HANDSPY

HandSpy: Melhoramentos a um Sistema de Gestão de Experiências sobre Processos Cognitivos da Escrita

Hugo Miguel Pessegueiro de Sousa
Mestrado integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2013

Orientador
Prof. Dr. José Paulo Leal – DCC-FCUP

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Agradecimentos

Quero agradecer, desde já, a todas as pessoas que contribuíram para o desenvolvimento deste projeto.

Um especial agradecimento ao Professor José Paulo Leal pela orientação e tempo disponibilizado durante o projeto.

Agradeço também aos membro da equipa do projeto Daar, Rui Alves, Teresa Limpo, Teresa Rato e Joana Carvalho, pelo feedback e paciência durante o desenvolvimento deste projeto.

Um agradecimento especial a minha namorada Marta Silva, e aos meus amigos em especial ao Armindo Pereira, Tiago Melo, José Pedro, Nino Rocha, Pedro Bragança, André Costa, André Gaspar, André Francisco, João Sampaio, Filipe Boaventura e André Ruela.

Por fim, agradeço o apoio incondicional da minha família em especial aos meus pais, aos meus tios, Nuno Pessegueiro, José Pessegueiro, Luís Pessegueiro, Carmo Pessegueiro e a minha Avó Otília Costa, por acreditarem sempre em mim e naquilo que faço. A eles, dedico todo este trabalho.

Resumo

As experiências científicas geram frequentemente uma grande quantidade de dados. Em particular as experiências realizadas no âmbito das ciências humanas requerem a análise e validação de imensos dados provenientes dos participantes nessas experiências. Para auxiliar os cientistas existem sistemas para gerir não só os dados recolhidos, mas também os dados de análise das próprias experiências.

O HandSpy é um sistema que implementa um ambiente colaborativo para gestão de experiências no estudo dos processos cognitivos da escrita, concebido para cobrir todas as fases do processo de experimentação, desde a definição das tarefas a ser realizadas pelos participantes, até à geração de resultados. Este sistema de gestão colaborativo de experiências é constituído por um interface gráfico na Web e ligado a uma base de dados online. Este sistema foi implementado para armazenar ficheiros no formato InkML, um formato de dados XML para representar tinta digital, possibilitando que a recolha dos dados não tenha que ser feita por um equipamento específico.

Esta dissertação descreve o desenvolvimento de uma nova versão do HandSpy, com o objetivo de incorporar ferramentas de visualização dos dados em formato vídeo, melhorar aspetos de eficiência e design do interface. Para implementar esta nova versão foram usadas ferramentas como o GWT, para criação do um interface mais apelativo, o XQuery para aumentar a performance de acesso aos dados e a biblioteca FFMPEG para a geração de vídeos a partir dos dados. A nova versão do HandSpy já se encontra em pleno funcionamento e tem tido uma reação positiva por parte dos investigadores que a utilizam.

Abstract

The scientific experiments, carried out under the cognitive processes of writing, generate a large quantity of information for validating, from different participants used in these experiments. In order to assist the scientists to process and analyze all these data some attempts were made to create systems to manage not only the data generated, but also information about their own experiences.

The HandSpy 1.0 is a system that implements a collaborative environment for managing experiments in the study of the cognitive processes of writing, designed to cover all the stages of the trial, from the definition of the tasks to be performed by the participants until the development of the results. This management collaborative system is constituted by experiences for a graphic web interface and connected to an online database. This system was implemented to store the file format InkML, which is an XML data format to represent digital ink allowing that the collection of data is not made by a specific equipment.

This thesis describes the development of a new HandSpy version who came up with the objective of incorporating new data exposition tools (for example the video) to simplify the analysis, to improve the efficiency and the interface design. To implement this new version were used methodologies, like the GWT, to create a more appellative interface, the XQuery to improve the performance of data access and other structural modifications. Nowadays this new HandSpy version is fully functioning and has a great feedback from the investigators who use it.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A recolha de dados é uma parte essencial da investigação cientifica. Porém, nas últimas décadas verificou-se um crescimento exponencial na obtenção de dados devido aos avanços dos meios tecnológicos. Estes avanços motivaram os investigadores das mais diversas áreas científicas a usarem dispositivos digitais (não só computadores) nos seus estudos, tendo surgindo hardware de recolha de dados em vários formatos e também software capaz de os analisar. Uma das principais preocupações dos programadores que criam este tipo de software, é torná-lo de tal forma intuitivo que qualquer investigador consiga realizar o seu trabalho.

Os processos cognitivos da escrita, por exemplo, envolve imensos participantes e gera um grande volume de dados. Este tipo de experiências para serem realizadas necessitam de dois componentes fundamentais: um hardware de gravação e escrita (como por exemplo uma caneta digital ou uma mesa digitalizadora) e uma plataforma de tratamento dos dados para que sejam analisados pelos investigadores. Na secção seguinte, serão abordado em pormenor esses processos e a sua importância.

1.1 Processos Cognitivos da Escrita

Saber escrever é uma competência fundamental para que se tenha sucesso, não só a nível profissional ou académico mas também socialmente. Dada a importância desta atividade, os processos cognitivos que se desenrolam durante a sua execução têm

uma elevada importância para os investigadores que a estudam.

Graças à evolução tecnológica, a investigação dos processos cognitivos de escrita tem vindo a ter um grande desenvolvimento. Esta área de investigação processa um volume de dados com alguma dimensão, logo a necessidade de usar meios tecnológicos para os armazenar e posteriormente analisar é uma peça fundamental para o sucesso da investigação.

O projeto DAAR – Desenvolver, Automatizar e Autorregular os Processos Cognitivos na Composição Escrita – é um projeto de investigação científica que estuda processos cognitivos de escrita na Faculdade de Psicologia e Ciências da Educação da Universidade do Porto.
As experiências no âmbito desse projeto são divididas em duas partes:

1. Recolha de dados através de canetas digitais, sendo os participantes alunos de escolas do primeiro ciclo e segundo de ensino.
2. Análise destes dados na plataforma Web, depois de carregados para o repositório, por investigadores do projeto.

O HandSpy é a plataforma Web usada pelos investigadores do projeto DAAR para estudo dos textos redigidos pelos participantes. Através dos resultados fornecidos por esta aplicação, os investigadores podem usar novos métodos e estratégias para o ensino da escrita nas escolas. Para se entender melhor como esses resultados são obtidos é necessário entender alguns conceitos e mecanismos de recolha do HandSpy.

1.2 Mecanismos de Recolha

1.2.1 Smartpen

Como já foi anteriormente referido, o mecanismo de recolha usado pelo HandSpy é a caneta digital. A escolha recaiu neste tipo de dispositivo devido ao seu baixo custo (rondando o seu preço entre 50 e 100 euros), permitindo adquirir um maior número de dispositivos e realizar várias recolhas em simultâneo. A recolha pode ser feita nas próprias escolas, já que este equipamento é fácil de transportar devido às suas pequenas dimensões. Dado ter um formato muito semelhante a uma caneta

convencional, como se pode observar na Figura 1.1, não se torna um fator de distração para crianças, como acontecia anteriormente com as mesas digitalizadoras.

Figura 1.1: Livescribe Smartpen

As *smartpens* utilizadas na recolha são da marca Livescribe e têm a possibilidade de registar quer traços quer voz. Uma *smartpen* deste tipo lê a posição que se encontra através de uma câmara de infravermelhos e consegue situar-se numa folha de papel apropriado como se de um GPS se tratasse. Além disso é possível programar a *smartpen* por via de um pequeno programa em Java Micro Edition.

As principais limitações são a utilização de um papel especial para recolha dos traços, alguns erros de gravação e o não reconhecimento de carateres.

1.2.2 Aplicação de Papel

Durante a recolha cada *smartpen* Liverscribe está associada a uma aplicação de papel especificamente criada para a execução da experiência. A aplicação de papel é um caderno com uma descrição digital em cada folha num ficheiro AFD. Este tipo de aplicação é desenvolvido no eclipse IDE através de um *plugin* fornecido pela Livers-cribe. O *plugin* permite criar um documento AFD, com regiões ativas que contêm uma imagem de fundo em formato PostScript, com um cabeçalho e botões de ação. O *layout* usado tem três regiões ativas: a região superior onde é colocado o cabeçalho, o botão de início e o botão de fim. Estas regiões podem utilizar uma imagem de fundo de modo a garantir a sua exata posição.

Na Figura 1.2 está representada a aplicação de papel com as regiões ativas em des-

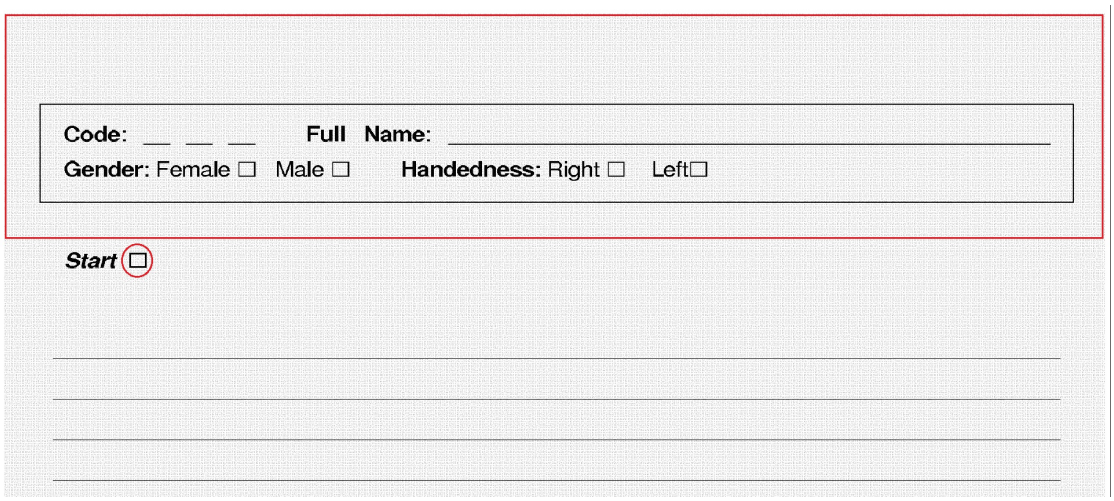

Figura 1.2: Aplicação de papel - Regiões ativas em vermelho

taque. Depois do participante iniciar a sua tarefa todos esses dados gerados são gravados num ficheiro InkML. Na subsecção seguinte é detalhado o conteúdo de um ficheiro InkML e o tipo de informação nele registadas.

1.2.3 Geração dos Ficheiros de Dados

Depois de recolhidos, os dados são extraídos da *smartpen* e convertidos no formato InkML. Cada ficheiro InkML é constituído por diversos elementos <*trace*>, que contém as coordenadas X e Y num determinado `timestamp`, em milissegundos, sendo incrementado imediatamente após o participante tocar com a ponta da caneta na folha de papel.

```
...
<trace>
 2534 685 37297520816, 2537 684 37297520829, 2539 681 37297520843,
 2544 678 37297520856, 2546 677 37297520883, 2546 678 37297520896,
 2548 680 37297520909, 2549 682 37297520923, 2554 695 37297520949,
 2559 707 37297520963, 2565 720 37297520976, 2572 732 37297520989,
 2580 749 37297521016, 2582 753 37297521029, 2584 754 37297521043,
 2584 754 37297521056, 2584 754 37297521083, 2584 754 37297521096,
 2584 754 37297521109, 2584 754 37297521123, 2584 753 37297521149,
 2585 751 37297521163, 2586 748 37297521176, 2592 738 37297521189,
 2603 710 37297521216, 2611 694 37297521229, 2615 684 37297521243,
 2620 673 37297521256, 2620 670 37297521283, 2621 670 37297521296
</trace>
...
```

Tabela 1.1: Exemplo de um elemento trace InkML

A Listagem 1.1 é um exemplo de um elemento $<$*trace*$>$gerado com as informações de cada ponto, seguindo o esquema [X, Y, Timestamp] separados por vírgulas. Estes dados foram gerados a partir de uma recolha feita com a *smartpen*. A informação usada para criar os ficheiros InkML é obtida através dos documentos AFD armazenados na memória interna do dispositivo.

1.3 Conceito Básicos

Para realizar experiências sobre os processos cognitivos da escrita foram definidos alguns conceitos. Na lista seguinte estão indicados os principais conceitos utilizados durante o processo de análise dos dados na plataforma Web do HandSpy.

Protocolo O protocolo é constituído por meta dados (o cabeçalho) e um conjunto de traços, podendo ser apresentados num formato multimédia como vídeo, imagem ou então numa tabela de valores. É único para cada participante numa determinada tarefa.

Tarefa (*Task*) Os textos registados nos protocolos podem ser agrupados em diferentes tarefas, sendo as mais comuns, cópia, alfabeto, narrativa, ditado e opinião. Isto permite que se possa analisar o efeito de uma tarefa numa determinada produção de texto.

Participante Os protocolos produzidos representam os textos concebidos pelos participantes. A suas características pessoais, como sexo, idade, grau de escolaridade ou língua materna, podem afetar a produção da escrita e são objeto de análise nas experiências.

Jorro (*Burst*) A produção de texto ocorre em jorro, isto é, quando produzimos um determinado texto, ele não acontece linearmente no tempo. Através dos dados recolhidos pelos dispositivos digitais é possível perceber quanto tempo um participante escreve continuamente.

Pausa (*Pause*) Um jorro de escrita e terminado por uma pausa. A pausa é o intervalo de tempo em que o participante parou de escrever até começar outro *burst*.

Limite (*Threshold*) é o tempo mínimo de pausa entre 2 *bursts* consecutivos. No

âmbito do projeto DAAR esse valor é habitualmente de 2 segundos, mas são feitas análises com valores diferentes.

