = (SeekBar) findViewById(R.id.seekBar6);
                final SeekBar sb7 = (SeekBar) findViewById(R.id.seekBar7);

                final GraphView graph = (GraphView)
findViewById(R.id.graph);

                if (radioButton10.isChecked())
                {
                    graph.removeAllSeries();
                    int count = 2000;
                    DataPoint[] values = new DataPoint[count];
                    for (int i=0; i<count; i++) {
                        double w = 90 * Math.cos(i);
                        double z = 90 * Math.sin(i);
                        DataPoint t = new DataPoint(w, z);
                        values[i] = t;
                    }
                    PointsGraphSeries<DataPoint> circulo = new
PointsGraphSeries<>(values);
                    circulo.setSize(1);
                    circulo.setColor(Color.BLACK);
                    graph.addSeries(circulo);


                    final double b = (sb5.getProgress()-90);
                    final double a = sb4.getProgress() - 90;
                    double k = 90 * Math.cos(Math.toRadians(-a+90));
                    double l = 90 * Math.sin(Math.toRadians(-a+90));

                    if(b>0) {
                        double x =  (90-b) * Math.cos(Math.toRadians(-a));
                        double y =  (90-b) * Math.sin(Math.toRadians(-a));

                        double x1 =  15 * Math.cos(Math.toRadians(-a));
                        double y1 =  15 * Math.sin(Math.toRadians(-a));

                        double x2 =  b * Math.cos(Math.toRadians(-a+180));
                        double y2 =  b * Math.sin(Math.toRadians(-a+180));



                        LineGraphSeries<DataPoint> series = new
```

```
LineGraphSeries<>(new DataPoint[]{
                                new DataPoint(k, l),
                                new DataPoint(0, 0),
                                new DataPoint(x1, y1),
                                new DataPoint(0, 0),
                                new DataPoint(-k,-l),
                        });
                        series.setThickness(5);
                        series.setColor(Color.argb(255,9,157,9));
                        graph.addSeries(series);

                        PointsGraphSeries<DataPoint> series3 = new
PointsGraphSeries<DataPoint>(new DataPoint[]{
                                new DataPoint(x, y),
                                new DataPoint(x2, y2),
                        });
                        series3.setColor(Color.argb(255,9,157,9));
                        graph.addSeries(series3);
                        series3.setCustomShape(new
PointsGraphSeries.CustomShape() {
                            @Override
                            public void draw(Canvas canvas, Paint paint,
float x, float y, DataPointInterface dataPoint) {
                                paint.setStrokeWidth(3);
                                canvas.drawLine(x - 5, y - 5, x + 5, y + 5,
paint);
                                canvas.drawLine(x + 5, y - 5, x - 5, y + 5,
paint);
                            }
                        });


                    }

                    if (b<0)
                    {
                        double x =  (-90-b) * Math.cos(Math.toRadians(-a));
                        double y =  (-90-b) * Math.sin(Math.toRadians(-a));

                        double x1 =  -15 * Math.cos(Math.toRadians(-a));
                        double y1 =  -15 * Math.sin(Math.toRadians(-a));

                        double x2 =  b * Math.cos(Math.toRadians(-a+180));
                        double y2 =  b * Math.sin(Math.toRadians(-a-180));

                        LineGraphSeries<DataPoint> series = new
LineGraphSeries<>(new DataPoint[]{
                                new DataPoint(k, l),
                                new DataPoint(0, 0),
                                new DataPoint(x1, y1),
                                new DataPoint(0, 0),
                                new DataPoint(-k,-l),
                        });
                        series.setThickness(5);
                        series.setColor(Color.argb(255,9,157,9));
                        graph.addSeries(series);

                        PointsGraphSeries<DataPoint> series3 = new
PointsGraphSeries<DataPoint>(new DataPoint[]{
```

```
                                new DataPoint(x, y),
                                new DataPoint(x2, y2),
                            });
                            series3.setColor(Color.argb(255,9,157,9));
                            graph.addSeries(series3);
                            series3.setCustomShape(new
PointsGraphSeries.CustomShape() {
                                @Override
                                public void draw(Canvas canvas, Paint paint,
float x, float y, DataPointInterface dataPoint) {
                                    paint.setStrokeWidth(3);
                                    canvas.drawLine(x - 5, y - 5, x + 5, y + 5,
paint);
                                    canvas.drawLine(x + 5, y - 5, x - 5, y + 5,
paint);
                                }
                            });

                        }

                        final double c = sb6.getProgress() - 90;
                        final double d = (sb7.getProgress()-90);
                        double p = 90 * Math.cos(Math.toRadians(-c+90));
                        double o = 90 * Math.sin(Math.toRadians(-c+90));

                        if (d>0){
                            double e = (90-d) * Math.cos(Math.toRadians(-c));
                            double f = (90-d) * Math.sin(Math.toRadians(-c));

                            double e1 = 15 * Math.cos(Math.toRadians(-c));
                            double f1 = 15 * Math.sin(Math.toRadians(-c));

                            double e2 = d * Math.cos(Math.toRadians(-c+180));
                            double f2 = d * Math.sin(Math.toRadians(-c+180));

                            LineGraphSeries<DataPoint> series2 = new
LineGraphSeries<>(new DataPoint[]{
                                    new DataPoint(p, o),
                                    new DataPoint(0, 0),
                                    new DataPoint(e1, f1),
                                    new DataPoint(0, 0),
                                    new DataPoint(-p,-o),
                            });
                            series2.setThickness(5);
                            series2.setColor(Color.argb(255,242,17,17));
                            graph.addSeries(series2);


                            PointsGraphSeries<DataPoint> series4 = new
PointsGraphSeries<DataPoint>(new DataPoint[]{
                                    new DataPoint(e, f),
                                    new DataPoint(e2, f2),
                            });
                            graph.addSeries(series4);
                            series4.setColor(Color.argb(255,242,17,17));
                            series4.setCustomShape(new
PointsGraphSeries.CustomShape() {
                                @Override
                                public void draw(Canvas canvas, Paint paint,
```

```
float x, float y, DataPointInterface dataPoint) {
                                paint.setStrokeWidth(3);
                                canvas.drawLine(x - 5, y - 5, x + 5, y + 5,
paint);
                                canvas.drawLine(x + 5, y - 5, x - 5, y + 5,
paint);
                            }
                        });
                    }

                    if (d<0){
                        double e = (-90-d) * Math.cos(Math.toRadians(-c));
                        double f = (-90-d) * Math.sin(Math.toRadians(-c));

                        double e1 = -15 * Math.cos(Math.toRadians(-c));
                        double f1 = -15 * Math.sin(Math.toRadians(-c));

                        double e2 = d * Math.cos(Math.toRadians(-c+180));
                        double f2 = d * Math.sin(Math.toRadians(-c+180));

                        LineGraphSeries<DataPoint> series2 = new
LineGraphSeries<>(new DataPoint[]{
                                new DataPoint(p, o),
                                new DataPoint(0, 0),
                                new DataPoint(e1, f1),
                                new DataPoint(0, 0),
                                new DataPoint(-p,-o),
                        });
                        series2.setThickness(5);
                        series2.setColor(Color.argb(255,242,17,17));
                        graph.addSeries(series2);

                        PointsGraphSeries<DataPoint> series4 = new
PointsGraphSeries<DataPoint>(new DataPoint[]{
                                new DataPoint(e, f),
                                new DataPoint(e2, f2),
                        });
                        graph.addSeries(series4);
                        series4.setColor(Color.argb(255,242,17,17));
                        series4.setCustomShape(new
PointsGraphSeries.CustomShape() {
                            @Override
                            public void draw(Canvas canvas, Paint paint,
float x, float y, DataPointInterface dataPoint) {
                                paint.setStrokeWidth(3);
                                canvas.drawLine(x - 5, y - 5, x + 5, y + 5,
paint);
                                canvas.drawLine(x + 5, y - 5, x - 5, y + 5,
paint);
                            }
                        });
                    }

                }


                if (radioButton11.isChecked())
                {
                    SharedPreferences az1 = getSharedPreferences("az1", 0);
```

```
                    String az1f = az1.getString("az1f", "0");
                    SharedPreferences pendor1 =
getSharedPreferences("pendor1", 0);
                    String pendor1f = pendor1.getString("pendor1f", "0");
                    SharedPreferences queda1 =
getSharedPreferences("queda1", 0);
                    String queda1f = queda1.getString("queda1f", "0");
                    SharedPreferences grau10 =
getSharedPreferences("grau10", 0);
                    String grau1f = grau10.getString("grau1f", "0");

                    double a11 = Double.parseDouble(grau1f) + 90;
                    double b11 = Double.parseDouble(pendor1f);

                    if(a11>=360){
                        double a11f = a11 - 360;
                        tv79.setText(String.valueOf(az1f) + ";" +
String.valueOf(pendor1f) +
                                String.valueOf(queda1f) + "  α=" +
String.valueOf(String.format("%3.0f", a11f))
                                + ";" + "β=" +
String.valueOf(String.format("%2.0f", b11)));
                    } else {
                        tv79.setText(String.valueOf(az1f) + ";" +
String.valueOf(pendor1f) +
                                String.valueOf(queda1f) + "  α=" +
String.valueOf(String.format("%3.0f", a11))
                                + ";" + "β=" +
String.valueOf(String.format("%2.0f", b11)));
                    }

