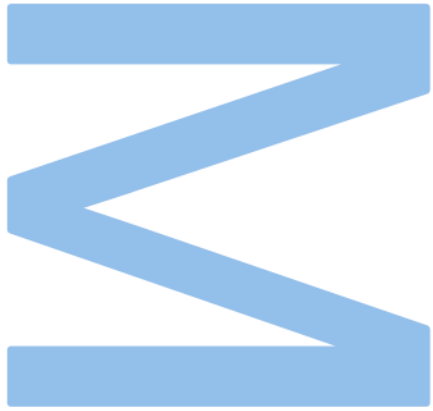
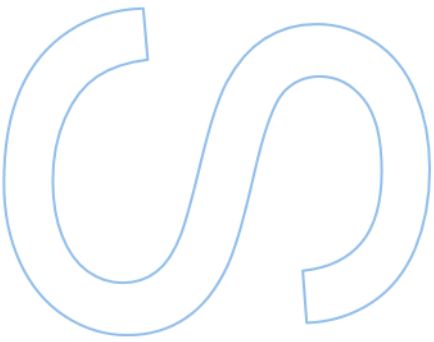
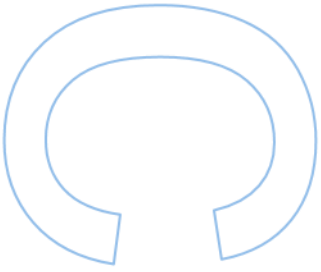
Gestão de Dados em Cidades Inteligentes







Ana Beatriz Martins Saldanha
Mestrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2022

Orientador
João Miguel Maia Soares de Resende, Professor Auxiliar
Departamento de Informática
Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa

Coorientador
Patrícia Raquel Vieira Sousa, Investigadora
Centro de Cibersegurança e Privacidade
Universidade do Porto


U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Declaração de Honra

Eu, Ana Beatriz Martins Saldanha, inscrita no Mestrado em Engenharia de Redes e Sistemas Informáticos da Faculdade de Ciências da Universidade do Porto declaro, nos termos do disposto na alínea a) do artigo 14.º do Código Ético de Conduta Académica da U.Porto, que o conteúdo da presente dissertação de projeto reflete as perspetivas, o trabalho de investigação e as minhas interpretações no momento da sua entrega.

Ao entregar esta dissertação de projeto, declaro, ainda, que a mesma é resultado do meu próprio trabalho de investigação e contém contributos que não foram utilizados previamente noutros trabalhos apresentados a esta ou outra instituição.

Mais declaro que todas as referências a outros autores respeitam escrupulosamente as regras da atribuição, encontrando-se devidamente citadas no corpo do texto e identificadas na secção de referências bibliográficas. Não são divulgados na presente dissertação de projeto quaisquer conteúdos cuja reprodução esteja vedada por direitos de autor.

Tenho consciência de que a prática de plágio e auto-plágio constitui um ilícito académico.

Ana Beatriz Martins Saldanha

16 de novembro de 2022

Abstract

Nowadays, the term *Internet of Things* (IoT) has been the subject of much attention and research. Consumer goods, industrial components, cars, sensors and other objects in our daily lives are being connected to the Internet and strong data analysis capabilities that are guaranteed to improve the way we live [83]. In recent years, there has been a dramatic growth of IoT devices in the market. According to IoT Analytics, a leading provider of market insights and strategic business intelligence for the IoT, the percentage of these connected devices has risen by 18% to more than 14.4 billion devices [39]. Given the existence of so many devices and the connectivity between them, there is a problem associated with the ability of users of these devices to understand what kind of data they are exchanging, and whether this exchange is being done in a secure manner.

The idea of this dissertation focused, first, on the study the available state of the art and the collection of information on techniques that would enable data management on data trading platforms. After that, the design of a data marketplace interface and implementation of solutions for data lifecycle management, authentication and access control. This, in order to allow users to sell and manage information shared with the smart cities.

After the development of the data marketplace was completed, this application was presented to 10 users for testing purposes, in order to evaluate the implemented features and their usability.

Resumo

Nos dias de hoje, o termo *Internet of Things* (IoT) tem sido alvo de grande atenção e investigação. Bens de consumo, componentes industriais, carros, sensores e outros objetos do nosso quotidiano estão a ser ligados à Internet e a fortes capacidades de análise de dados que garantem melhorar o nosso dia a dia na forma como vivemos [83]. Nos últimos anos, tem havido um crescimento drástico do número de dispositivos IoT no mercado. De acordo com a IoT Analytics, líder fornecedor de estudos de mercado e inteligência empresarial estratégica para a IoT, a percentagem destes dispositivos ligados aumentou 18%, para um total de mais de 14,4 mil milhões de dispositivos [39]. Dada a existência de tantos dispositivos e a conetividade entre eles, existe um problema associado à capacidade dos utilizadores destes dispositivos em compreenderem que tipo de dados são trocados, e se esta troca está a ser feita de uma forma segura.