Os dois conceitos *burst* e *pause* são os fundamentais, a partir dos quais são calculados outros como a media de *bursts* ou duração de pauses, o número médio de palavras por *burst* ou então em que localização foi iniciada uma *pause* e o seu fim.

1.4 Evolução do HandSpy

Em 2012 foi lançada a versão 1.0 da plataforma web do HandSpy, no âmbito do projeto DAAR, com o propósito de auxiliar o investigadores na gestão dos dados recolhidos e a sua utilização colaborativa. Esta versão foi concebida como um sistema Web e baseia-se numa base de dados XML para armazenamento dos dados recolhidos a partir de *smartpens*.

A utilização do HandSpy 1.0 está dividida em diferentes fases com uma ordem cronológica. Inicialmente os investigador carrega os dados recolhidos. De seguida, a esses dados são atribuídas as tarefas feitas pelos participantes e finalmente são analisados individualmente. Para facilitar o processo de análise o HandSpy usa um mecanismo de filtragem por tarefa. Esta seleção também pode ser feita através de critérios sobre o participante. Por exemplo, podem ser selecionados os participantes que escrevam com a mão direita ou então participantes do sexo masculino ou ambos em simultâneo. Esta plataforma Web foi implementada usando a linguagem de programação Javascript, muito utilizada para criação de aplicações Web.

Apesar da versão 1.0 do HandSpy responder as necessidades básicas dos seus utilizadores, houve a necessidade de acrescentar novas funcionalidades de visualização dos dados, assim como melhorar o design de modo a torná-la mais intuitiva e aumentar a eficiência de certas operações.

Os principais aspetos de inovação da nova versão do HandSpy são:

- A utilização de uma nova tecnologia na implementação ao nível do cliente, o GWT, em que a codificação da interface Web é feita em Java, permitindo uma melhor integração com o servidor da aplicação, que também é desenvolvido nessa linguagem.

- A introdução de métodos de visualização dos dados de uma forma temporizada, usando vídeos gerados a partir dos dados recolhidos.
- a modificação nas consultas ao repositório de dados, através do uso de uma linguagem de interrogação em bases de dados XML, o XQuery.
- A criação um design mais adequado às necessidades e dificuldades dos utilizadores.

Estas modificações deram então origem a uma nova versão, o HandSpy 2.0. Nos capítulos seguintes são apresentadas as tecnologias usadas, as mudanças operadas no HandSpy, bem como a avaliação efetuada a esta nova versão.

1.5 Organização da dissertação

A presente dissertação está organizada da seguinte forma:

- Capítulo 2 apresenta os princípios de um sistema de gestão de experiência em que se baseia o HandSpy.
- Capítulo 3 resume as principais tecnologias de utilizadas neste trabalho.
- Capítulo 4 descreve a conceção e implementação da nova versão.
- Capítulo 5 avalia de usabilidade do HandSpy 2.0.
- Capítulo 6 conclui a dissertação e faz propostas de trabalho futuro.

Capítulo 2

Estado de Arte

Este capítulo aborda a criação de um sistema colaborativo de gestão de experiências sobre os processos cognitivos da escrita. Inicialmente são descritas as principais propriedades de um sistema de gestão de experiências. Na segunda parte deste capítulo é descrito o HandSpy 1.0, um sistema de gestão de experiências que contém as caraterísticas necessárias ao estudo dos processos cognitivos da escrita.

2.1 Sistema de Gestão de Experiências

Com a elevada quantidade de dados recolhidos em experiências através de dispositivos digitais, há necessidade de criar sistemas de gestão para esses mesmos dados. O número de ficheiros recolhidos torna-se por vezes um grande desafio se não se recorre a sistemas de gestão de experiências. Este tipo de sistema é geralmente constituído por um interface do utilizador e um repositório de dados e tenta recriar os passos de uma experiência. Os dados descrevem as diferentes facetas da experiência, como por exemplo a lista dos participantes[11].

A Figura 2.1 representa o ciclo de vida de uma experiência, em que se pode observar as etapas que um conjunto de dados tem de percorrer num sistema de gestão de experiências e o seu relacionamento com as entidades. Seguidamente, vão ser enumeradas as etapas de uma experiência que se encontram representadas na Figura 2.1.

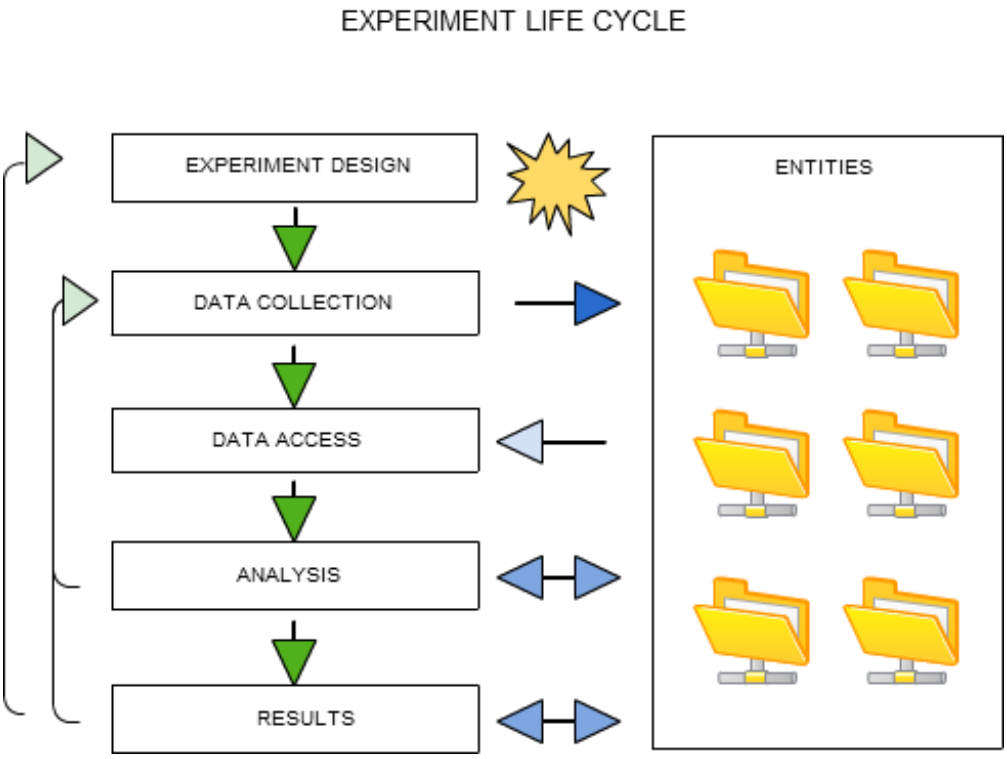

Figura 2.1: Ciclo de vida de uma experiência

Design da experiência: Esta fase contém a definição da experiência realizada. As definições podem ser alteradas no decorrer do ciclo da experiência, sendo armazenadas na base de dados juntamente com os dados recolhidos.

Recolha de dados: É neste estado que os dados são introduzidos na experiência, podendo ser repetido diversas vezes, caso haja necessidade de recolher mais dados. Isto deve-se à invalidade dos dados adquiridos ou à incapacidade de conceber resultados conclusivos.

Acesso aos dados: Os dados são recuperados para análise, validação ou partilha.

Análise: É a fase principal em que os dados são analisados, verificados e há a geração de resultados. Caso estes não sejam conclusivos, poderá ser necessário voltar à fase de recolha.

Resultados: Fase final do ciclo de vida de uma experiência, onde os resultados são gerados a partir da análise e validação destes resultados. Neste ponto da experiência os dados podem ser exportados em caso de sucesso, ou poderá ser necessário redesenhar a experiência ou recolher mais dados.

2.2 HandSpy 1.0

HandSpy 1.0 é uma aplicação Web criada para gerir experiências no estudo dos processos cognitivos da escrita. A aplicação baseia-se num modelo cliente-servidor seguindo uma arquitetura de 3 camadas: apresentação, lógica e dados. A camada de apresentação está desenvolvida em JavaScript, a camada lógica está escrita em Java e a camada de dados é uma base de dados XML. A base de dados XML justifica-se porque os dados recolhidos estão em InKML – formato de dados XML para representar tinta digital – e os restantes estarem em outras linguagens XML desenvolvidas para o efeito. Como referido, estes dados são recolhidos através de uma caneta digital e posteriormente carregados para a aplicação Web. Ao nível de design e estrutura do interface esta aplicação divide-se em seis ambientes de trabalho distintos para mais fácil compreensão do utilizador, como é ilustrado na Figura 2.2.

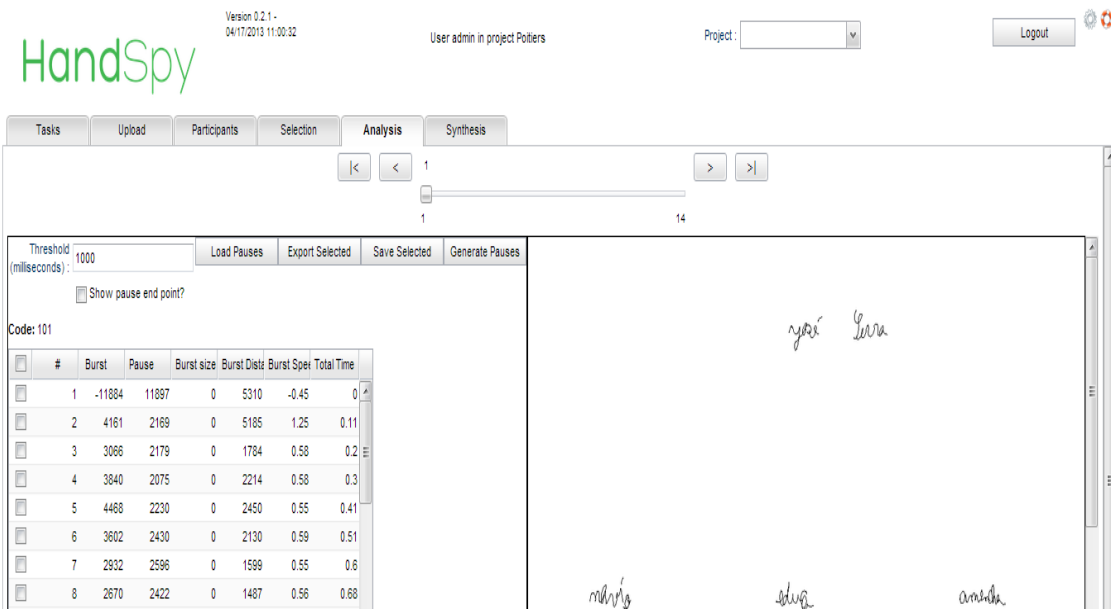

Figura 2.2: Desenho gráfico do HandSpy 1.0

Tasks: Nesta área de trabalho são identificadas as tarefas executadas pelos participantes num determinados estudo. Por exemplo, se os participantes redigiram uma tarefa de ditado e outra de alfabeto são agrupadas em tarefas diferentes.

Upload: Carregamento dos dados recolhidos pelos dispositivos digitais (caneta) e posteriormente divididos pelas tarefas dos participantes. Aqui o utilizador tem possibilidade de ver todos os dados de InkML recolhidos em imagem, para me-

lhor identificação do tipo de tarefa realizada.

Participants: Através de um ficheiro CSV, o utilizador pode importar todos os dados referentes aos participantes, como por exemplo, idade, língua materna, sexo e o seu código que permite associar aos seus dados aos protocolos recolhidos pelo dispositivo digital.

Selection: É baseado em tarefas e características dos participantes como escrever com mão esquerda, sexo ou idade. Isto permite selecionar e agrupar só os dados correspondentes a uma tarefa ou uma determinada característica que o utilizador posteriormente queira analisar em simultâneo.

Analysis: Uma das etapas mais importantes para realização de um determinado estudo, permitindo ao utilizador analisar todos os dados ao pormenor de cada protocolo segundo uma seleção anteriormente feita. O utilizador poderá excluir dados menos importantes ou errados que possam ter sido recolhidos e seguidamente guardar os outros na base de dados. Esta área de trabalho também efetua alguns cálculos com os dados importantes no processo cognitivo da escrita tais como a velocidade média, o número de palavras escritas e as médias totais de um determinado dado específico.

Synthesis: Depois de todo o processo feito na Analysis, a plataforma permite fazer o *download* de todos os dados e resultados relevantes. Estes resultados são então exibidos numa tabela que apresenta estatísticas globais sobre os dados processados na aba Analysis.

É de referir que o HandSpy 1.0 respeita as principais propriedades de um sistema de gestão de experiências, contendo todas as fases de um ciclo de vida de uma experiência.

Capítulo 3

Tecnologias de Referência

Este capítulo descreve sucintamente todas as tecnologias usadas na implementação da nova versão do HandSpy.

3.1 Interfaces Web

A vulgarização de aplicações Web deu origem a interface com o utilizador mais interativas. O JavaScript é uma linguagem de *scripting* muito utilizada na criação de interface Web e é suportada por todos os navegadores de referência. O JavaScript permite a utilização de *toolkits* que se tornaram uma prática comum no desenvolvimento de páginas web devido às funcionalidades dos seus *widgets*. Os *widgets* são objetos gráficos que podem ser utilizados para controlar a exibição e entrada de dados. Os mais comuns são *Menus* e *Tool Stripes*, *Tabset*, *Dialogs*, *Buttons*, *Tables* e *Trees*.