                    SharedPreferences az2 = getSharedPreferences("az2", 0);
                    String az2f = az2.getString("az2f", "0");
                    SharedPreferences pendor2 =
getSharedPreferences("pendor2", 0);
                    String pendor2f = pendor2.getString("pendor2f", "0");
                    SharedPreferences queda2 =
getSharedPreferences("queda2", 0);
                    String queda2f = queda2.getString("queda2f", "0");
                    SharedPreferences grau20 =
getSharedPreferences("grau20", 0);
                    String grau2f = grau20.getString("grau2f", "0");

                    double c11 = Double.parseDouble(grau2f) + 90;
                    double d11 = Double.parseDouble(pendor2f);


                    if(c11>=360){
                        double c11f = c11-360;
                        tv81.setText(String.valueOf(az2f) + ";" +
String.valueOf(pendor2f) + String.valueOf(queda2f)
                                + "  α=" +
String.valueOf(String.format("%3.0f",c11f)) + ";" + "β=" +

String.valueOf(String.format("%2.0f",d11)));
                    } else {
                        tv81.setText(String.valueOf(az2f) + ";" +
String.valueOf(pendor2f) + String.valueOf(queda2f) + "  " +
                                "α=" +
```

```
String.valueOf(String.format("%3.0f", c11)) + ";" + "β=" +
String.valueOf(String.format("%2.0f", d11)));
                    }

                    graph.removeAllSeries();

                    int count = 2000;
                    DataPoint[] values = new DataPoint[count];
                    for (int i=0; i<count; i++) {
                        double w = 90 * Math.cos(i);
                        double z = 90 * Math.sin(i);
                        DataPoint t = new DataPoint(w, z);
                        values[i] = t;
                    }
                    PointsGraphSeries<DataPoint> circulo = new
PointsGraphSeries<>(values);
                    circulo.setSize(1);
                    circulo.setColor(Color.BLACK);
                    graph.addSeries(circulo);


                    final double a = Double.parseDouble(grau1f)+90;
                    final double b = Double.parseDouble(pendor1f);

                    double k = 90 * Math.sin(Math.toRadians(a+90));
                    double l = 90 * Math.cos(Math.toRadians(a+90));

                    double x =  (90-b) * Math.sin(Math.toRadians(a));
                    double y =  (90-b) * Math.cos(Math.toRadians(a));

                    double x1 =  15 * Math.sin(Math.toRadians(a));
                    double y1 =  15 * Math.cos(Math.toRadians(a));

                    double x2 =  b * Math.sin(Math.toRadians(a+180));
                    double y2 =  b * Math.cos(Math.toRadians(a+180));

                    if(a>360){
                        double ac = a - 360;
                        tv86.setText(String.valueOf(ac));
                    }
                    if (a<360){
                    tv86.setText(String.valueOf(a));
                    }
                    tv89.setText(String.valueOf(b));


                    LineGraphSeries<DataPoint> series = new
LineGraphSeries<>(new DataPoint[]{
                            new DataPoint(k, l),
                            new DataPoint(0, 0),
                            new DataPoint(x1, y1),
                            new DataPoint(0, 0),
                            new DataPoint(-k,-l),
                    });
                    series.setThickness(5);
                    series.setColor(Color.argb(255,9,157,9));
                    graph.addSeries(series);

                    PointsGraphSeries<DataPoint> series3 = new
```

```
PointsGraphSeries<DataPoint>(new DataPoint[]{
                            new DataPoint(x, y),
                            new DataPoint(x2, y2),
                    });
                    series3.setColor(Color.argb(255,9,157,9));
                    graph.addSeries(series3);
                    series3.setCustomShape(new
PointsGraphSeries.CustomShape() {
                        @Override
                        public void draw(Canvas canvas, Paint paint, float
x, float y, DataPointInterface dataPoint) {
                            paint.setStrokeWidth(3);
                            canvas.drawLine(x - 5, y - 5, x + 5, y + 5,
paint);
                            canvas.drawLine(x + 5, y - 5, x - 5, y + 5,
paint);
                        }
                    });

                    final double c = Double.parseDouble(grau2f)+90;
                    final double d = Double.parseDouble(pendor2f);

                    double e = (90-d) * Math.sin(Math.toRadians(c));
                    double f = (90-d) * Math.cos(Math.toRadians(c));

                    double e1 = 15 * Math.sin(Math.toRadians(c));
                    double f1 = 15 * Math.cos(Math.toRadians(c));

                    double e2 = d * Math.sin(Math.toRadians(c+180));
                    double f2 = d * Math.cos(Math.toRadians(c+180));

                    double p = 90 * Math.sin(Math.toRadians(c+90));
                    double o = 90 * Math.cos(Math.toRadians(c+90));

                    if(c>360){
                        double cc = c - 360;
                        tv92.setText(String.valueOf(cc));
                    } if (c<360){
                    tv92.setText(String.valueOf(c));
                    }

                    tv95.setText(String.valueOf(d));

                    LineGraphSeries<DataPoint> series2 = new
LineGraphSeries<>(new DataPoint[]{
                            new DataPoint(p, o),
                            new DataPoint(0, 0),
                            new DataPoint(e1, f1),
                            new DataPoint(0, 0),
                            new DataPoint(-p,-o),
                    });
                    series2.setThickness(5);
                    series2.setColor(Color.argb(255,242,17,17));
                    graph.addSeries(series2);



                    PointsGraphSeries<DataPoint> series4 = new
PointsGraphSeries<DataPoint>(new DataPoint[]{
```

```
                            new DataPoint(e, f),
                            new DataPoint(e2, f2),
                    });
                    graph.addSeries(series4);
                    series4.setColor(Color.argb(255,242,17,17));
                    series4.setCustomShape(new
PointsGraphSeries.CustomShape() {
                        @Override
                        public void draw(Canvas canvas, Paint paint, float
x, float y, DataPointInterface dataPoint) {
                            paint.setStrokeWidth(3);
                            canvas.drawLine(x - 5, y - 5, x + 5, y + 5,
paint);
                            canvas.drawLine(x + 5, y - 5, x - 5, y + 5,
paint);
                        }
                    });

                }

                final double as =
Double.parseDouble(tv86.getText().toString());
                final double bs =
Double.parseDouble(tv89.getText().toString());
                final double aj =
Double.parseDouble(tv92.getText().toString());
                final double bj =
Double.parseDouble(tv95.getText().toString());

                final Spinner spinner10;
                spinner10 = (Spinner) findViewById(R.id.spinner10);
                String esc = spinner10.getSelectedItem().toString();

                if (esc.equals("Planar")){
                    double A = Math.abs(aj-as);
                    double B = bj;
                    double C = bj-bs;
                    if (A<=90){
                        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A -
17)) * 180/Math.PI;

tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
                        SharedPreferences f1 = getSharedPreferences("f1",
0);
                        SharedPreferences.Editor editor7 = f1.edit();
                        editor7.putString("fumf", String.valueOf(f1i));
                        editor7.commit();
                    }
                    if (A>90 && A<=180){
                        double A1 = 180 - A;
                        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 -
17)) * 180/Math.PI;

tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
                        SharedPreferences f1 = getSharedPreferences("f1",
0);
                        SharedPreferences.Editor editor7 = f1.edit();
                        editor7.putString("fumf", String.valueOf(f1i));
                        editor7.commit();
```

```
                    }
                    if (A>180 && A<=270){
                        double A1 = A - 180;
                        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 -
17)) * 180/Math.PI;

tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
                        SharedPreferences f1 = getSharedPreferences("f1",
0);
                        SharedPreferences.Editor editor7 = f1.edit();
                        editor7.putString("fumf", String.valueOf(f1i));
                        editor7.commit();
                    }
                    if (A>270 && A<=360){
                        double A1 = 360 - A;
                        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 -
17)) * 180/Math.PI;

tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
                        SharedPreferences f1 = getSharedPreferences("f1",
0);
                        SharedPreferences.Editor editor7 = f1.edit();
                        editor7.putString("fumf", String.valueOf(f1i));
                        editor7.commit();
                    }

                    double f2i = 0.5625 + 0.00512821 * Math.atan(0.17 * bj
- 5) * 180/Math.PI;
                    double f3i = - 30 + 0.333333 * Math.atan(bj-bs) *
180/Math.PI;

tv88.setText(String.valueOf(String.format("%2.3f",f2i)));

tv91.setText(String.valueOf(String.format("%2.3f",f3i)));
                    SharedPreferences f2 = getSharedPreferences("f2", 0);
                    SharedPreferences.Editor editor8 = f2.edit();
                    editor8.putString("fdoisf", String.valueOf(f2i));
                    editor8.commit();
                    SharedPreferences f3 = getSharedPreferences("f3", 0);
                    SharedPreferences.Editor editor9 = f3.edit();
                    editor9.putString("ftresf", String.valueOf(f3i));
                    editor9.commit();
                }

                if (esc.equals("Toppling")){
                    double A = Math.abs(aj-as-180);
                    double C = bj+bs;
                    if (A<=90){
                        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A -
17)) * 180/Math.PI;

tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
                        SharedPreferences f1 = getSharedPreferences("f1",
0);
                        SharedPreferences.Editor editor7 = f1.edit();
                        editor7.putString("fumf", String.valueOf(f1i));
                        editor7.commit();
                    }
                    if (A>90 && A<=180){
```