O suporte para Asynchronous JavaScript e XML (Ajax) é outra das caraterísticas importantes do JavaScript. O Ajax permite a criação de páginas dinâmicas, possibilitando que o cliente efetue pedidos ao servidor e que a página permaneça operacional enquanto aguarda respostas. Outra propriedade do Ajax é permitir a atualização dos dados da página Web sem implicar a atualização da mesma pelo utilizador. Existem enúmeras *frameworks* em JavaScript que além de oferecerem *widgets* com design diferente têm muitas outras funcionalidades. Na Tabela 3.1 estão representadas algumas das *framework* mais populares e as suas principais características.

Nome	License	DOM wrapped	Canvas	AJAX Support
Smartclient	LGPL&Comercial	Sim	Sim	Sim
Dojo	BSD&AFL	Sim	Sim	Sim
GWT	Apache	Sim	Sim	Sim
jQuery	MIT&GPL	Sim	Sim	Sim
Prototype	MIT	Não	Sim	Sim
MooTools	MIT	Não	Sim	Sim

Tabela 3.1: Comparação entre Frameworks Javascript

3.2 Entender o GWT

O Google Web Toolkit (GWT) é uma *framework* de código aberto lançada em Maio de 2006 pela Google e é uma ferramenta para o desenvolvimento de aplicações Web. O GWT diferencia-se das outras *framework* devido à linguagem de codificação ser o Java, obtendo-se assim a vantagem de o servidor e o cliente poderem ser codificados na mesma linguagem, o que simplifica os testes e a utilização de utilização de IDEs.

O GWT tem como principal característica compilar código Java para Javascript *(java-to-javascript)*, produzindo assim código capaz de ser executado em qualquer navegador. No entanto, esta *framework* não exclui a utilização do JavaScript por completo. Através do JavaScript Native Interface (JSNI) é possível interagir e integrar código em JavaScript permitindo ao programador usufruir das bibliotecas nessa linguagem[10].

O GWT contém várias bibliotecas de *widgets* e *panels*, como *TextBox, CkeckBox* ou *Grid* e ainda outras mais complexas como *ToolBar, DialogBox* e *Tree*, que podem ser estendidas criando assim funcionalidades mais avançadas.

3.2.1 Widgets e Panels

Ao nível dos objetos gráficos a API do GWT traz um variado conjunto de componentes. Estes estão agrupados numa hierarquia de classes, que podem ser usados diretamente ou alteradas e personalizadas em componentes mais complexos. A classe "UIObject" é a raiz desta hierarquia e está dividida em duas categorias principais *widgets* e *panels*, que definem ramos distintos da hierarquia. Os *widgets* são os componentes de interação com o utilizador, como botões ou caixas de texto. Os *panels* são componentes que têm como função integrar outros componentes no *layout*[18].

3.2.1.1 Widgets

Como foi anteriormente referido, os *widgets* são os componentes visíveis de uma aplicação web GWT. Nesta *framework* os *widgets* têm um papel fundamental na construção da aplicação e seguem o conceito de programação orientada a objetos do Java, o que permite serem facilmente modelados.

No ponto de vista do Java um *widget* é uma classe. Assim o programador poderá facilmente criar um componente e modelar as suas propriedades como um objeto Java, que por sua vez o compilador GWT converterá para código HTML e JavaScript necessários nas aplicações Web[18].

3.2.1.2 Panels

Nas aplicações GWT os *panels* têm a função de estruturar e posicionar os *widgets* na página Web. O *rootpanel* é um caso especial de *panel*, porque é através dos seus métodos que os outros *panels* vão ser agregados ao navegador. Os *panels* seguem uma filosofia idêntica aos *widgets* sendo objetos Java que podem ser modelados[18].

3.2.2 Comunicação Servidor Cliente

Nas aplicações Web a comunicação com servidor é um requisito necessário para acesso ao repositório de dados e outras funcionalidades executadas em *"back-office"*. No GWT a comunicação com o servidor é feita através de um mecanismo conhecido por Remote Procedure Calls (RPC). Na secção abaixo vamos abordar este tema mais detalhadamente.

O GWT RPC é uma forma de comunicação sobre HTTP entre o cliente e servidor de uma aplicação Web. A implementação de um serviço RPC GWT é baseada em Java *servlets*. No código do cliente é usado uma classe de *proxy* gerada automaticamente para fazer pedidos ao servidor. O código do lado do servidor (*servlet*), invocado a partir do cliente é habitualmente referido como um serviço.[8]

A comunicação cliente-servidor do GWT RPC é feita assincronamente. Isto significa que a comunicação com o servidor é independente da interação com o utilizador.

Esta vantagem da comunicação assíncrona permite que o utilizador continue a usar a aplicação sem ter que esperar que todos os dados sejam carregados[10].

Para construir um mecanismo de GWT RPC é necessário criar três componentes em Java:

1. Definir um interface para o serviço, que por sua vez estende `yourService` e contém todos os métodos RPC. Esta interface tem normalmente como sufixo "Service".
2. Criar uma classe de implementação com sufixo "ServiceImpl" que estende `yourServiceServlet`. Nesta classe são implementados os métodos definidos no interface mencionados no ponto um.
3. Para concluir a implementação dos três componentes falta definir um interface assíncrono que habitualmente tem sufixo "ServiceAsync", para o serviço ser chamado no lado do cliente.

No GWT RPC o servidor é responsável por executar os serviços invocados pelo cliente através dos métodos implementados na classe "ServiceImpl", ambos têm a capacidade de serializar e desserializar os dados que são transmitidos entre eles.

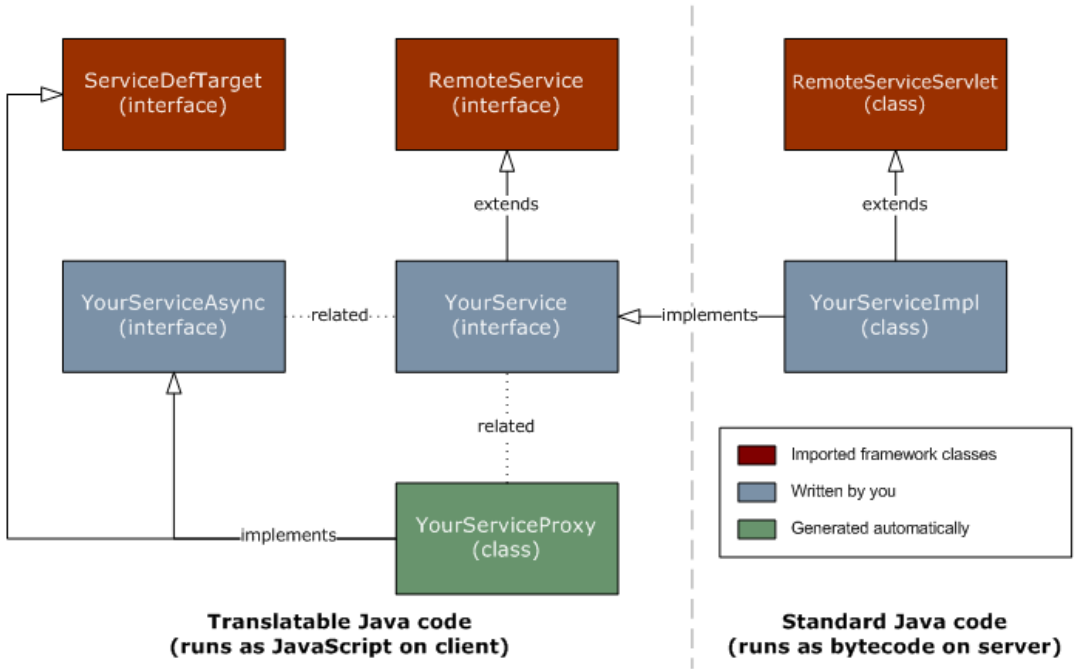

Figura 3.1: Diagrama do sistema de comunicação RPC[8].

Outro aspeto importante de salientar é que a implementação do serviço não é feita sobre o interface assíncrono, mas sobre o interface (Service). Cada implementação do serviço é um Servlet, estendendo `RemoteServiceServlet` e não HttpServlet, que vai tratar da transmissão e conversão de dados entre o cliente e o servidor. Na Figura 3.1 está esquematizado o mecanismo RPC entre cliente e o servidor usado nas aplicações GWT.

3.3 Smart GWT

A biblioteca Smart GWT é uma extensão da *framework* GWT que contém um conjunto de *widgets* e de outras características mais avançadas que tornam as aplicações Web mais ricas. O Smart GWT é mais que um novo conjunto de *widgets*. Esta biblioteca tem um novo conceito na sua arquitetura de comunicação. Os seus componentes são projetados para maximizar a capacidade de resposta e minimizar a carga do servidor.

3.3.1 Visão Geral da Arquitetura

Na arquitetura Smart Client a comunicação entre cliente e servidor tem como base o conceito de partilha de dados ”DataSoucer”. Este conceito separa o acesso aos dados da sua formatação e interação. Assim, pode haver diferentes formas de acesso aos dados e diferentes formas de apresentação, como tabelas, árvores ou formulário, podendo ser combinados de várias formas.

Ao utilizar “DataSource” como uma definição de dados compartilhada reduz a redundância entre o código da interface e código do servidor, aumentando a agilidade e reduzindo o esforço de manutenção. O processo de obtenção de ”DataSources” pode ser feito de várias maneiras, num *web server* ou fontes pré-existentes de metadados como XML ou JSON.[17].

3.4 Multimédia na Web

Esta secção tem como objetivo apresentar de uma forma mais detalhada algumas ferramentas que são usadas para incorporar ficheiros multimédia na Web. São abordadas

tecnologias como o HTML5 usado para incorporar vídeo em páginas Web, o formato PNG para criação de imagem e o software FFmpeg para conversão de áudio e vídeo.

3.4.1 Formato de Imagem PNG

O formato Portable Network Graphics, mais conhecido por PNG, surge como resposta a limitações técnicas e legais do formato de imagem GIF. As limitações técnicas deste formato surgiram quando este deixou de corresponder às exigências dos utilizadores, que começaram a utilizar hardware gráfico com mais de 256 cores. Por outro lado, à limitação legal surge devido a entidade que concebeu este formato que obriga os utilizadores a adquirir uma licença para utilização do mesmo nas aplicações que o empregam, limitando assim a criação de aplicações que o queriam usar[13].

O formato PNG reteve algumas das caracterizas do formato GIF, como suportar imagens 256 cores, canal de dados gráficos, compressão sem perdas, apresentação progressiva da imagem, transparência parcial, informação textual e independência da plataforma de hardware e software.

Foram também introduzidas outras características no formato PNG, como imagens de cores reais (imagens até 48 bits), transparência por meio do canal, deteção de corrupção de dados, maior rapidez na apresentação da imagem, o algoritmo de compressão e de código aberto, ordenação única dos bytes segundo a ordem de transmissão da rede e informação sobre a correção gama aplicada a imagem[13].

3.4.1.1 Formato PNG na Web

Até 1999 os formatos JPEG e GIF eram os mais presentes nas aplicações Web, mas com o surgimento do PNG algumas aplicações começaram a adotar este formato. Como foi referido anteriormente, o PNG surgiu principalmente para substituir o GIF e tem três principais vantagens, a transparência por meio do canal, correção gama e apresentação progressiva da imagem. Relativamente ao JPEG, o PNG não foi criado com o intuito de o substituir, contudo este é mais apropriado do que o JPEG para imagens com poucas cores[16].

3.4.2 Vídeo na Web

Atualmente, a maioria dos vídeos/filmes apresentados na web necessitam de um *plugin* (como o Flash) para serem reproduzidos. No entanto, diferentes navegadores podem ter diferentes *plugins* e com o surgimento do HTML5 esse problema foi resolvido. O HTML5 define um novo elemento que especifica uma forma padrão para incorporar um vídeo/filme numa página da web: o elemento <video>. Este elemento é suportado nos navegadores de referência (Internet Explorer 9, Opera, Firefox, Chrome e Safari), exceto no Internet Explorer 8 e versões anteriores. Para mostrar um vídeo em HTML5 é necessário uma formatação semelhante à seguinte:

```
...
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  <source src="movie.webm" type="video/webm">
  Your browser does not support the video tag.
</video>
...
```

Tabela 3.2: Elemento em HTML5 para incorporar vídeo num navegador

No elemento vídeo o atributo `controls` adiciona controlos de vídeo, como *play, pause* e volume. É uma boa prática incluir o atributo `width` e `height`, para reservar o espaço necessário ao vídeo quando a página é carregada. Contudo, se este atributo não for incluído, o navegador não irá reservar o espaço apropriado para o vídeo e o efeito será que o *layout* da página muda durante o carregamento. É também possível inserir um conteúdo de texto, entre <video>e </video>para alertar que aquele navegador não suporta o elemento vídeo. O elemento <video>permite várias <source>que podem ser formatos de vídeo diferentes. Atualmente, existem três formatos de vídeo suportados para o elemento <video>: MP4, WebM, e Ogg[6]:

O formato Webm usa o *codec* de vídeo VP8 e o de áudio Vorbis. O formato Webm é suportado pelo Chrome, Opera e Firefox, como se pode observar na Tabela 3.3, porém pode ser suportado no Safari e Internet Explorer através da instalação de um *add-on*. O formato Ogg utiliza um codec de vídeo Theora e de áudio Vorbis, sendo suportado pelo Firefox, Opera, Chrome e Safari através da instalação de um *add-on*. No entanto,

Navegador	Mp4	WebM	Ogg
Internet Explorer	Sim	Não	Não
Chrome	Sim	Sim	Sim
Firefox	Não	Sim	Sim
Safari	Sim	Não	Não
Opera	Não	Sim	Sim

Tabela 3.3: Formatos de vídeo suportados pelos navegadores

este formato não é suportado pelo Internet Explorer. O formato Webm é, geralmente, preferido sobre Ogg, uma vez que proporciona uma melhor qualidade e é compatível com mais navegadores. Contudo, este formato pode ser utilizado para apoiar versões de navegadores mais antigos, onde o suporte para Webm ainda não está disponível. MP4 usa o *codec* H264 para o vídeo e o ACC ou o MP3 para áudio. Este formato é suportado pelo Internet Explorer, Chrome e Safari, mas Chromium, Firefox e Opera não o suportam[7].

3.4.3 FFmpeg

O FFmpeg é um software multimédia capaz de codificar, descodificar e criar *stream* de vídeo e áudio em diversos formatos. É um software livre que inclui diversas bibliotecas de código aberto para converter entre formatos. FFmpeg contém bibliotecas como a *libavcodec* para codificar e descodificar *codecs* de áudio e vídeo e *libavutil* que contém funções para simplificar a programação, incluindo geradores de números aleatórios entre outras. Todas estas bibliotecas podem ser usadas com diversas ferramentas: *ffmepg* é uma ferramenta para converte ficheiros multimédia entre formatos na linha de comandos, *ffserver* é um servidor de *streaming* para transmissões ao vivo e *ffprobe* um analisador de fluxo de multimédia. É usada em inúmeros projetos livres e proprietários como, VLC e Google Chrome[4].

3.5 Repositórios de Dados

3.5.1 Base de dados em XML

Com o crescente uso do XML como formato de dados de armazenamento e consulta, foi necessário criar repositórios de dados para o suportar. Nesta secção são apresentados alguns tipos de base dados XML, o XML Nativo e XML Enabled, bem como a linguagem de consulta, o XQuery, que é usada neste tipo de base de dados.

3.5.1.1 Base de dados XML Enabled

A vantagem de utilizar bases de dados relacionais, e relutância de usar base de dados XML Nativas, levou ao aparecimento de base de dados XML Enabled. Recentemente vários motores de bases de dados relacionais têm algum suporte para XML, quer ao nível da importação/exportação, quer ao nível dos tipos de dados. As bases de dados XML Enabled oferecem três formas de armazenamento dos ficheiros XML dentro da estrutura relacional tradicional:

- A estrutura do ficheiro XML é utilizada para criar um conjunto de tabelas com base no XML schema[1].
- Os ficheiros XML são armazenados em Character Large Object (CLOB). O CLOB é uma grande coleção de dados de carateres em um DBMS (Database Manager System) utilizados para armazenar texto simples[2].
- Os ficheiros XML podem ser armazenados num tipo de XML nativo em sistemas Oracle object-relational DBMS, Microsoft SQL Server, IBM DB2 e PostgeSQL, e têm funções embutidas que permitem a gestão do conteúdos dos ficheiros.

3.5.1.2 Base de dados XML Nativas

Atualmente existem duas formas principais para armazenamento de dados em XML. Podemos armazenar os dados de uma forma tradicional em base de dados relacional ou numa base de dados XML Nativo. As bases de dados de XML são independentes de qualquer esquema, permitindo realizar indexação, que é a chave da eficiência deste

sistema[9].

Nas bases de dados XML nativas as consultas podem ser feitas usando dois tipos de linguagem, XPath e XQuery. O XPath é uma linguagem que se baseia no conceito de caminho. O XQuery pode incluir algumas instruções XPath, e possibilita gerar um ficheiro XML a partir de um conjunto de resultado com base no conteúdo do ficheiro XML. O XQuery Java API (XQJ) permite a criação e submissão de expressões XQuery à base de dados e manipulação de resultados definidos como objetos Java. A vantagem de usar bases de dados XML nativas em comparação com o uso das XML Enabled, é que as consultas não precisam de ser transformadas em consultas SQL. Existem vários sistemas de base de dados em XML nativas que são distribuídas sobre a forma comercial ou licença de software livre. Na Tabela 3.4 são apresentados alguns desses sistemas e comparação entre eles.

Nome	License	XQuery	XSLT	XML:DB API	XQJ API
MarkLogic Server	Comercial	Sim	Sim	Não	Sim
eXist	LGPL	Sim	Sim	Sim	Sim
BaseX	LGPL	Sim	Sim	Sim	Sim
Oracle B. DB XML	GPL	Sim	Sim	Sim	Sim
Sedna	Apache	Sim	Sim	Sim	Sim

Tabela 3.4: Comparação das bases de dados XML Nativas

3.5.1.3 XQuery

As base de dados XML usam uma linguagem de consulta de interrogação extremamente robusta para um determinado tipo de dados. Essa linguagem é denominada XQuery e foi projetada para ser aplicável a qualquer fonte de dados em XML. O XQuery é uma linguagem funcional que faz uso de expressões para representar as suas consultas, suportando os seguintes tipos de expressões na sua implementação:

- Expressões XPath.
- Expressões FLWOR.
- Expressões com operadores e funções.
- Expressões que envolvem os construtores dos elementos
- Expressões condicionais.

- Expressões quantificadas.
- Expressões de teste ou modificação dos dados.

As várias expressões podem ser usadas em conjunto. O XQuery permite que qualquer expressão XPath possa ser usada numa das suas cláusulas, como por exemplo na cláusula IF, mas só é válida se ela retornar um booleano[12].

Capítulo 4

HandSpy 2.0

Existe uma preocupação crescente com a usabilidade das aplicações para que consigam responder às necessidades dos seus utilizadores. Isto obriga a uma atualização permanente das mesmas, sejam elas de *Desktop* ou Web.

Sendo o HandSpy uma aplicação Web também teve essa necessidade de se atualizar. Deste modo esta nova versão do HandSpy teve como principal objetivo melhorar o design da aplicação e, por isso, procurou ser o reflexo daquilo que os utilizadores pretendiam. Uma vez que esta ferramenta é baseada numa aplicação Web, que usa um sistema cliente-servidor, foi essencial fazer restruturações tanto no repositório de dados como na parte lógica.

Este capítulo explica as transformações efetuadas entre a versão 1.0 e a 2.0. É de salientar que todas as alterações feitas no repositório de dados e na parte lógica da aplicação tiveram o cuidado de serem compatíveis com a versão 1.0, podendo assim o utilizador optar por usar essa versão se assim o desejar.

4.1 Interface da Aplicação

Na versão 2.0 o interface da aplicação é sem dúvida a principal alteração realizada relativamente à versão anterior, verificando-se grandes modificações na disposição dos componentes utilizados, organizado-os de uma forma mais lógica. Foram também acrescentadas novas formas de visualização dos dados, com principal destaque para

a reprodução de vídeos. Por último, mas não menos importante, a mudança no paradigma de implementação da aplicação passando-se a usar o SmartClient GWT[3] como principal ferramenta em vez do JavaScritpt. Esta secção descreve mais detalhadamente as modificações realizadas na interface da aplicação.

4.1.1 Design

Um dos principais objetivos da nova versão do HandSpy passa por criar um Interface com design agradável, intuitivo e contemporâneo. Para tal foi necessário interagir com alguns utilizadores durante o desenho da aplicação. Nesta secção é explicado o redesenho da interface com o utilizador da nova HandSpy 2.0.

4.1.1.1 Desenho da Aplicação

Esta nova versão do HandSpy é constituída no seu *layout* por um conjunto de quatro separadores, como se poder observar na Figura 4.1. Outro aspeto importante neste novo *layout* é que toda a zona de trabalho foi desenhada como tendo uma resolução fixa de 1028px largura e 768px altura, sendo composta pelas áreas dos separadores, botão de *logout* e pelo *logotipo*. Cada separador está dividido em duas regiões, a primeira é constituída por uma barra de ferramentas e a segunda por uma área de trabalho. De seguida lista é apresentada a explicação para cada um dos separadores:

- **Project Tab** contém na sua barra de ferramentas botões para manipular os eventos, relacionados com as tarefas e participantes, apresentados na área de trabalho. Este separador é dividido em três áreas, contendo informações relativas às tarefas aos participantes e ao projeto.
- **Upload Tab** tem como principal função fazer o carregamento de protocolos e atribuir-lhes uma tarefa e um participante. Essa associação é feita através do preenchimento do formulário existente na barra de ferramentas, onde o utilizador pode carregar o protocolo, definir a tarefa e o código do participante. A área de trabalho é repartida verticalmente: no lado esquerdo estão representadas miniaturas dos protocolos já carregados e no lado direito é visualizado em tamanho real aquele que está selecionado.

- **Analysis Tab** é sem dúvida um dos separadores mais complexos do HandSpy, incorporando na sua barra de ferramentas a seleção de protocolos, os controladores multimédia e as formas de navegação entre os protocolos selecionados. Tem também uma divisão vertical idêntica à do Upload, no lado esquerdo tem uma tabela contendo informações referentes ao *Burst, Pause, Time, Distance, Words*. A última linha da tabela apresenta o somatório de todos os valores registados. À direita é apresentado um protocolo em forma de imagem ou vídeo que inclui anotações retratam a informação contida nas linhas da tabela.
- **Administration Tab**, é uma das novas funcionalidades adicionadas na versão 2.0 do HandSpy, que permite gerir os utilizadores e os projetos existentes e tem a opção de adicionar e remover projetos e utilizadores, assim como associar um utilizador a um perfil.

É de referir que este novo desenho da aplicação teve como princípio agregar as funcionalidades existentes e criar outras, conferindo uma maior cronologia às etapas existentes na aplicação. Essa política também foi seguida nas barras de ferramentas, na Figura 4.1 está representado o separador de Analysis da nova versão do HandSpy que demonstra, de uma forma genérica, o esboço do novo *layout*.

Um separador é constituída por duas regiões distintas, como se observa na Figura 4.1. Na região 1 está representada a barra de ferramentas, comum a todos os separadores, que possui um conjunto de botões que executam eventos sobre a região 2. A segunda região é repartida em duas ou mais sub-regiões que servem quase exclusivamente para apresentar informação. Esta janela encontra-se dividida em duas sub-regiões, em que a assinalada com um 3 representa uma tabela com dados e a assinalada com um 4 uma imagem representativa dos mesmos.

4.2 Multimédia

Com a utilização do XML para armazenamento dos dados a sua conversão em vários formatos torna-se mais fácil e eficiente. Aproveitando essa vantagem foi implementada uma nova forma de visualização dos dados no HandSpy, possibilitando aos seus utilizadores não só o estudo dos dados recolhidos através de imagem, como na versão

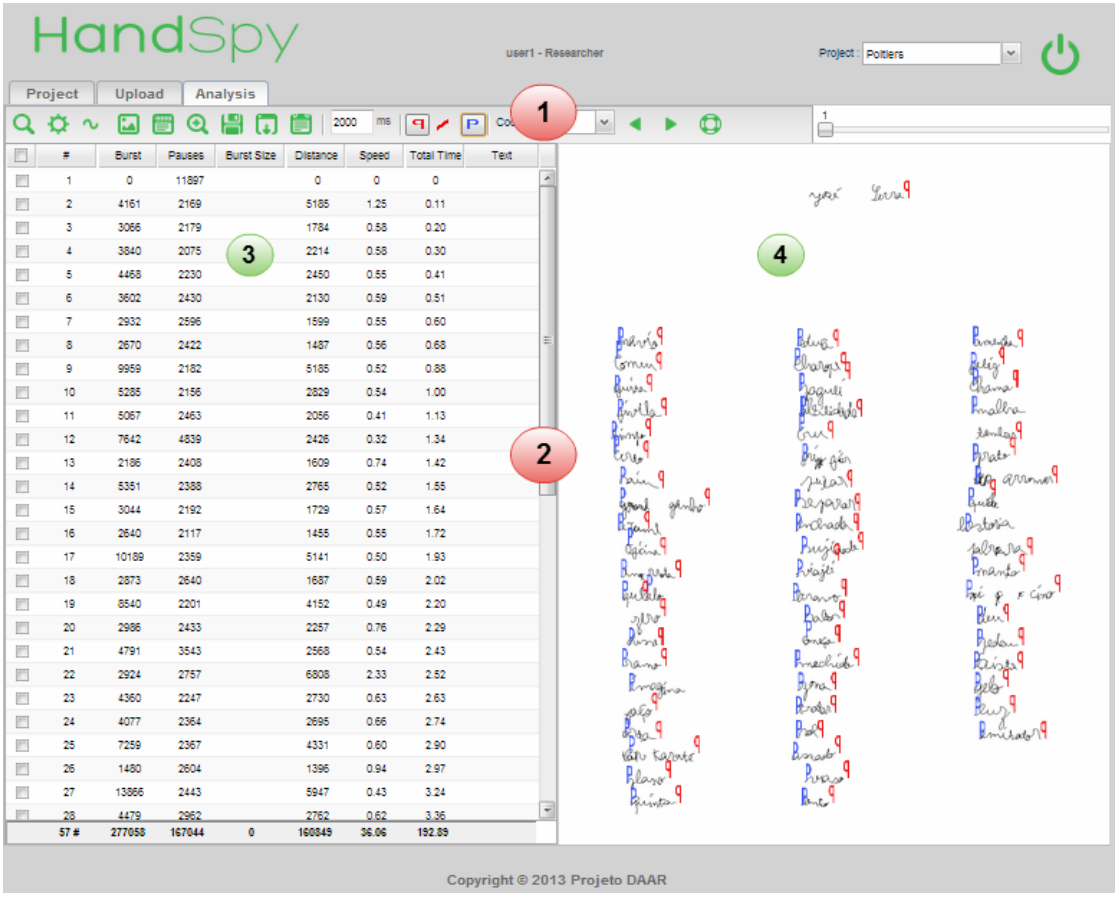

Figura 4.1: Interface do HandSpy 2.0

1.0, mas também em vídeo. Foram também incorporadas novas funcionalidades de interação com as imagens e os vídeos. Todas estas atualizações descritas situam-se no separador de Análise e serão apresentadas de uma forma mais aprofundada nas secções seguintes.