```
                        double A1 = 180 - A;
                        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 -
17)) * 180/Math.PI;

tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
                        SharedPreferences f1 = getSharedPreferences("f1",
0);
                        SharedPreferences.Editor editor7 = f1.edit();
                        editor7.putString("fumf", String.valueOf(f1i));
                        editor7.commit();
                    }
                    if (A>180 && A<=270){
                        double A1 = A - 180;
                        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 -
17)) * 180/Math.PI;

tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
                        SharedPreferences f1 = getSharedPreferences("f1",
0);
                        SharedPreferences.Editor editor7 = f1.edit();
                        editor7.putString("fumf", String.valueOf(f1i));
                        editor7.commit();
                    }
                    if (A>270 && A<=360){
                        double A1 = 360 - A;
                        double f1i = 0.64 - 0.006 * Math.atan (0.1 * (A1 -
17)) * 180/Math.PI;

tv85.setText(String.valueOf(String.format("%2.3f",f1i)));
                        SharedPreferences f1 = getSharedPreferences("f1",
0);
                        SharedPreferences.Editor editor7 = f1.edit();
                        editor7.putString("fumf", String.valueOf(f1i));
                        editor7.commit();
                    }
                    double f2i = 1;
                    double f3i = -13 - 0.142857 * Math.atan(C-120) *
180/Math.PI;

tv88.setText(String.valueOf(String.format("%1.0f",f2i)));

tv91.setText(String.valueOf(String.format("%2.3f",f3i)));
                    SharedPreferences f2 = getSharedPreferences("f2", 0);
                    SharedPreferences.Editor editor8 = f2.edit();
                    editor8.putString("fdoisf", String.valueOf(f2i));
                    editor8.commit();
                    SharedPreferences f3 = getSharedPreferences("f3", 0);
                    SharedPreferences.Editor editor9 = f3.edit();
                    editor9.putString("ftresf", String.valueOf(f3i));
                    editor9.commit();
                }
                if (esc.equals("Escolha")){
                    tv85.setText("");
                    tv88.setText("");
                    tv91.setText("");
                    tv94.setText("");
                }

                final Spinner spinner9;
```

```
                spinner9 = (Spinner) findViewById(R.id.spinner9);
                String met = spinner9.getSelectedItem().toString();
                if (met.equals("Escolha")) {
                    tv94.setText("");
                } else if (met.equals("Taludes Naturais")) {
                    double f4i = 15;

tv94.setText(String.valueOf(String.format("%2.0f",f4i)));
                    SharedPreferences f4 = getSharedPreferences("f4", 0);
                    SharedPreferences.Editor editor10 = f4.edit();
                    editor10.putString("fquatrof", String.valueOf(f4i));
                    editor10.commit();
                } else if (met.equals("Pré-corte")) {
                    double f4i = 10;

tv94.setText(String.valueOf(String.format("%2.0f",f4i)));
                    SharedPreferences f4 = getSharedPreferences("f4", 0);
                    SharedPreferences.Editor editor10 = f4.edit();
                    editor10.putString("fquatrof", String.valueOf(f4i));
                    editor10.commit();
                } else if (met.equals("Smooth Blasting")) {
                    double f4i = 8;

tv94.setText(String.valueOf(String.format("%1.0f",f4i)));
                    SharedPreferences f4 = getSharedPreferences("f4", 0);
                    SharedPreferences.Editor editor10 = f4.edit();
                    editor10.putString("fquatrof", String.valueOf(f4i));
                    editor10.commit();
                } else if (met.equals("Explosivos ou Mecânico")) {
                    double f4i = 0;

tv94.setText(String.valueOf(String.format("%1.0f",f4i)));
                    SharedPreferences f4 = getSharedPreferences("f4", 0);
                    SharedPreferences.Editor editor10 = f4.edit();
                    editor10.putString("fquatrof", String.valueOf(f4i));
                    editor10.commit();
                } else if (met.equals("Desmonte Deficiente")) {
                    double f4i = -8;

tv94.setText(String.valueOf(String.format("%1.0f",f4i)));
                    SharedPreferences f4 = getSharedPreferences("f4", 0);
                    SharedPreferences.Editor editor10 = f4.edit();
                    editor10.putString("fquatrof", String.valueOf(f4i));
                    editor10.commit();
                }

                break;
            }

            case R.id.button8: {
                Intent i = new Intent(getApplicationContext(), Tab9.class);
                startActivity(i);

            }
            break;
            case R.id.imageButton7:{
                AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
                dlg.setTitle("Fatores de Ajustamento");
                dlg.setMessage("Defina as orientações do talude e da
```

```
descontinuidade, introduzindo o seu valor ou utilizando a Bússola de
Geólogo." +
                        "\nPosteriormente, escolha o método de escavação e
o tipo de rotura em análise." +
                        "\nPressione o botão Calcular para a apresentação
dos fatores de ajustamento.");
                dlg.show();
            }
            break;
        }
    }
```

ANEXO X
SMR – *Tab8*

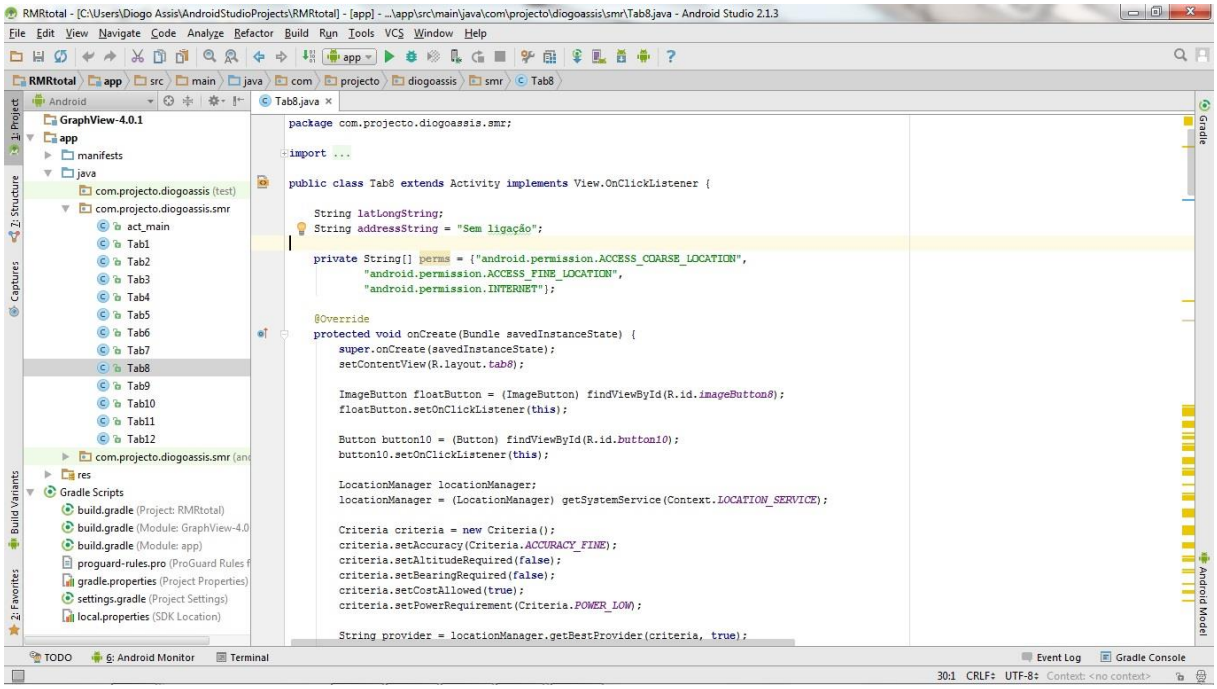


Fig. X. 1 – Código fonte da atividade *Tab8*

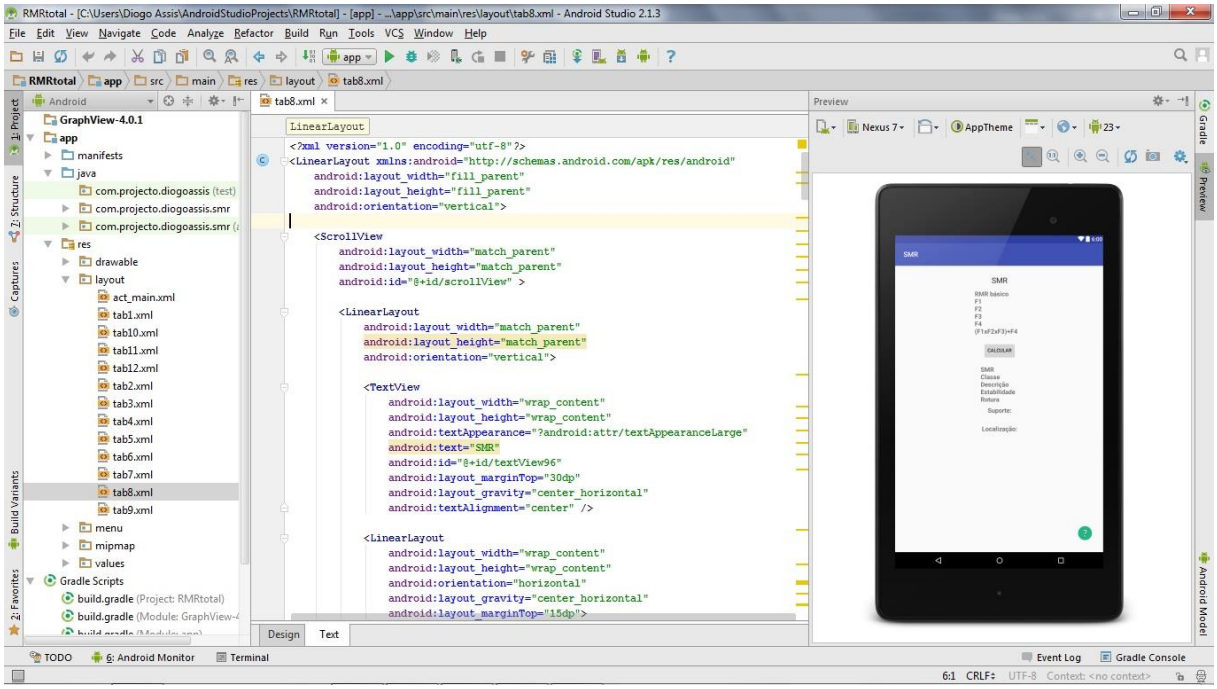


Fig. X. 2 – Código XML do *layout Tab8*