4.2.1 Anotações Gráficas

As anotações utilizadas na representação gráfica dos dados têm como princípio representar momentos de relevância e situá-los geograficamente, permitindo assim ao investigador uma perceção pormenorizada dos dados em análise.

Como se pode observar na Figura 4.2 existem atualmente 3 tipos de anotações, sendo **p** o início do *burst*, **q** o fim do *burst* é o começo da *pause* e a **linha** a ligação entre o fim de um *burst* e o começo do seguinte. A anotação **linha** foi introduzida nesta nova versão com o principal propósito de distinguir os conjuntos individualizados de um bloco

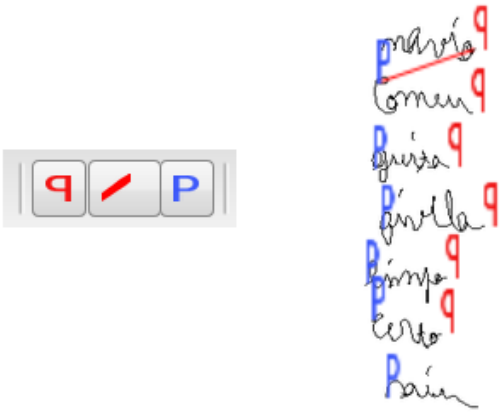

Figura 4.2: Anotações **P**, **Q** e **Linha**

q e **p**. É de referir que a **linha** só aparece no último bloco selecionado para facilitar a sua localização no contexto das restantes anotações. Um conjunto das anotações **q** , **p** e **linha** corresponde a uma linha específica da tabela existente em Analysis.

4.2.2 Seleções Gráficas

A seleção de linhas da tabela do separador Analysis, que correspondem a um determinado conjunto de anotações na imagem, já era possível na versão 1.0. Contudo na nova versão há a possibilidade do utilizador selecionar zonas da imagem ou vídeo e estas selecionarem linhas na tabela.

Essa seleção é feita da seguinte forma: o utilizador desenha uma determinada área a verde na imagem ou no vídeo, arrastando o rato e criando um retângulo. Esta área corresponderá a um conjunto de linhas da tabela como se pode observar na Figura 4.3. Este processo pode também ser utilizado para desselecionar uma zona já selecionada, para isso basta o utilizador carregar no botão menos da janela de ferramentas de desenho, passando o retângulo de verde a uma tonalidade avermelhada.

Cada linha da tabela contém as coordenadas de onde as suas anotações acontecem na imagem ou vídeo, se elas se intercetarem com as do retângulo desenhado essa linha será selecionada. Este processo é idêntico para o evento de desselecionar. Todos estes eventos, que permitem ao utilizador desenhar objetos na imagem ou no vídeo, são possíveis devido à utilização do *canvas* do HTML5 conjuntamente com o JavaScript.

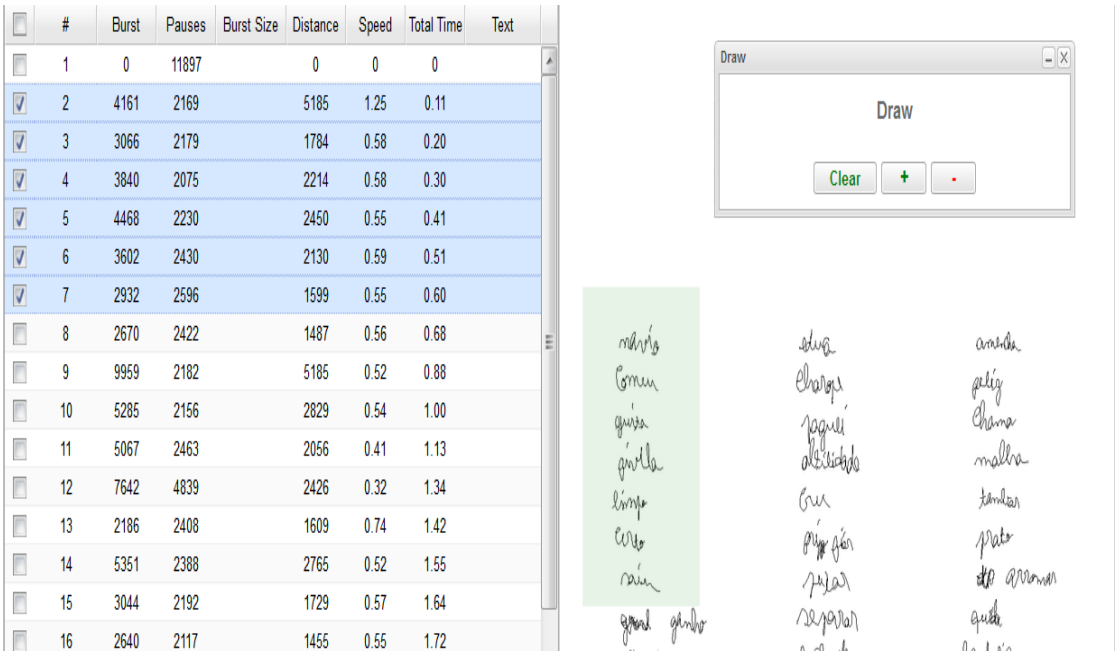

Figura 4.3: Seleção dos dados

4.3 Reprodução de Vídeos

A incorporação do vídeo, como uma nova forma de visualização dos dados recolhidos, tem o propósito de mostrar de uma forma sincronizada aquilo que aconteceu na produção dos textos. Foram realizados diversos testes e estudos prévios antes de se conseguir um resultado apropriado para ser integrado na aplicação.

4.3.1 Implementação

Um vídeo é um conjunto de imagens que são reproduzidas sequencialmente a uma determinada velocidade dando uma noção de movimento. Para criar imagens no HandSpy foi implementado um algoritmo em Java, que utiliza os ficheiros InkML para criar imagens em determinados momentos de tempo. Na implementação optou-se por recorrer a um algoritmo ja existente no HandSpy, que através num conjunto de *strokes* consegue criar uma imagem. Na Tabela 4.1 está representado o algoritmo utilizado.

```
...
public void getImage(OutputStream stream){
              for (String point : commas.split(trace.getValue().trim())) {
                     String[] values = blancs.split(point.trim());
                     xPoints[nPoints] = (int) (Integer.parseInt(values[0])
                         * xScale);
                     yPoints[nPoints] = (int) (Integer.parseInt(values[1])
                         * yScale);
                     nPoints++;
                     graphics.drawPolyline(xPoints, yPoints, nPoints);
              }
              ImageIO.write(image, "png", stream);
}
...
```

Tabela 4.1: Algoritmo de geração de uma imagem

Apesar de este algoritmo gerar um polígono, através de um conjunto de traços, não permite criar um conjunto de imagens durante uma sequencia no tempo. Foi por isso necessário conceber uma nova versão deste algoritmo para permitir a criação de uma imagem a um instante de tempo definido.

Na Tabela 4.2 está apresentada uma nova versão modificada do algoritmo anterior. Este algoritmo inicialmente itera sobre o elemento `trace` dos ficheiros InkML para obter a informação de cada ponto que está no seguinte formato [X, Y, Timestamp]. Quando o `timestamp` de um ponto é superior ao tempo guardado na variável `nextime`, somada à taxa de vídeo em milissegundos, esse ponto é adicionado ao polígono e é criada uma imagem.

```
...
private void makeFramesOfVideo(IdProtocol IdProtocol){
       for (String point :
           commas.split(trace.getValue().trim())) {
              String[] values = blancs.split(point.trim());
              xPoints[nPoints] = (int)
                  (Integer.parseInt(values[0]) * xScale);
              yPoints[nPoints] = (int)
                  (Integer.parseInt(values[1]) * yScale);
              time = Long.parseLong(values[2]); // Time in
                  milliseconds
              if (previousTime == 0) {
                     previousTime = time;
                     nextTime = time + 1000 / FRAMES_PER_SECOND;
                     } else {
                     if (time >= nextTime) {
                     graphics.drawPolyline(xPoints, yPoints,
                         nPoints);
                     if (nPoints > 1) {
                            xPoints[0] = xPoints[nPoints - 1];
                            yPoints[0] = yPoints[nPoints - 1];
                            xPoints[1] = xPoints[nPoints];
                            yPoints[1] = yPoints[nPoints];
                            nPoints = 1;
                            }
                            while (time >= nextTime) {
                                   previousTime = nextTime;
                            nextTime = previousTime + 1000 /
                                FRAMES_PER_SECOND;
                                   frame++;
                                   addFrame(frame);
                                                 }
                                          }
                                   }
                     graphics.drawPolyline(xPoints, yPoints,
                         nPoints);
                     addFrame(frame);;
                     }
              }
                     addFrame(frame);
       }
       ...
```

Tabela 4.2: Algoritmo de geração de imagens

O diagrama da Figura 4.4 representa o funcionamento do algoritmo da Tabela 4.2 onde se observam as imagens criadas durante alguns instantes de tempo. Cada imagem criada é uma réplica da anterior mais alguns traços do polígono.

Após a criação das imagens estas são convertidas em vídeo através de um programa

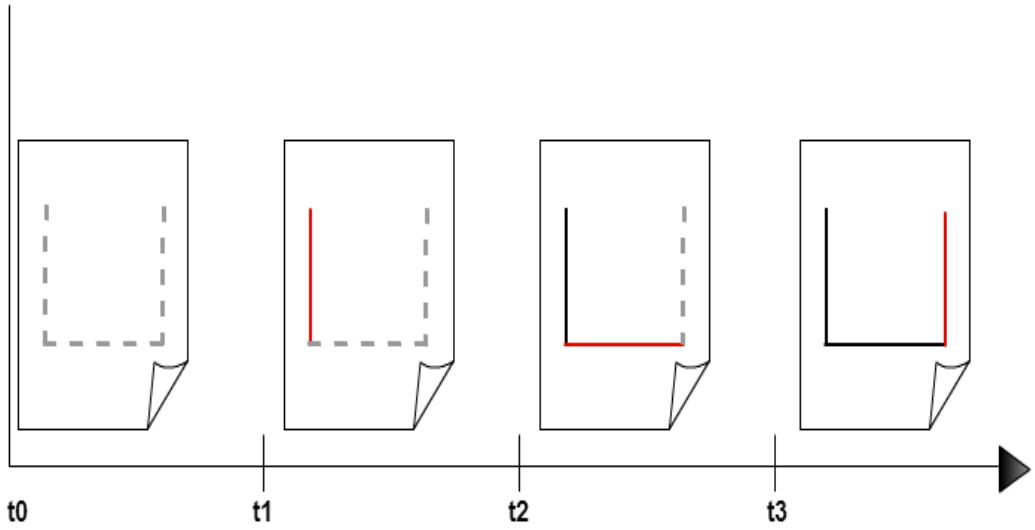

Figura 4.4: Diagrama de execução do algoritmo, durante o tempo

de conversão FFmpeg, criando-se o mesmo vídeo em formatos diferentes, com a finalidade de cobrir os formatos suportados pelos navegadores de referência. A conversão das imagens para vídeo é realizada a uma taxa de 20 frames por segundo. A taxa de vídeo é um compromisso entre a fluidez da animação e o tamanho, isto é, se a taxa do vídeo for muito elevada temos uma fluidez muito maior, a animação é mais rápida e o tamanho do vídeo é menor em espaço. A fluidez da animação é inversamente proporcional ao tamanho do vídeo, ou seja, se a taxa aumentar a fluidez aumenta e o tamanho diminui e vice-versa.

Todo este processo foi alvo de vários testes antes de se obter um resultado final. Inicialmente optou-se por criar os vídeos *"on-the-fly"* mas isto não se concretizou devido à demora na criação dos mesmos, tendo-se optado por armazená-los no servidor. Foram testados os melhores formatos de imagem e vídeo em termos de qualidade e otimização, optando-se por usar o PNG em vez do Mpeg nas imagens e nos vídeos a utilização do Webm e Mp4 em detrimento do Ogg.

4.3.2 Interação

O vídeo permite realizar anotações e seleções similares à imagem, no entanto a sua reprodução depende muito daquilo que está selecionado. Por exemplo, se nenhuma linha (bloco *burst-pause*) estiver selecionada na tabela o vídeo é reproduzido no seu

todo, caso contrário ele só é reproduzido durante o tempo que os blocos de *burst-pause* forem selecionados. Apesar de o vídeo ser uma forma de visualização diferente da imagem, foi necessário que cada um deles tivesse as mesmas funcionalidades, tornando assim o HandSpy uma aplicação mais coerente e intuitiva para o utilizador.

4.4 Utilização de Perfis

Como não existia um sistema de controlo e administração de sessões e utilizadores na versão anterior, foi indispensável nesta nova versão criar uma forma dos gerir. Para resolver este problema foi adotado um sistema que associa um determinado perfil a uma sessão, isto é, cada utilizador tem um perfil associado e esse perfil permite-lhe ter determinadas permissões na aplicação. Na lista seguinte estão descritos os vários perfis existentes no HandSpy:

SuperAdministrator: Este perfil está associado ao *admin*, e permite-lhe somente ter acesso ao separador de administração.

Administrator: Pode ser associado a vários utilizadores dando-lhes permissões mais sensíveis, tais como as de remoção de conteúdo, mas só nos projetos a que pertencem, e ao separador de administração, que lhes permite criar e associar utilizadores aos projetos que administram.

Researcher: Os utilizadores têm acesso aos separadores de trabalho (Project, Upload e Analysis), porém não podem remover qualquer tipo de dados da aplicação. Neste perfil todos os componentes de remoção na barra de ferramentas aparecem inativos e com uma cor acinzentada.

Para controlar mais eficazmente todo este sistema de associar um utilizador a um perfil existe na Administration Tab um conjunto de funcionalidades de criação e remoção de utilizadores e associação de utilizadores a um determinado perfil. A introdução deste sistema veio colmatar uma lacuna existente na versão 1.0, dando aos investigadores uma maior autonomia na gestão da plataforma Web do HandSpy.

4.5 Construção do Interface

A construção da interface passou por uma grande mudança a nível estrutural na sua implementação. Esta alteração ocorreu na linguagem de programação utilizada, em que deixou-se de utilizar o JavaScript para usar o GWT. Esta mudança teve como objetivo criar uma aplicação mais robusta, eficiente e intuitiva para o utilizador, tirando partido de todas as potencialidades da linguagem de programação Java que torna melhor a comunicação com a parte lógica de aplicação, tornando o seu processo mais transparente.

A nova estrutura da arquitetura, no lado do cliente, é composta por um conjunto de funções bem individualizadas para cada área de trabalho, como se pode observar na Figura 4.5. É considerada uma Área de trabalho cada um dos separadores que incorpora a aplicação. Na lista abaixo estão descritas as principais funções abstratas para cada Área de Trabalho e restante hierarquia.

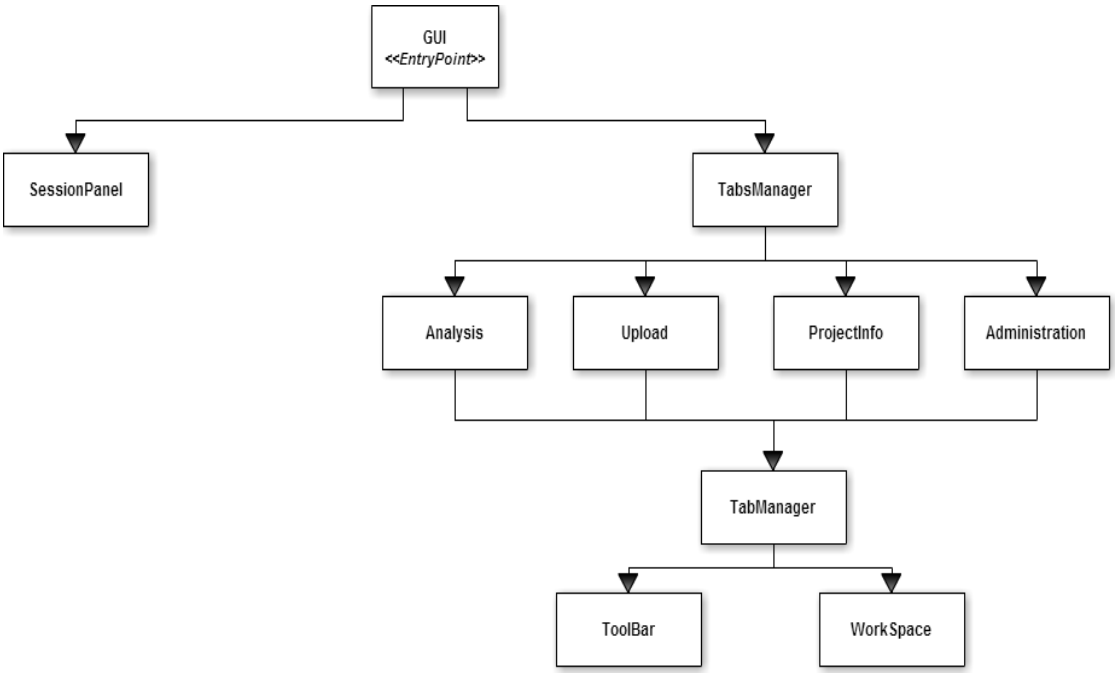

Figura 4.5: Diagrama da arquitetura do cliente.

Gui: é a função que está no topo da hierarquia da aplicação, sendo a raiz de toda a estrutura do programa no lado do cliente. No módulo GWT, esta função é aquela que foi implementada no *EntryPoint* da aplicação, similar ao método principal de um programa Java padrão.

SessionPanel: Esta função é responsável por verificar se o utilizador está autenticado e pela criação dos componentes de *login* e *logout*.

TabsManager: Com a existência de perfis associados ao utilizador, a criação das áreas de trabalho são distintas, dependendo do perfil. A função TabsManager tem como objetivo criar os separadores que estão associados a um perfil.

TabManager: É uma função específica para cada separador que incorpora em si a barra de ferramentas e a área de trabalho.

ToolBar: Nesta função é criada a barra de ferramentas e todos os componentes e eventos que a integram.

WorkSpace: Esta função contém todos aqueles componentes que exibem a informação e é usualmente composta por *widgets*, como *ListGrid* e *HTMLPanels*.

A arquitetura da aplicação, no lado de cliente, divide-se em dois grupos bem distintos: funções globais, como Gui, SessionPanel e TabsManager, e as específicas por separador a função TabManager, ToolBar e WorkSpace. O Separador de Analysis ainda estende mais algumas funções na sua hierarquia devido ao número elevado de eventos e componentes que integra.

4.6 Versão 1.0 vs 2.0

Como foi anteriormente referido a grande mudança feita na atualização do HandSpy foi o interface do utilizador. Esta secção tem como principal objetivo demonstrar as diferenças visuais existentes entre as duas versões. Nas Figuras 4.6 e 4.7 estão apresentadas as duas versões do HandSpy, no lado esquerdo a versão 1.0 e no lado direito a 2.0.

Ao comparar visualmente estas duas versões pode-se constatar que na versão 2.0 há uma estrutura mais organizada e coerente dos componentes e do aproveitamento dos espaços do ecrã. Com a inclusão da barra de ferramentas todos os botões estão muito mais organizados. Cada botão tem um ícone que exprime a sua função e estão todos organizados de uma forma sequencial na barra de tarefas, facilitando o trabalho dos investigadores. Apesar da versão 1.0 cumprir todas as funções para que foi criada,tinha

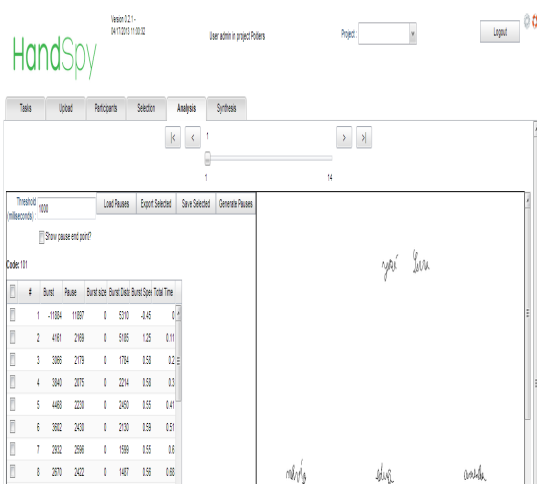

Figura 4.6: Versão 1.0 do HandSpy

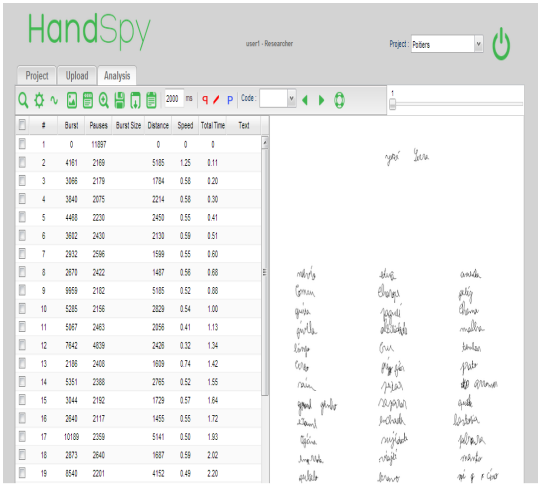

Figura 4.7: Versão 2.0 do HandSpy

aspetos no seu interface que continham enúmeras lacunas. Com este novo interface, à primeira vista pode-se afirmar que o HandSpy 2.0 tem um aspeto mais contemporâneo.

4.7 Alterações na Arquitetura

As atualizações e a restruturação da arquitetura têm como finalidade melhorar a eficiência e robustez da aplicação. A mudança para o GWT teve um impacto significativo na arquitetura da camada lógica. Nesta secção são discutidas todas as alterações feitas ao nível do servidor e da base de dados do HandSpy.

4.7.1 Lógica

Na camada lógica do HandSpy foi fundamental alterar o sistema de pedidos feitos ao servidor. Isto deveu-se à utilização de um novo canal de comunicação entre o cliente-servidor, denominado RPC. O RPC, como mencionado na secção 3.2, é usado em aplicações Web que recorrem a *framework* GWT para fazer a comunicação entre o cliente-servidor. Contudo, nem todos os pedidos realizados pelo cliente recorrem ao RPC para comunicar com o servidor, alguns continuam a ser feitos através do processo anteriormente criado para a versão 1.0 devido à utilização do SmartClient GWT que permite usar pedidos REST para criar DataSource em formatos facilmente manipulados para apresentar informação. No HandSpy o formato usado na criação das DataSource é XML, uma vez que a base de dados é em XML Nativo. Os pedidos

REST são usados sobretudo quando é necessário obter muita informação do servidor para ser apresentada.

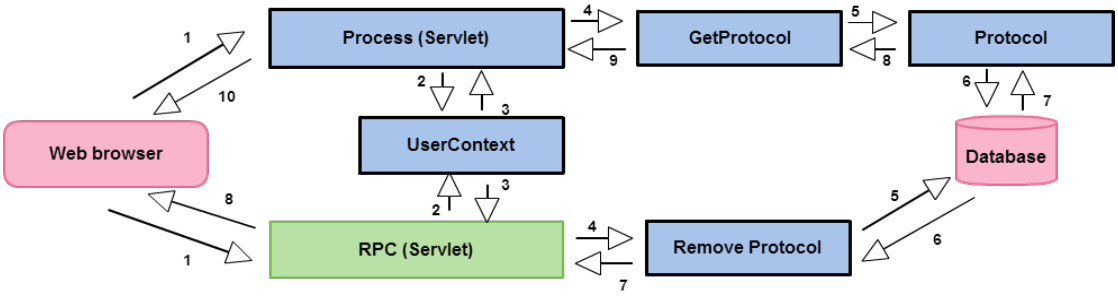

Figura 4.8: Diagrama de fluxo de um pedido ao servidor.

Na Figura 4.8 está representado um diagrama de fluxo de pedidos feitos pelo cliente ao servidor e posteriormente a resposta, usando os dois métodos de comunicação REST e RPC, onde REST é usado para o pedido de uma imagem e o RPC para um pedido de remoção de um protocolo. Quando o pedido é feito através de REST, o `Process` recebe-o e verifica a autenticação da sessão no `UserContext`. No caso de ser uma sessão válida é executado o comando `GetProtocol`, que por sua vez acede à base de dados através da classe `Protocol` para ir buscar os dados do ficheiro InkML e gerar uma imagem. Depois de este processo terminar a imagem é envida para o cliente através do fluxo de resposta.

No RPC a autenticação da sessão é comum com a do REST, contudo no caso do RCP não é necessário executar um comando porque o RPC tem implementada uma classe que estende `RemoteServiceServlet` no lado do servidor que executa todos os pedidos feitos. No caso esquematizado na Figura 4.8, depois do pedido chegar ao servidor, o `Remove Protocol` acede a base dados e remove o protocolo indicado. Quando terminado este processo é enviada uma resposta para o cliente através do RPC, confirmando se o pedido foi executado com sucesso ou falhou. Esta modificação da comunicação na versão 2.0 deve-se ao fato da tecnologia usada na implementação do cliente ser em GWT que usa o RPC para comunicar com o servidor. A vantagem da utilização do RPC como método de comunicação entre o cliente-servidor é ser feita quase ao mesmo nível porque ambos são codificados em Java.

4.8 Base de Dados

No HandSpy os dados recolhidos são armazenados em ficheiros XML, por isso é usado um repositório em XML Nativo. Na versão 2.0 este repositório de dados sofreu algumas atualizações estruturais muito significativas. Na Figura 4.9 está esquematizado o novo modelo do repositório de dados.