```
public class Tab8 extends Activity implements View.OnClickListener {

    String latLongString;
    String addressString = "Sem ligação";

    private String[] perms = {"android.permission.ACCESS_COARSE_LOCATION",
            "android.permission.ACCESS_FINE_LOCATION",
            "android.permission.INTERNET"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab8);

        ImageButton floatButton = (ImageButton)
findViewById(R.id.imageButton8);
        floatButton.setOnClickListener(this);

        Button button10 = (Button) findViewById(R.id.button10);
        button10.setOnClickListener(this);

        LocationManager locationManager;
        locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

        Criteria criteria = new Criteria();
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setAltitudeRequired(false);
        criteria.setBearingRequired(false);
        criteria.setCostAllowed(true);
        criteria.setPowerRequirement(Criteria.POWER_LOW);

        String provider = locationManager.getBestProvider(criteria, true);

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //    ActivityCompat#requestPermissions
            // here to request the missing permissions, and then overriding
            //   public void onRequestPermissionsResult(int requestCode,
String[] permissions,
            //                                          int[] grantResults)
            // to handle the case where the user grants the permission. See
the documentation
            // for ActivityCompat#requestPermissions for more details.
            return;
        }
        Location location = locationManager.getLastKnownLocation(provider);
        updateWithNewLocation(location);

        locationManager.requestLocationUpdates(provider, 2000, 10,
                locationListener);
    }

    private final LocationListener locationListener = new
```

```
LocationListener() {
        public void onLocationChanged(Location location) {
            updateWithNewLocation(location);
        }

        public void onProviderDisabled(String provider) {
            updateWithNewLocation(null);
        }

        public void onProviderEnabled(String provider) {
        }

        public void onStatusChanged(String provider, int status, Bundle
extras) {
        }
    };

    private void updateWithNewLocation(Location location) {

        //String latLongString;
        //String addressString = "No address found";

        if (location != null) {
            double lat = location.getLatitude();
            double lng = location.getLongitude();
            latLongString = "Lat:" + lat + "\nLong:" + lng;

            Geocoder gc = new Geocoder(this, Locale.getDefault());
            try {
                List<Address> addresses = gc.getFromLocation(lat, lng, 1);
                StringBuilder sb = new StringBuilder();
                if (addresses.size() > 0) {
                    Address address = addresses.get(0);

                    for (int i = 0; i < address.getMaxAddressLineIndex();
i++)
                        sb.append(address.getAddressLine(i)).append("\n");

                    //sb.append(address.getLocality()).append("\n");
                    //sb.append(address.getPostalCode()).append("\n");
                    sb.append(address.getCountryName());
                }
                addressString = sb.toString();
            } catch (IOException e) {
            }
        } else {
            latLongString = "Sem sinal GPS";
        }

    }


    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button10: {
                final TextView tv103 = (TextView)
findViewById(R.id.textView103);
                final TextView tv104 = (TextView)
```

```
findViewById(R.id.textView104);
                final TextView tv105 = (TextView)
findViewById(R.id.textView105);
                final TextView tv106 = (TextView)
findViewById(R.id.textView106);
                final TextView tv107 = (TextView)
findViewById(R.id.textView107);
                final TextView tv108 = (TextView)
findViewById(R.id.textView108);


                final TextView tv114 = (TextView)
findViewById(R.id.textView114);
                final TextView tv115 = (TextView)
findViewById(R.id.textView115);
                final TextView tv116 = (TextView)
findViewById(R.id.textView116);
                final TextView tv117 = (TextView)
findViewById(R.id.textView117);
                final TextView tv118 = (TextView)
findViewById(R.id.textView118);

                final TextView tv120 = (TextView)
findViewById(R.id.textView120);

                final TextView tv122 = (TextView)
findViewById(R.id.textView122);

                SharedPreferences rmrb = getSharedPreferences("rmrb", 0);
                String rmr = rmrb.getString("rmr", "0");

                SharedPreferences f1 = getSharedPreferences("f1", 0);
                String fumf = f1.getString("fumf", "0");

                SharedPreferences f2 = getSharedPreferences("f2", 0);
                String fdoisf = f2.getString("fdoisf", "0");

                SharedPreferences f3 = getSharedPreferences("f3", 0);
                String ftresf = f3.getString("ftresf", "0");

                SharedPreferences f4 = getSharedPreferences("f4", 0);
                String fquatrof = f4.getString("fquatrof", "0");

                double a = Double.parseDouble(rmr);
                double b = Double.parseDouble(fumf);
                double c = Double.parseDouble(fdoisf);
                double d = Double.parseDouble(ftresf);
                double e = Double.parseDouble(fquatrof);
                double f = (b * c * d) + e;
                double smr = a + f;


                tv103.setText(String.valueOf(String.format("%2.0f", a)));
                tv104.setText(String.valueOf(String.format("%2.3f", b)));
                tv105.setText(String.valueOf(String.format("%2.3f", c)));
                tv106.setText(String.valueOf(String.format("%2.3f", d)));
                tv107.setText(String.valueOf(String.format("%2.0f", e)));
                tv108.setText(String.valueOf(String.format("%2.3f", f)));
```

```
                if (smr <= 20) {
                    tv114.setText(String.valueOf(String.format("%2.0f",
smr)));
                    tv116.setText("Muito má");
                    tv117.setText("Completamente instável");
                    tv118.setText("Planar ou circular");
                }
                if (smr > 20 && smr <= 40) {
                    tv114.setText(String.valueOf(String.format("%2.0f",
smr)));
                    tv116.setText("Má");
                    tv117.setText("Instável");
                    tv118.setText("Planar ou em cunha");
                }
                if (smr > 40 && smr <= 60) {
                    tv114.setText(String.valueOf(String.format("%2.0f",
smr)));
                    tv116.setText("Normal");
                    tv117.setText("Parcialmente estável");
                    tv118.setText("Algumas juntas ou" +
                            "\nmuitas cunhas");
                }
                if (smr > 60 && smr <= 80) {
                    tv114.setText(String.valueOf(String.format("%2.0f",
smr)));
                    tv116.setText("Boa");
                    tv117.setText("Estável");
                    tv118.setText("Alguns blocos");
                }
                if (smr > 80 && smr <= 100) {
                    tv114.setText(String.valueOf(String.format("%2.0f",
smr)));
                    tv116.setText("Muito boa");
                    tv117.setText("Completamente estável");
                    tv118.setText("Nenhuma");
                }

                if (smr >= 10 && smr <= 20) {
                    tv115.setText("V a");
                    tv120.setText("Muro gravidade ou parede ancorada" +
                            "\nReescavação");
                }

                if (smr > 20 && smr <= 30) {
                    tv115.setText("IV b");
                    tv120.setText("Betão projetado armado" +
                            "\nMuro de suporte e/ou betão" +
                            "\nReescavação" +
                            "\nDrenagem profunda");
                }

                if (smr > 30 && smr <= 40) {
                    tv115.setText("IV a");
                    tv120.setText("Ancoragens" +
                            "\nBetão projetado" +
                            "\nMuro de suporte e/ou betão" +
                            "\nReescavação" +
                            "\nDrenagem");
                }
```

```
                if (smr > 40 && smr <= 50) {
                    tv115.setText("III b");
                    tv120.setText("Vala na base do talude e/ou redes
metálicas" +
                            "\nPregagens sistemáticas" +
                            "\nAncoragens" +
                            "\nBetão projetado" +
                            "\nMuro de suporte e/ou betão de
regularização");
                }

                if (smr > 50 && smr <= 60) {
                    tv115.setText("III a");
                    tv120.setText("Vala na base do talude e/ou redes
metálicas" +
                            "\nPregagens pontuais ou sistemáticas" +
                            "\nBetão projetado pontualmente");
                }

                if (smr > 60 && smr <= 70) {
                    tv115.setText("II b");
                    tv120.setText("Vala na base do talude e/ou vedação" +
                            "\nRedes metálicas" +
                            "\nPregagens pontuais ou sistemáticas");
                }

                if (smr > 70 && smr <= 80) {
                    tv115.setText("II a");
                    tv120.setText("Vala na base do talude e/ou vedação" +
                            "\nPregagens pontuais ou sistemáticas");
                }

                if (smr > 80 && smr <= 90) {
                    tv115.setText("I b");
                    tv120.setText("Não necessário");
                }

                if (smr > 90 && smr <= 100) {
                    tv115.setText("Ia");
                    tv120.setText("Não necessário");
                }


                tv122.setText(latLongString +
                        "\n" + addressString);

            }

            break;
            case R.id.imageButton8:{
                AlertDialog.Builder dlg  = new AlertDialog.Builder(this);
                dlg.setTitle("SMR");
                dlg.setMessage("Pressione o botão Calcular para
apresentação do índice SMR." +
                        "\nCaso o GPS e a Internet estejam ligados é
indicada a sua localização.");
                dlg.show();
            }
```

```
                    break;