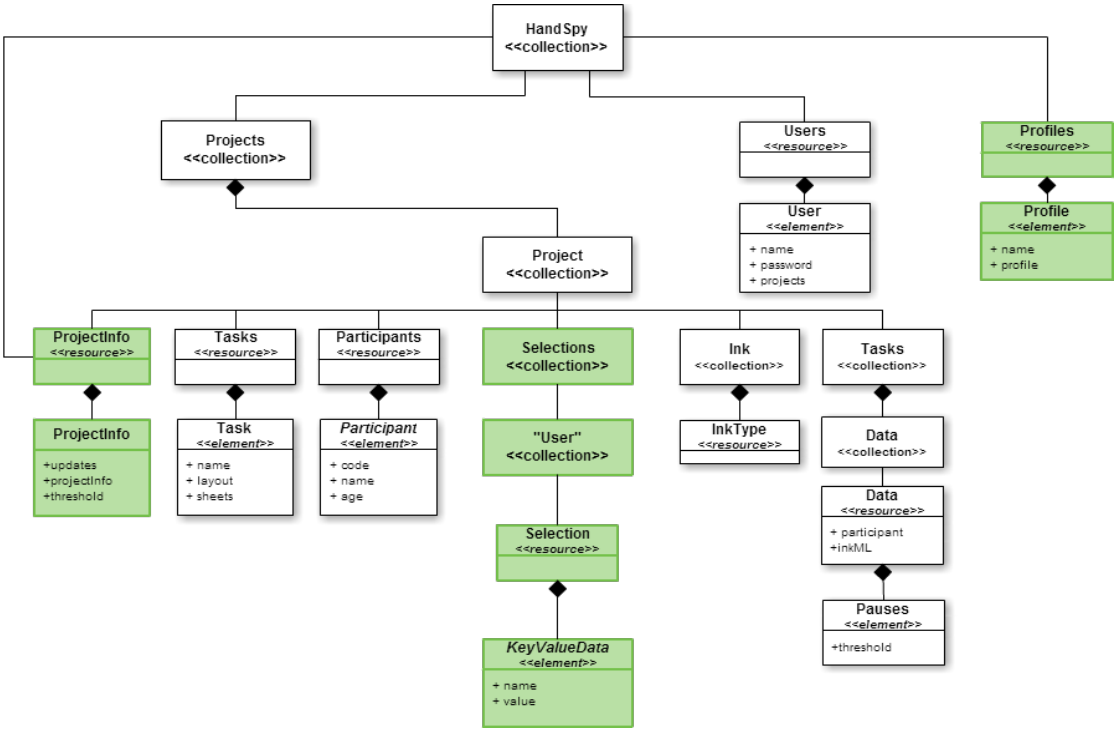

Figura 4.9: Diagrama da base de dados

Como se pode visualizar na Figura 4.9 a estrutura do repositório de dados tem inicialmente dois conjuntos de entidades que armazenam informações sobre o utilizador, `Profile` e `Users`. Ao mesmo nível destas duas entidades encontra-se uma coleção que contém os vários projetos existentes. Cada projeto tem um conjunto de entidades, sendo estas `ProjectInf`, `Tasks`, `Participants` e as seguintes colecções: `Selection`, `Ink` e `Tasks`.

4.8.1 Definição do Esquema da Base de Dados

Como já foi referido, o esquema da base de dados é definido através de um conjunto de coleções e entidades que estão relacionadas entre si. A lista seguinte descreve

pormenorizadamente cada entidade representada na Figura 4.9.

- A entidade `Profile` é composta pelos seguintes elementos: nome do utilizador, o seu perfil (exemplo investigador ou administrador) e os projetos a que o mesmo tem acesso.
- `Users`, esta entidade contém informações sobre o *username* do utilizador e a password para este se autenticar.
- `ProjectInfo` é uma entidade que se situa dentro da coleção `Project` e é definida por três elementos: `updates` onde é guardada a última atualização feita na aplicação, `projectInfo` tem informações gerais sobre o projeto e `threshold` que contém todos os `threshold` adicionados a um determinado projeto.
- `Tasks` é composto pelo atributo `name` que é usado para definir o nome da tarefa realizada e `sheet` que define o intervalo do caderno associado a esta tarefa.
- `Participants`, esta entidade tem o atributo `code` que corresponde ao código de cada participante e é único e o elemento `KeyValueData` que contém os detalhes dos participantes.
- `Selection` é também uma entidade com o elemento `KeyValueData` que e guarda as últimas seleções de pesquisa feitas por cada utilizador.
- `InkType` armazena os ficheiros no formato InkML e nunca é alterado, contendo simplesmente ficheiros de leitura.
- `Data` contém dois atributos que são o `participant`, que identifica o participante, e o InkML, que armazena dados guardados depois de analisados no separador de `Analysis`. Estes dados guardados são referentes a um tempo definido no elemento `threshold`.

4.8.2 Modificações na Arquitetura

Na Figura 4.9 estão representados a verde todos os estados que foram alterados ou acrescentados à estrutura da base de dados:

- A primeira alteração efetuada no modelo foi a adição da entidade `Profile` que guarda o perfil que um utilizador pode ter dentro de um projeto.

- Criação de uma coleção `Selection` que por sua vez está dividida em outras coleções designadas pelo nome dos utilizadores e que dentro destas tem uma entidade que guarda as seleções efetuadas por um utilizador. Esta alteração permite que cada utilizador tenha uma seleção independente da dos restantes.
- Acrescento da entidade `ProjectInfo` que guarda informações úteis sobre o projeto, como já foi referido em cima.

4.9 Aumento da Performance

Uma das razões principais para uma nova versão do HandSpy foi o aumento da performance nas pesquisas feitas à base de dados. Para isso foram alteradas as metodologias de pesquisa que inicialmente eram feitas em XPath e optou-se por usar uma linguagem muito utilizada na pesquisa em bases de dados XML Nativas, o XQuery. Com a inclusão desta nova linguagem, verificou-se uma significativa melhoria na performance das pesquisas feitas. A razão dessa melhoria deve-se ao fato de anteriormente haver junções de dados de vários documentos feitos na camada lógica, que com o XQuery, passaram a ser feitas na base de dados. Por exemplo, ao utilizar o XPath numa pesquisa de procotolos com participantes do sexo masculino era necessário obter todos os códigos dos participantes do sexo masculino, fazendo uma pesquisa num ficheiro, para que depois esses códigos fossem cruzados com a coleção de protocolos da base de dados. Com a utilização do XQuery não é necessário usar todo este processo pois a própria *query* cruza os ficheiros necessário e retorna o resultado. Na Tabela 4.3 estão representados alguns testes para verificar o aumento da performance com a utilização do XQuery comparativamente com o XPath. Para que tudo isto fosse possível foi necessário atualizar o Software da base de dados para a versão 2.0, uma vez que a versão 1.4 da eXist-db[5] apresentava *bugs* que tornavam inutilizável o XQuery.

Todos estes testes consistem em pesquisas feitas no eXide IDE à base de dados do HandSpy, fornecido pela eXist-db na sua instalação. O IDE permite não só visualizar os resultados obtidos mas também o tempo de execução. Inicialmente estes testes foram divididos em dois grupos pesquisas simples, em que apenas é utilizado um

Pesquisas Simples	XPath	XQuery
Teste 1	>10s	0,032s
Teste2	>10s	0,027s
Pesquisas Complexas	XPath	XQuery
Teste 3	>30s	16,871s
Teste 4	>30s	2,432s

Tabela 4.3: Comparação da performance entre XPath e XQuery

ficheiro XML para a pesquisa e as complexa onde são relacionados vários ficheiros ou coleções entre si. Podemos verificar que as diferenças são muito significativas em todos os testes. Por exemplo, nas pesquisas simples o tempo de pesquisa do XQuery nunca ultrapassa um segundo comparativamente com XPath que é sempre superior a 10 segundos. Esta modificação revelou-se de uma importância tal que, quando observamos a Tabela 4.3 de pesquisas complexas, os tempos das pesquisas com XPath são de tal forma elevados que podem iludir o utilizador para a existência de uma falha do sistema.

Capítulo 5

Avaliação de Usabilidade HandSpy 2.0

No âmbito do projeto (Desenvolver, Automatizado e Autorregulado os Processos Cognitivos na Composição da Escrita), realizado na Faculdade de Psicologia, foi concebido o sistema HandSpy. Este sistema destina-se a relacionar a automação de processos de escrita e autorregulação de outros com o desenvolvimento desta competência.

Antes de chegar ao HandSpy 2.0 este sistema já tinha sofrido um processo de avaliação de usabilidade. Esta avaliação foi feita pelos utilizadores que acompanharam o processo de desenvolvimento do HandSpy 1.0. Todos estes utilizadores intervieram numa fase de recolha e armazenamento dos dados no sistema e, posteriormente, na sua análise. Além da avaliação que foi feita durante todo o desenvolvimento foi também realizada uma avaliação com base no preenchimento de um questionário. Atualmente nem todos os utilizadores que acompanharam o desenvolvimento do HandSpy 1.0 se encontram a trabalhar no projeto DAAR. Porém todos eles já tiveram o contacto com o primeiro sistema criado e participaram no desenvolvimento da nova versão do sistema. Foi-lhes então pedido para realizar um questionário de modo a avaliar a usabilidade deste novo sistema que será comparado com a avaliação feita ao sistema anterior. Essa avaliação será exposta nas secções seguintes.

5.1 Avaliação Heurística

A avaliação heurística é um dos métodos mais eficientes e populares na identificação de um problema na utilização de um interface pelo utilizador. É usada para resolver esses problemas através de um conjunto de regras e métodos. Rolf Molich e Jakob Nielsen[15], descrevem uma avaliação heurística como *"um método informal de fácil análise em que um interface é apresentado a um grupo de avaliadores e lhes é pedido para comentarem sobre ele"*. O consultor de usabilidade Jakob Nielsen, depois de ter avaliado várias heurísticas de usabilidade, criou um conjunto de heurísticas que ainda hoje são usadas[14].

Visibilidade do estado do sistema: O sistema deve sempre informar o utilizador em que estado ele se encontra, através de um *feedback* apropriado e num tempo razoável

Compatibilidade: O sistema deve usar uma linguagem familiar para o utilizador, em vez de termos orientados ao sistema. As informações devem aparecer numa ordem natural e lógica.

Controlo do utilizador e liberdade: O utilizador utiliza frequentemente funções por engano. O sistema deve ter um suporte que permita desfazer ou fazer para recuperar de funções que foram escolhidas por engano.

Consistência e padrões: O interface deve usar cores, palavras, *layout* ou situações padronizadas para não confundir o utilizador.

Prevenção de erros Evitar a ocorrência de erros, apresentando aos utilizadores a opção de confirmação antes de uma operação crítica.

Reconhecimento ao invés de lembrar: Minimizar a carga de memória do utilizador, tornando objetos, ações e opções sempre visíveis. As instruções para o uso do sistema devem estar sempre visíveis ou facilmente recuperadas sempre que for necessário.

Flexibilidade e eficiência de uso: Permissão para que o utilizador personalize ações frequentes, o que muitas vezes permite acelerar a interação para outros utilizadores experientes e inexperientes.

Estética e design minimalista: As informações apresentadas devem ser relevantes.

Ajudar os utilizadores a reconhecer, diagnosticar e recuperar de erros: As mensagens de erro têm que ser claras, indicando com precisão o problema e sugerindo uma solução. Estas mensagens devem conter uma linguagem simples e não devem usar código.

Ajuda e documentação: Ajuda e documentação deve estar sempre disponível. Esta informação deve ser de pesquisa fácil, concreta e não ser muito grande, focando-se na tarefa realizada pelo utilizador.

5.2 Avaliação do HandSpy 2.0

A avaliação de usabilidade feita ao HandSpy 2.0 é baseada nos resultados de um questionário apresentado no Apêndice A, sendo uma réplica do realizado no estudo da avaliação de usabilidade da versão anterior do HandSpy, com a intenção de perceber se houve avanços significativos nesta nova versão. Este questionário é sustentado pelo conjunto das heurísticas de avaliação de usabilidade apresentado na secção anterior. Consiste num sistema de resposta em escolha múltipla, estando estas representadas num gráfico de percentagens na Figura 5.1. As respostas recolhidas foram processadas da seguinte forma:

- Para cada grupo de perguntas as respostas possíveis eram: **Não se Aplica - Nunca - Quase Nunca - Regular - Quase sempre - Sempre.**
- O número total das respostas válidas é calculado subtraindo-lhe às que foram respondidas as **Não se Aplica.**
- Todas as outras opções de resposta são consideradas válidas para avaliação.

Ao analisar o gráfico obtido pelo questionário torna-se claro que as principais debilidades do sistema são ao nível da eficiência e do controlo do utilizador sobre a aplicação. Os comentários mais negativos feitos pelos avaliadores sobre a eficiência do sistema são *"a inexistência de teclas de atalho para executar as funções mais usadas"* e *"o sistema não permite desativar temporariamente algumas das funções"*. Relativamente ao controlo do utilizador, os aspetos fracos referidos pelos avaliadores foram a *"não*

possibilidade de cancelar uma operação a decorrer" e *"não há hipótese de eliminar qualquer alteração que tenha sido realizada e voltar ao estado anterior"*. Como se pode constatar nas restantes heurísticas, os comentários obtidos são muito positivos, destacando-se entre elas a confiança na aplicação, a facilidade de aprendizagem, a estética e o design minimalista, reconhecimento ao invés de lembrar, consistência e padrões e compatibilidade.

A resposta referente à avaliação global do sistema *"considerando todos os parâmetros que analisou, como classificaria o HandSpy 2.0?"* foi unânime: todos os avaliadores o classificaram como sendo *"bom"*. Isto revela que esta nova versão do sistema tem potencial para se afirmar a nível científico.

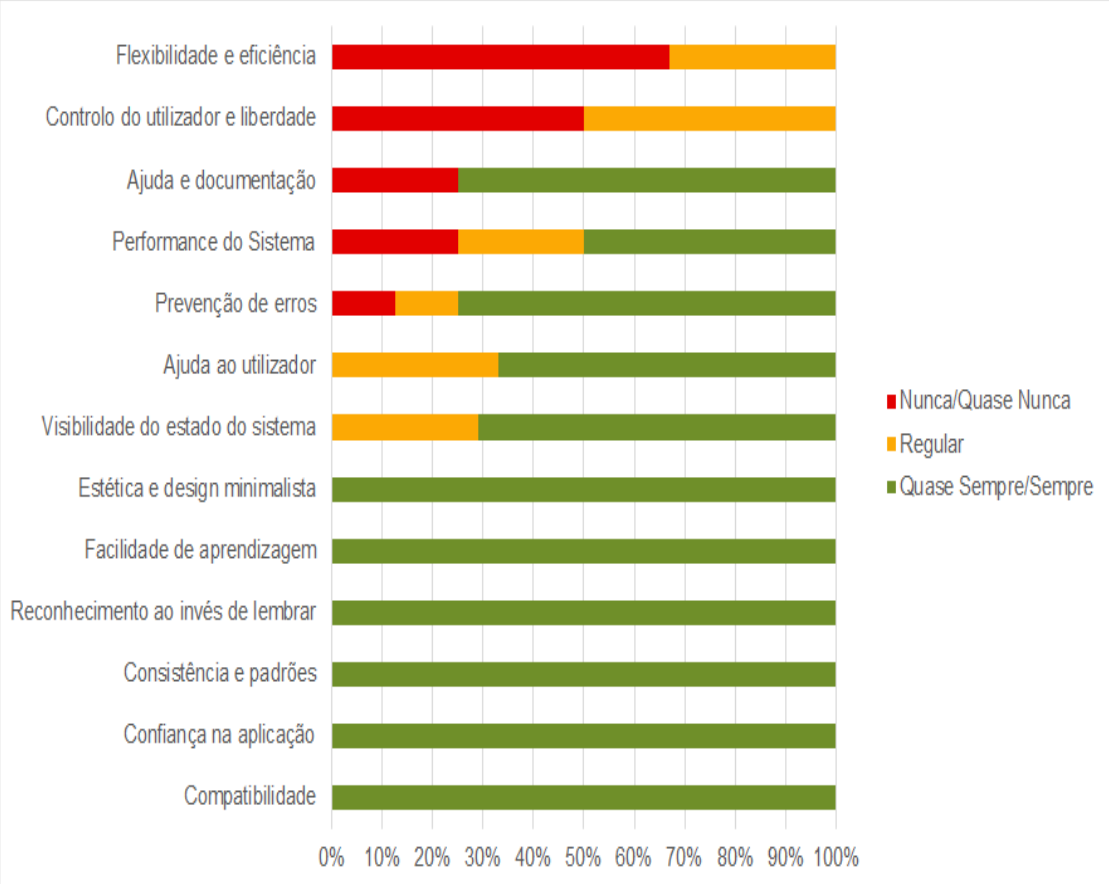

Figura 5.1: Resultados da avaliação heurística do HandSpy 2.0

5.3 Comparação da avaliação heurística entre as versões do HandSpy

Com a finalidade de revelar as melhorias existentes na nova versão do HandSpy podemos comparar os gráficos resultantes dos questionários de usabilidade feitos aos avaliadores sobre cada versão. Na Figura 5.1 e 5.2 estão ilustrados os gráficos obtidos. Ao observá-los é evidente uma melhoria significativa da versão 2.0 relativamente à 1.0. Os pontos fracos visíveis no HandSpy 1.0, como a ajuda e documentação, facilidade de aprendizagem, ajuda ao utilizador e confiança na aplicação sofreram uma grande evolução positiva. No entanto, ao nível da flexibilidade e eficiência essa evolução foi pouco significativa continuando a ser um ponto fraco neste sistema. Verificou-se um aperfeiçoamento em todos os restantes aspetos, logo podemos inferir que o HandSpy 2.0 é uma aplicação com muito mais potencial que o seu antecessor.

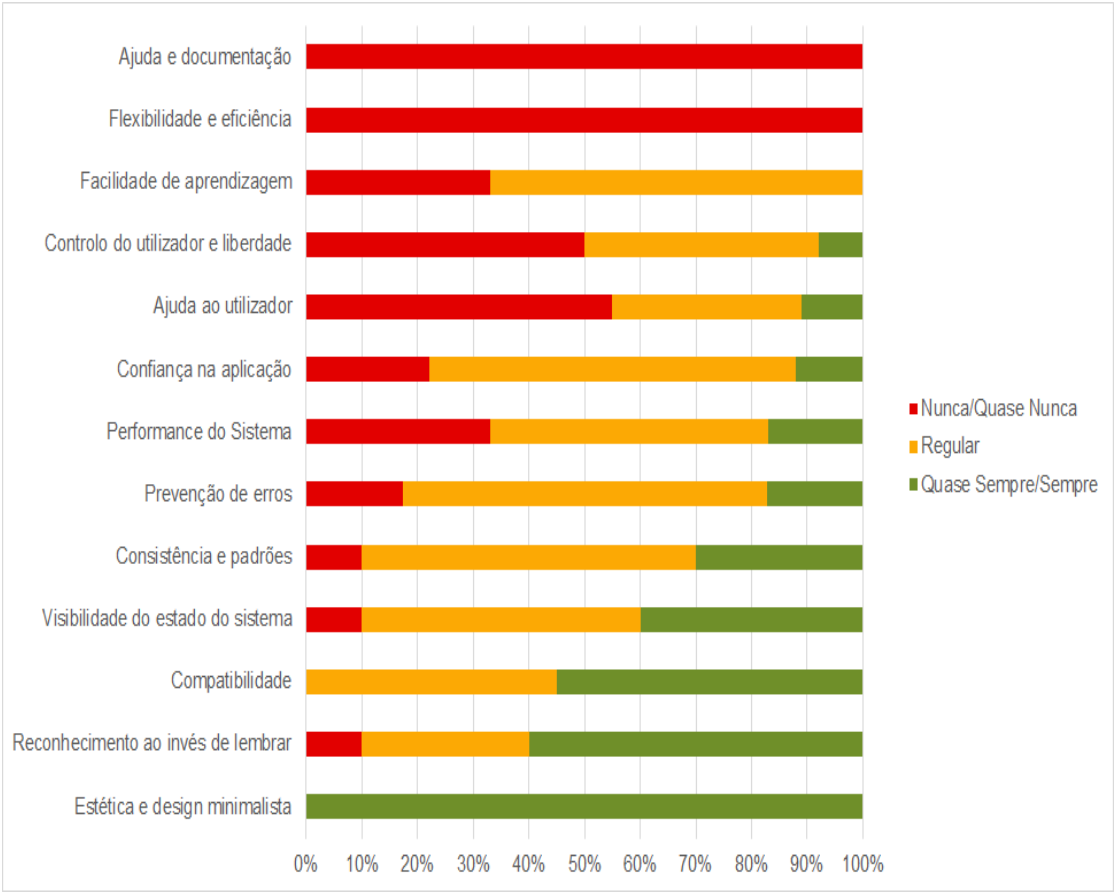

Figura 5.2: Resultados da avaliação heurística do HandSpy 1.0

Capítulo 6

Conclusão

O estudo dos processos cognitivos da escrita tem vindo a ter uma grande evolução com o uso de novos dispositivos de recolha. Esta dissertação teve como objetivo descrever a conceção, implementação e avaliação de uma aplicação web, o HandSpy 2.0. Este sistema tem como finalidade a gestão e apoio de estudos de investigação de escrita com grandes quantidades de dados, permitindo assim acelerar o processo de análise dos mesmos. Outro fundamento para a sua conceção foi a necessidade de colmatar algumas falhas do seu antecessor, o HandSpy 1.0, e acrescentar novos componentes na análise dos dados recolhidos, aumentado assim a robustez e eficiência da aplicação e tornando-a cada vez mais versátil e contemporânea. Todo este processo de criação e migração para nova versão foi acompanhado por vários investigadores do projeto DAAR para uma maior perceção das suas necessidades e dificuldades durante o uso e adaptação da aplicação, criando-se assim uma aplicação mais à imagem do idealizado. É de referir que esta nova versão já se encontra em uso por investigadores de vários projetos e tem tido um *feedback* muito positivo.

6.1 Trabalho Futuro

Como trabalho futuro, a evolução do HanSpy deverá consistir em alguns melhoramentos no interface e introdução de novas formas de dados para estudo. Com base nos resultados dos questionário realizado o caminho a seguir deve-se focar em melhorar algumas falhas encontradas nas respostas e sugestões dos investigadores. Na lista

seguinte são apresentadas as principais propostas futuras a implementar no HandSpy.

- Melhoramento de algumas lacunas existentes a nível do interface.
- Introduzir novas formas de dados para estudo, como por exemplo som e ritmo cardíaco.
- Incorporar a criação dos cadernos na aplicação Web.
- Aperfeiçoar a interação dos vídeos com a tabela de dados.
- Aumentar os níveis de flexibilidade e eficiência da plataforma web.
- Reconhecimento de carateres pelo HandSpy.

Referências

[1] Creating XMLType Tables and Columns Based on XML Schema, 2005. http://goo.gl/bSsuj.

[2] BLOBs and CLOBs, 2012. http://goo.gl/i0ddz.

[3] Smart GWT, 2012. http:/http://smartclient.com/.

[4] About FFmpeg, 2013. http://www.ffmpeg.org.

[5] eXist-db Open Source Native XML Database, 2013. http://exist-db.org.

[6] HTML5 Video, 2013. http://www.w3schools.com/html/.

[7] Media formats supported by the HTML audio and video elements, 2013. https://developer.mozilla.org/en-US/docs/HTML/Supported_media_formats.

[8] Server Communication, 2013. http://http://www.gwtproject.org/.

[9] J. Gerritsen. Native XML Databases. *5th Twente Student Conference on IT*, 2006.

[10] Robert Hanson and Adam Tacy. *GWT in Action*. Manning Publications Co., Greenwich, CT, 2007.

[11] Yannis E. Ioannidis and Miron Livny. Conceptual schemas: Multi-faceted tools for desktop scientific experiment management. *Journal of Intelligent and Cooperative Information Systems*, 1:451–474, 1992.

[12] D. Florescu M. Stefanescu J. Robie James Clark, D. Chamberlin and J. Simeon. XQuery 1.0: An XML Query Language, 2001. http://www.w3.org/TR/xquery/.

[13] João Manuel Brisson Lopes. Formatos de imagem. Technical report, Instituto Superior Técnico, Universidade Técnica de Lisboa, 2008.

[14] J. Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, CHI '94, pages 152–158, New York, NY, USA, 1994. ACM.

[15] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90, pages 249–256, New York, NY, USA, 1990. ACM.

[16] Greg Roelofs. *PNG The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, 1999.

[17] Isomorphic Software. *Smart GWT - Quick Start Guide*. Isomorphic Software, Inc., San Francisco, CA, 2012.

[18] Bram Smeets Uri Boness and Roald Bankras. *Beginning Google Web Toolkit*. Apress, Berkeley, CA, 2008.

Apêndice A

Acrónimos

AFD	Anoto Functionality Document
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CLOB	Character Large Object
DAAR	Develop Automate and Auto Regulating
DBMS	Database Managment Systems
IDE	Integrated Development Environment
XQJ	XQuery Java API
XML	eXtensible Markup Language
GWT	Google Web Toolkit
JSNI	JavaScript Native Interface
HTML	HyperText Markup Language
RPC	Remote Procedure Calls
HTTP	Hypertext Transfer Protocol
PNG	Portable Network Graphics
GIF	Graphics Interchange Format
CVS	Comma-Separated Values
JPEG	Joint Photographic Experts Group
SQL	Structured Query Language
FLWOR	For-Let-Where-Order-Return
REST	Representational State Transfert

Apêndice B

HandSpy 2.0 Usability Questionnaire

For the next questions give one of the following answers

Never		Almost never	
Regular		Almost always	
Always		Does not apply	

Visibility

1. When I ask the system for help the answer is clear
2. When I perform a task the system informs me about what is happening
3. The buttons I use to perform the most important tasks are clearly identified
4. The buttons status (selected/unselected) is clearly shown

Compatibility

1. When I try to perform a task I quickly find the intended button
2. The buttons order is in a familiar sequence
3. When I select a button the result is what I expected
4. The functions of a list opened by a button belong all to the same category of that button

User control and freedom

1. When I make a mistake the system allows me to undo it
2. I can interrupt an action and resume it whenever I wish
3. I can cancel an ongoing operation
4. I can eliminate any change I'm performing and return to the previous state

Consistency and Standards

1. The buttons and windows location is maintained when I swap screens
2. The buttons maintain the same meaning when I change screens
3. The color code meaning is consistent
4. The scroll function can be used in all windows

Error

1. The system warns me when data entry problems occur before I execute the validation
2. When the validation occurs the system produces an error message if the data format is not the expected one
3. The system warns me if I am about to make a serious mistake
4. There is a clear separation between the buttons that enables the possibility of serious mistakes to happen and all the others buttons

Recognition

1. The colors used in the texts are accordingly with the accepted conventions for their meanings
2. The text inside each button transmits the idea of what is expected to happen when I use them
3. The information is on the part of the screen I expect in to be
4. The features are grouped by kinds in individualized logical zones

Flexibility

1. I am allowed to configure the screen setup

2. There are shortcut keys to execute the most used functions
3. The system allows me to temporarily deactivate some of the functions

Aesthetics

1. The information on the screen is the strictly necessary for me
2. The information on screen detaches itself from the background
3. Aesthetically the system is pleasant in the factors color, brightness, etc

Users Aid

1. The error/help messages are clear and adequate
2. The errors messages state the problem with precision
3. The messages are short and objective

Help and Documentation

1. I can easily perform information searches
2. The help function is easily visible
3. The information is precise complete and perceptible

Ease of Learning

1. The system is intuitive (I understand it easily)
2. It is easy for me to learn how to work with system
3. I don't need help to work with the system

System performance

1. The response time for the executed operations is fast enough
2. The response time when changing working tab is fast enough

Reliability

1. register on the system
2. choose the working project
3. manage tasks

4. upload data (InkML files) to the system
5. know the state of a protocol (linked to a participant or not)
6. manage participants list
7. create a selection
8. get calculations on a protocol
9. generate new calculations
10. export selected calculations

Rating Considering all the parameters that you analyzed how would you rank HandSpy?

Very Good	
Good	
Merely Adequate	
Inadequate	
Bad	