                }
            }
        }
```

ANEXO XI
Direção talude – *Tab9*

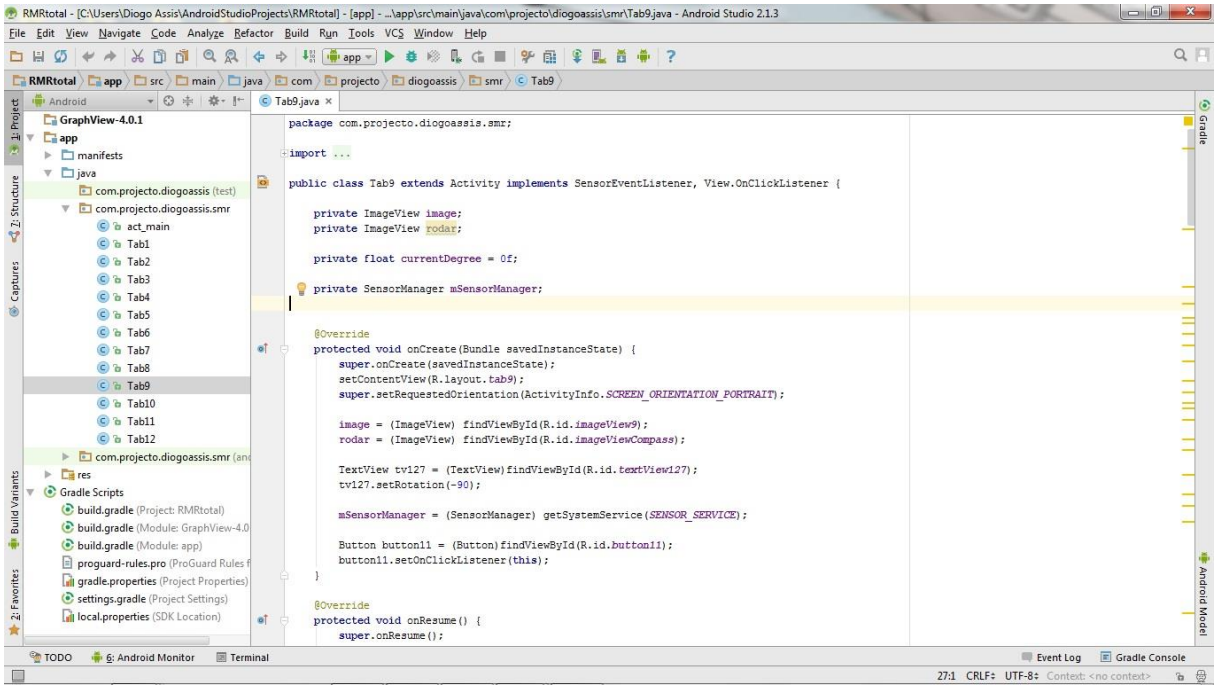


Fig. XI. 1 – Código fonte da atividade *Tab9*

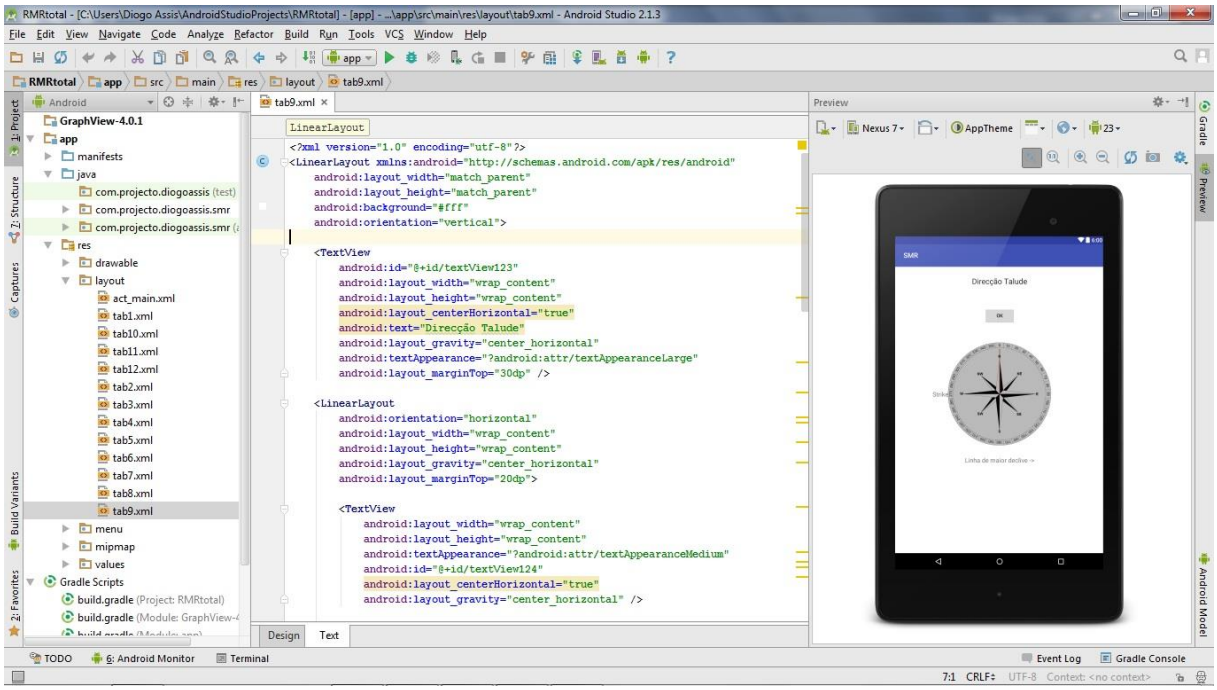


Fig. XI. 2 – Código XML do *layout Tab9*

```
public class Tab9 extends Activity implements SensorEventListener,
View.OnClickListener {

    private ImageView image;
    private ImageView rodar;

    private float currentDegree = 0f;

    private SensorManager mSensorManager;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab9);

super.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

        image = (ImageView) findViewById(R.id.imageView9);
        rodar = (ImageView) findViewById(R.id.imageViewCompass);

        TextView tv127 = (TextView)findViewById(R.id.textView127);
        tv127.setRotation(-90);

        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        Button button11 = (Button)findViewById(R.id.button11);
        button11.setOnClickListener(this);
    }

    @Override
    protected void onResume() {
        super.onResume();

        mSensorManager.registerListener(this,
mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
                SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause() {
        super.onPause();

        mSensorManager.unregisterListener(this);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {

        TextView tv125 = (TextView) findViewById(R.id.textView125);
        float degree = Math.round(event.values[0]);
        String grau1 = Float.toString(degree);
        double grau2 = Double.parseDouble(String.valueOf(grau1));

        if(grau2 == 0){
            String a = "NS";
            String b = "";
            String c = "E";
            tv125.setText(String.valueOf(a));
```

```
            SharedPreferences queda1 = getSharedPreferences("queda1", 0);
            SharedPreferences.Editor editor13 = queda1.edit();
            editor13.putString("queda1f", String.valueOf(c));
            editor13.commit();
        }
        if (grau2>0 && grau2<90){
            String a = "N";
            String b = "E";
            double g = grau2;
            String c = "SE";

tv125.setText(String.valueOf(a)+String.valueOf(String.format("%2.0f",g))+St
ring.valueOf(b));
            SharedPreferences queda1 = getSharedPreferences("queda1", 0);
            SharedPreferences.Editor editor13 = queda1.edit();
            editor13.putString("queda1f", String.valueOf(c));
            editor13.commit();
        }
        if(grau2 == 90){
            String a = "EW";
            String b = "";
            String c = "S";
            tv125.setText(String.valueOf(a));
            SharedPreferences queda1 = getSharedPreferences("queda1", 0);
            SharedPreferences.Editor editor13 = queda1.edit();
            editor13.putString("queda1f", String.valueOf(c));
            editor13.commit();
        }
        if (grau2>90 && grau2<180){
            String a = "N";
            String b = "W";
            double g = 180-grau2;
            String c = "SW";

tv125.setText(String.valueOf(a)+String.valueOf(String.format("%2.0f",g))+St
ring.valueOf(b));
            SharedPreferences queda1 = getSharedPreferences("queda1", 0);
            SharedPreferences.Editor editor13 = queda1.edit();
            editor13.putString("queda1f", String.valueOf(c));
            editor13.commit();
        }
        if(grau2 == 180){
            String a = "NS";
            String b = "";
            String c = "W";
            tv125.setText(String.valueOf(a));SharedPreferences grau =
getSharedPreferences("grau", 0);
            SharedPreferences queda1 = getSharedPreferences("queda1", 0);
            SharedPreferences.Editor editor13 = queda1.edit();
            editor13.putString("queda1f", String.valueOf(c));
            editor13.commit();
        }
        if (grau2>180 && grau2<270){
            String a = "N";
            String b = "E";
            double g = Math.abs(180-grau2);
            String c = "NW";

tv125.setText(String.valueOf(a)+String.valueOf(String.format("%2.0f",g))+St
```

```
ring.valueOf(b));
            SharedPreferences queda1 = getSharedPreferences("queda1", 0);
            SharedPreferences.Editor editor13 = queda1.edit();
            editor13.putString("queda1f", String.valueOf(c));
            editor13.commit();
        }
        if(grau2 == 270){
            String a = "EW";
            String c = "N";
            tv125.setText(String.valueOf(a));
            SharedPreferences queda1 = getSharedPreferences("queda1", 0);
            SharedPreferences.Editor editor13 = queda1.edit();
            editor13.putString("queda1f", String.valueOf(c));
            editor13.commit();
        }
        if (grau2>270 && grau2<360){
            String a = "N";
            String b = "W";
            double g = 360 - grau2;
            String c = "NE";

tv125.setText(String.valueOf(a)+String.valueOf(String.format("%2.0f",g))+St
ring.valueOf(b));
            SharedPreferences queda1 = getSharedPreferences("queda1", 0);
            SharedPreferences.Editor editor13 = queda1.edit();
            editor13.putString("queda1f", String.valueOf(c));
            editor13.commit();
        }

        SharedPreferences grau10 = getSharedPreferences("grau10", 0);
        SharedPreferences.Editor editor12 = grau10.edit();
        editor12.putString("grau1f", String.valueOf(grau2));
        editor12.commit();

        RotateAnimation ra = new RotateAnimation(
                currentDegree,
                -degree,
                Animation.RELATIVE_TO_SELF, 0.5f,
                Animation.RELATIVE_TO_SELF,
                0.5f);


        ra.setDuration(210);


        ra.setFillAfter(true);


        image.startAnimation(ra);
        currentDegree = -degree;

    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    @Override
```

```
    public void onClick(View v) {
        TextView tv125 = (TextView) findViewById(R.id.textView125);

        SharedPreferences az1 = getSharedPreferences("az1", 0);
        SharedPreferences.Editor editor11 = az1.edit();
        editor11.putString("az1f", tv125.getText().toString());
        editor11.commit();

        Intent i = new Intent(getApplicationContext(), Tab10.class);
        startActivity(i);
        finish();
    }
}
```

ANEXO XII
Inclinação talude – *Tab10*

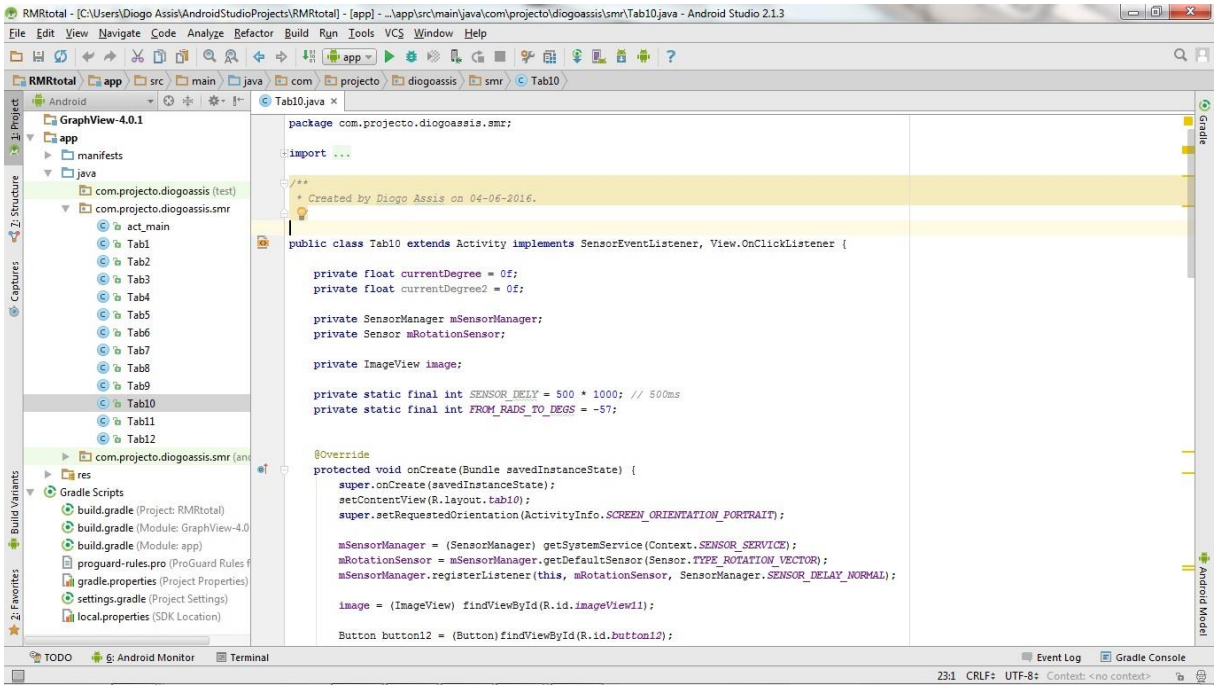


Fig. XII. 1 – Código fonte da atividade *Tab10*

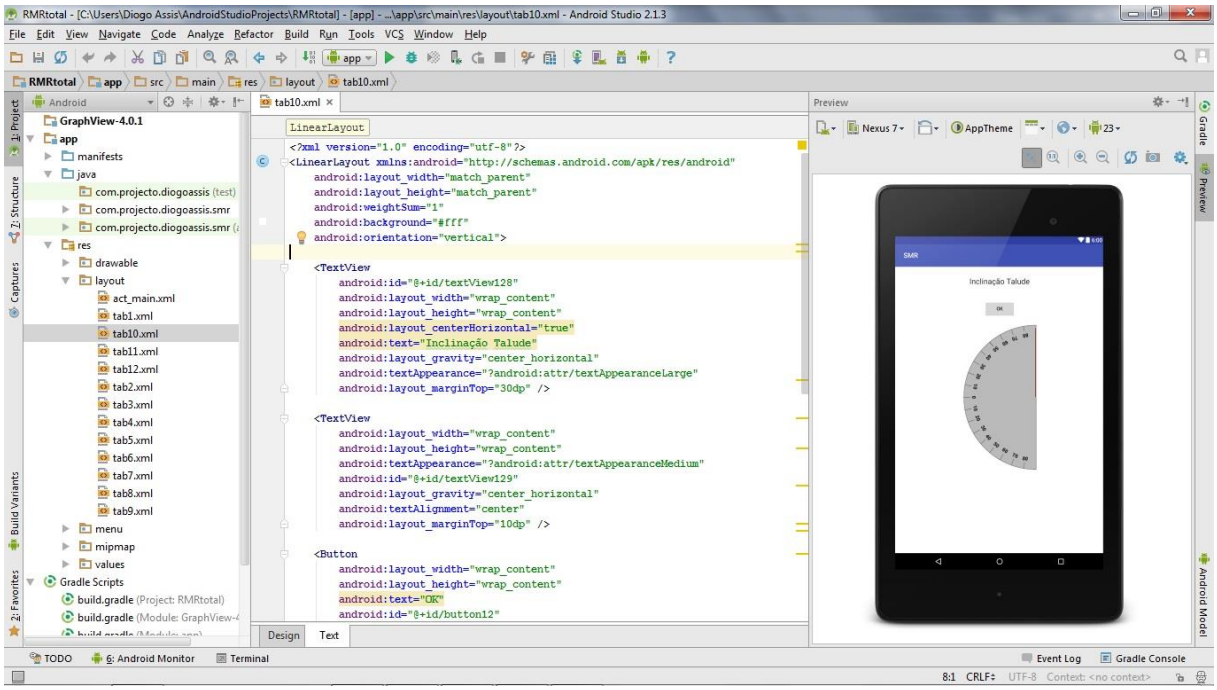


Fig. XII. 2 – Código XML do layout Tab10

```
public class Tab10 extends Activity implements SensorEventListener,
View.OnClickListener {

    private float currentDegree = 0f;
    private float currentDegree2 = 0f;

    private SensorManager mSensorManager;
    private Sensor mRotationSensor;

    private ImageView image;

    private static final int SENSOR_DELY = 500 * 1000; // 500ms
    private static final int FROM_RADS_TO_DEGS = -57;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab10);

super.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

        mSensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
        mRotationSensor =
mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);
        mSensorManager.registerListener(this, mRotationSensor,
SensorManager.SENSOR_DELAY_NORMAL);

        image = (ImageView) findViewById(R.id.imageView11);

        Button button12 = (Button)findViewById(R.id.button12);
        button12.setOnClickListener(this);

    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }


    @Override
    protected void onPause() {
        super.onPause();

        mSensorManager.unregisterListener(this);
    }


    @Override
    public void onSensorChanged(SensorEvent event) {
        if (event.sensor == mRotationSensor) {
            if (event.values.length > 4) {
                float[] truncatedRotationVector = new float[4];
                System.arraycopy(event.values, 0, truncatedRotationVector,
0, 4);
                update(truncatedRotationVector);
            } else {
```

```
                update(event.values);
            }
        }

    }

    private void update(float[] vectors) {
        float[] rotationMatrix = new float[9];
        SensorManager.getRotationMatrixFromVector(rotationMatrix, vectors);
        int worldAxisX = SensorManager.AXIS_X;
        int worldAxisZ = SensorManager.AXIS_Z;
        float[] adjustedRotationMatrix = new float[9];
        SensorManager.remapCoordinateSystem(rotationMatrix, worldAxisX,
worldAxisZ, adjustedRotationMatrix);
        float[] orientation = new float[3];
        SensorManager.getOrientation(adjustedRotationMatrix, orientation);
        float pitch = orientation[1] * FROM_RADS_TO_DEGS;
        float roll = orientation[2] * FROM_RADS_TO_DEGS;
        double roll2 = Math.abs(roll - 90);

        TextView tv129 = (TextView) findViewById(R.id.textView129);
        tv129.setText(String.valueOf(String.format("%2.0f", roll2)) + "
°");


        RotateAnimation ra = new RotateAnimation(
                currentDegree,
                currentDegree,
                Animation.RELATIVE_TO_SELF, 0.5f,
                Animation.RELATIVE_TO_SELF,
                1f);

        // how long the animation will take place
        ra.setDuration(0);

        // set the animation after the end of the reservation status
        ra.setFillAfter(true);

        image.startAnimation(ra);
        currentDegree = -(180-Math.abs(roll));


        SharedPreferences pendor1 = getSharedPreferences("pendor1", 0);
        SharedPreferences.Editor editor14 = pendor1.edit();
        editor14.putString("pendor1f",
String.valueOf(String.format("%2.0f", roll2)));
        editor14.commit();
    }


    @Override
     public void onClick(View v) {
        Intent i = new Intent(getApplicationContext(), Tab11.class);
        startActivity(i);
        finish();
    }

}
```

ANEXO XIII
Strike descontinuidade – *Tab11*

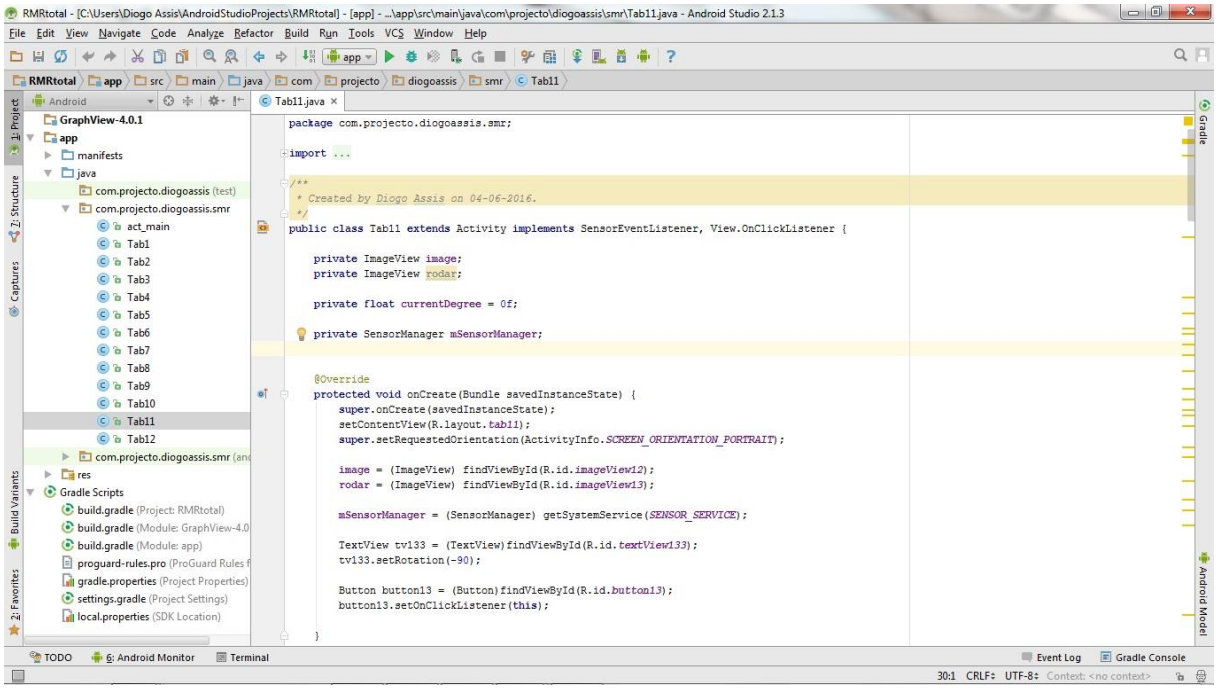


Fig. XIII. 1 – Código fonte da atividade *Tab11*

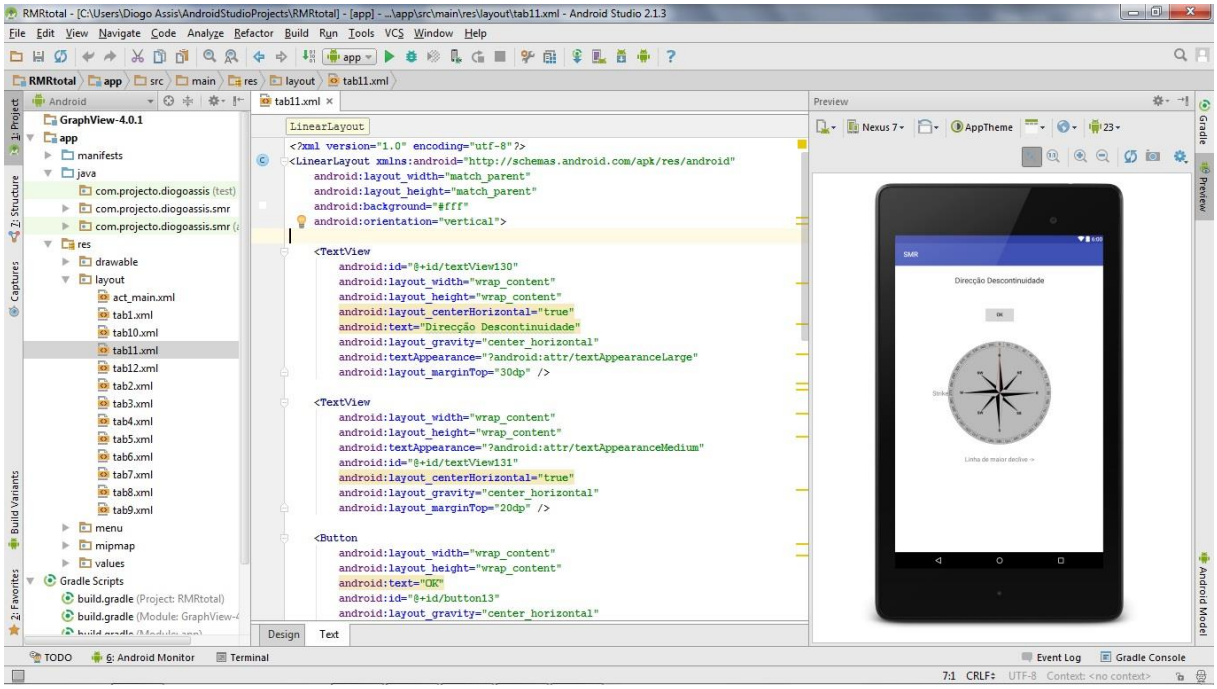


Fig. XIII. 2 – Código XML do *layout Tab11*

```
public class Tab11 extends Activity implements SensorEventListener,
View.OnClickListener {

    private ImageView image;
    private ImageView rodar;

    private float currentDegree = 0f;

    private SensorManager mSensorManager;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab11);

super.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

        image = (ImageView) findViewById(R.id.imageView12);
        rodar = (ImageView) findViewById(R.id.imageView13);

        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        TextView tv133 = (TextView)findViewById(R.id.textView133);
        tv133.setRotation(-90);

        Button button13 = (Button)findViewById(R.id.button13);
        button13.setOnClickListener(this);

    }

    @Override
    protected void onResume() {
        super.onResume();

        // for the system's orientation sensor registered listeners
        mSensorManager.registerListener(this,
mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
                SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause() {
        super.onPause();

        // to stop the listener and save battery
        mSensorManager.unregisterListener(this);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {

        TextView tv131 = (TextView) findViewById(R.id.textView131);
        // get the angle around the z-axis rotated
        float degree = Math.round(event.values[0]);
        String grau1 = Float.toString(degree);
        double grau2 = Double.parseDouble(String.valueOf(grau1));

        if(grau2 == 0){
```

```
            String a = "NS";
            String b = "";
            String c = "E";
            tv131.setText(String.valueOf(a));
            SharedPreferences queda2 = getSharedPreferences("queda2", 0);
            SharedPreferences.Editor editor17 = queda2.edit();
            editor17.putString("queda2f", String.valueOf(c));
            editor17.commit();

        }
        if (grau2>0 && grau2<90){
            String a = "N";
            String b = "E";
            double g = grau2;
            String c = "SE";

tv131.setText(String.valueOf(a)+String.valueOf(String.format("%2.0f",g))+St
ring.valueOf(b));
            SharedPreferences queda2 = getSharedPreferences("queda2", 0);
            SharedPreferences.Editor editor17 = queda2.edit();
            editor17.putString("queda2f", String.valueOf(c));
            editor17.commit();
        }
        if(grau2 == 90){
            String a = "EW";
            String b = "";
            String c = "S";
            tv131.setText(String.valueOf(a));
            SharedPreferences queda2 = getSharedPreferences("queda2", 0);
            SharedPreferences.Editor editor17 = queda2.edit();
            editor17.putString("queda2f", String.valueOf(c));
            editor17.commit();
        }
        if (grau2>90 && grau2<180){
            String a = "N";
            String b = "W";
            double g = 180-grau2;
            String c = "SW";

tv131.setText(String.valueOf(a)+String.valueOf(String.format("%2.0f",g))+St
ring.valueOf(b));
            SharedPreferences queda2 = getSharedPreferences("queda2", 0);
            SharedPreferences.Editor editor17 = queda2.edit();
            editor17.putString("queda2f", String.valueOf(c));
            editor17.commit();
        }
        if(grau2 == 180){
            String a = "NS";
            String b = "";
            String c = "W";
            tv131.setText(String.valueOf(a));SharedPreferences grau =
getSharedPreferences("grau", 0);
            SharedPreferences queda2 = getSharedPreferences("queda2", 0);
            SharedPreferences.Editor editor17 = queda2.edit();
            editor17.putString("queda2f", String.valueOf(c));
            editor17.commit();

        }
        if (grau2>180 && grau2<270){
```

```
            String a = "N";
            String b = "E";
            double g = Math.abs(180-grau2);
            String c = "NW";

tv131.setText(String.valueOf(a)+String.valueOf(String.format("%2.0f",g))+St
ring.valueOf(b));
            SharedPreferences queda2 = getSharedPreferences("queda2", 0);
            SharedPreferences.Editor editor17 = queda2.edit();
            editor17.putString("queda2f", String.valueOf(c));
            editor17.commit();
        }
        if(grau2 == 270){
            String a = "EW";
            String c = "N";
            tv131.setText(String.valueOf(a));
            SharedPreferences queda2 = getSharedPreferences("queda2", 0);
            SharedPreferences.Editor editor17 = queda2.edit();
            editor17.putString("queda2f", String.valueOf(c));
            editor17.commit();
        }
        if (grau2>270 && grau2<360){
            String a = "N";
            String b = "W";
            double g = 360 - grau2;
            String c = "NE";

tv131.setText(String.valueOf(a)+String.valueOf(String.format("%2.0f",g))+St
ring.valueOf(b));
            SharedPreferences queda2 = getSharedPreferences("queda2", 0);
            SharedPreferences.Editor editor17 = queda2.edit();
            editor17.putString("queda2f", String.valueOf(c));
            editor17.commit();
        }

        SharedPreferences grau20 = getSharedPreferences("grau20", 0);
        SharedPreferences.Editor editor16 = grau20.edit();
        editor16.putString("grau2f", String.valueOf(grau2));
        editor16.commit();

        // create a rotation animation (reverse turn degree degrees)
        RotateAnimation ra = new RotateAnimation(
                currentDegree,
                -degree,
                Animation.RELATIVE_TO_SELF, 0.5f,
                Animation.RELATIVE_TO_SELF,
                0.5f);

        // how long the animation will take place
        ra.setDuration(210);

        // set the animation after the end of the reservation status
        ra.setFillAfter(true);

        // Start the animation
        image.startAnimation(ra);
        currentDegree = -degree;

    }
```

```
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // not in use
    }

    @Override
    public void onClick(View v) {
        TextView tv131 = (TextView) findViewById(R.id.textView131);

        SharedPreferences az2 = getSharedPreferences("az2", 0);
        SharedPreferences.Editor editor15 = az2.edit();
        editor15.putString("az2f", tv131.getText().toString());
        editor15.commit();
        Intent i = new Intent(getApplicationContext(), Tab12.class);
        startActivity(i);
        finish();
    }
}
```

ANEXO XIV
Inclinação descontinuidade – *Tab12*

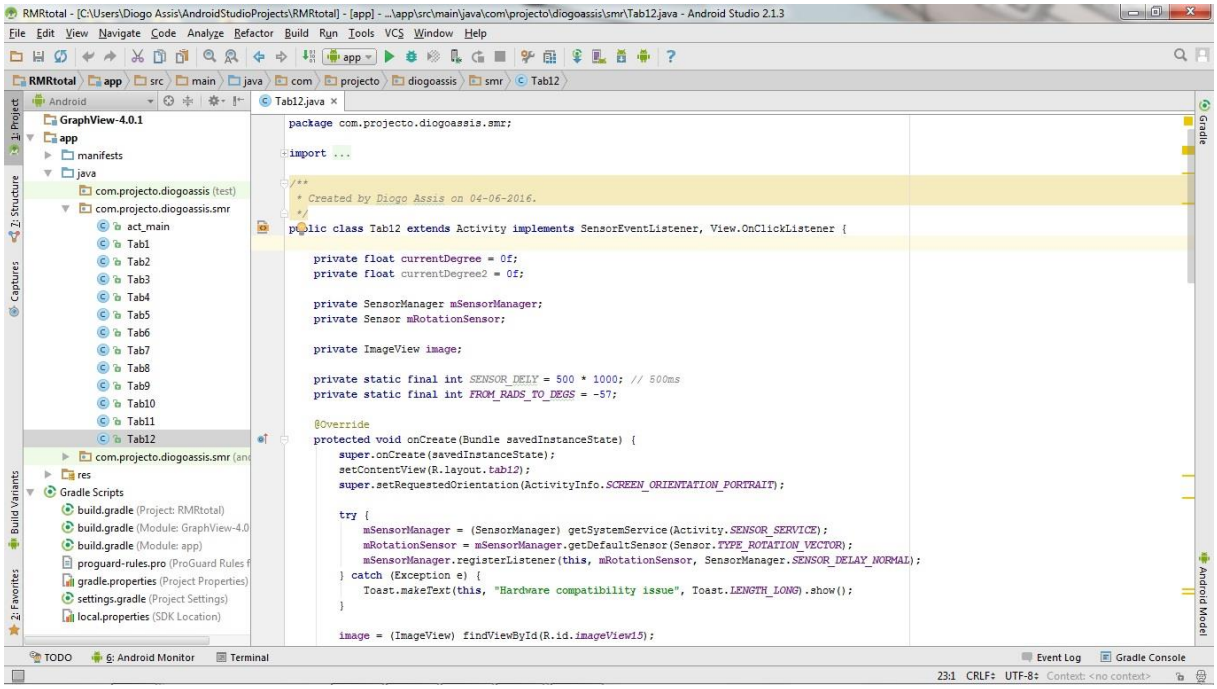


Fig. XIV. 1 – Código fonte da atividade *Tab12*

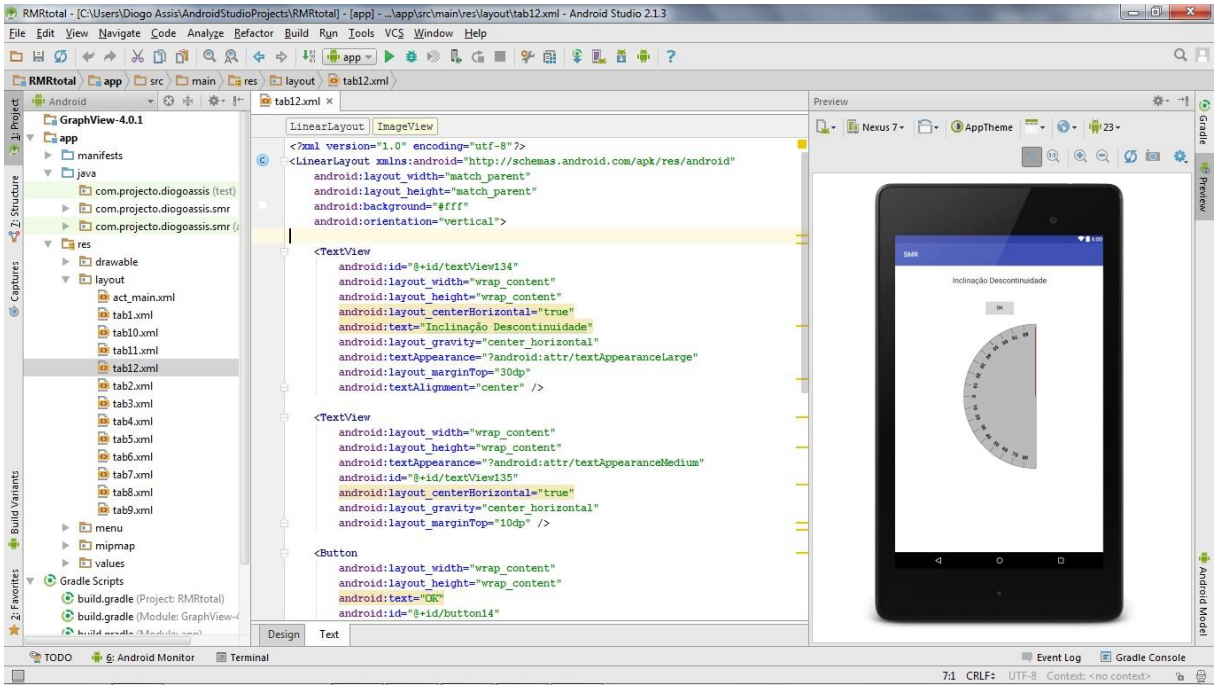


Fig. XIV. 2 – Código XML do *layout Tab12*

```
public class Tab12 extends Activity implements SensorEventListener,
View.OnClickListener {

    private float currentDegree = 0f;
    private float currentDegree2 = 0f;

    private SensorManager mSensorManager;
    private Sensor mRotationSensor;

    private ImageView image;

    private static final int SENSOR_DELY = 500 * 1000; // 500ms
    private static final int FROM_RADS_TO_DEGS = -57;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tab12);

super.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

        try {
            mSensorManager = (SensorManager)
getSystemService(Activity.SENSOR_SERVICE);
            mRotationSensor =
mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);
            mSensorManager.registerListener(this, mRotationSensor,
SensorManager.SENSOR_DELAY_NORMAL);
        } catch (Exception e) {
            Toast.makeText(this, "Hardware compatibility issue",
Toast.LENGTH_LONG).show();
        }

        image = (ImageView) findViewById(R.id.imageView15);

        Button button14 = (Button)findViewById(R.id.button14);
        button14.setOnClickListener(this);

    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }


    @Override
    protected void onPause() {
        super.onPause();

        mSensorManager.unregisterListener(this);

    }


    @Override
    public void onSensorChanged(SensorEvent event) {
        if (event.sensor == mRotationSensor) {
            if (event.values.length > 4) {
```

```
                float[] truncatedRotationVector = new float[4];
                System.arraycopy(event.values, 0, truncatedRotationVector,
0, 4);
                update(truncatedRotationVector);
            } else {
                update(event.values);
            }
        }

    }

    private void update(float[] vectors) {
        float[] rotationMatrix = new float[9];
        SensorManager.getRotationMatrixFromVector(rotationMatrix, vectors);
        int worldAxisX = SensorManager.AXIS_X;
        int worldAxisZ = SensorManager.AXIS_Z;
        float[] adjustedRotationMatrix = new float[9];
        SensorManager.remapCoordinateSystem(rotationMatrix, worldAxisX,
worldAxisZ, adjustedRotationMatrix);
        float[] orientation = new float[3];
        SensorManager.getOrientation(adjustedRotationMatrix, orientation);
        float pitch = orientation[1] * FROM_RADS_TO_DEGS;
        float roll = orientation[2] * FROM_RADS_TO_DEGS;
        double roll2 = Math.abs(roll - 90);

        TextView tv135 = (TextView) findViewById(R.id.textView135);
        tv135.setText(String.valueOf(String.format("%2.0f", roll2)) + "
°");


        RotateAnimation ra = new RotateAnimation(
                currentDegree,
                currentDegree,
                Animation.RELATIVE_TO_SELF, 0.5f,
                Animation.RELATIVE_TO_SELF,
                1f);

        // how long the animation will take place
        ra.setDuration(0);

        // set the animation after the end of the reservation status
        ra.setFillAfter(true);

        image.startAnimation(ra);
        currentDegree = -(180-Math.abs(roll));

        SharedPreferences pendor2 = getSharedPreferences("pendor2", 0);
        SharedPreferences.Editor editor18 = pendor2.edit();
        editor18.putString("pendor2f",
String.valueOf(String.format("%2.0f", roll2)));
        editor18.commit();
    }


    @Override
    public void onClick(View v) {
        finish();
    }
}
```