A ideia desta dissertação centrou-se, em primeiro lugar, no estudo do estado da arte disponível e na recolha de informações sobre técnicas que permitam a gestão de dados em plataformas de comercialização de dados. Posteriormente, a conceção de uma interface de um *data marketplace* e implementação de soluções para a gestão do ciclo de vida dos dados, autenticação e controlo de acesso. Tudo isto, a fim de permitir aos utilizadores a gestão das suas informações e partilha das mesmas com as *smart cities*.

Após a conclusão do desenvolvimento do *data marketplace*, esta aplicação foi apresentada a 10 utilizadores para efeitos de teste, a fim de avaliar as características implementadas e a sua usabilidade.

Agradecimentos

Em primeiro lugar, quero agradecer ao meu orientador, Professor João Miguel Maia Soares de Resende e a minha coorientadora, Patrícia Raquel Vieira Sousa, pela disponibilidade permanente e por me fazerem acreditar que era capaz de levar a cabo este trabalho.

Ao Tiago, pela paciência, pelo amor e por toda a força nos momentos mais difíceis.

Aos meus pais e aos meus irmãos um agradecimento do tamanho do mundo, por todo o apoio e incentivo. Sem eles nada disto teria sido possível.

A todos os meus amigos que este percurso académico me trouxe e que vou levar para sempre no coração.

Aos meus pais e aos meus irmãos

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Blocos de Código

Acrónimos

ABAC *Attribute Based Access Control*

ATM *Automated Teller Machine*

ACL *Access Control List*

AMQP *Advanced Message Queuing Protocol*

API *Application Programming Interface*

BSON *Binary JavaScript Object Notation*

CO2 Dióxido de Carbono

CSV *Comma Separated Value*

DAC *Discretionary Access Control*

DLT *Distributed Ledger Technology*

GPS *Global Positioning System*

HTTP *Hypertext Transfer Protocol*

IAM *Identity and Access Management*

IoT *Internet of Things*

IP *Identity Provider*

IP *Internet Protocol*

JWT *JSON Web Token*

JSON *JavaScript Object Notation*

KML *Keyhole Markup Language*

LED *Light Emitting Diode*

MAC *Mandatory Access Control*

MAM *Masked Authenticated Messaging*

MQTT *Message Queuing Telemetry Transport*

NoSQL *Not Only SQL*

NO2 Dióxido de Nitrogénio

OTP *One-Time Password*

PIN *Personal Identification Number*

RBAC *Role Based Access Control*

REST *Representational State Transfer*

SAML *Security Assertion Markup Language*

SDS *Smart Data Streaming*

SSO *Single Sign-On*

URI *Uniform Resource Identifier*

URL *Uniform Resource Locator*

XML *Extensible Markup Language*

FCUP *Faculdade de Ciências da Universidade do Porto*

Capítulo 1

Introdução

Internet of Things (IoT) é um tópico em ascensão nos dias de hoje, tanto a nível económico, social, como tecnológico [83]. Este, é um conceito no qual dispositivos do nosso quotidiano, como eletrodomésticos, relógios ou até roupas estão ligados à Internet e atuam de forma inteligente e sensorial [91]. O termo *"things"* refere-se aos sensores e atuadores, embutidos em objetos físicos que comunicam e trocam informações entre si, através da Internet [62]. O número de dispositivos IoT tem vindo a apresentar um aumento drástico ao longo dos anos. Segundo [83], estima-se que no ano de 2025 o número destes dispositivos será aproximadamente 100 biliões com um valor económico de mais de 11 triliões de dólares. A implementação em larga escala de dispositivos IoT, visa transformar e melhorar a forma como vivemos [83]. Estes dispositivos estão presentes em diversos domínios e um dos principais são as *smart cities*.

Ao longo dos anos, tem sido observado um aumento na população mundial que reside em áreas urbanas. Dados das Nações Unidas [74] apontam que, em 2014, 54% da população mundial estava concentrada em áreas urbanas, sendo expectável que esse valor suba para 66% em 2050. O exponencial aumento da população e urbanização tem causado alguns problemas nas cidades modernas. Alguns exemplos disso são o congestionamento do tráfego, alterações ambientais e o consumo de energia [3]. Assim, é fundamental que as cidades forneçam serviços e recursos que assegurem o bem estar das suas populações, ou melhor dizendo, que se tornem cidades "inteligentes", potenciando a qualidade dos serviços oferecidos aos cidadãos [101]. É aqui que a IoT exerce um papel importante ao possibilitar que inúmeros dispositivos sejam instalados nas cidades para que possam recolher e trocar informações continuamente [80]. Estas informações podem ser utilizadas no melhoramento de serviços como controlo de lugares em parques de estacionamento, iluminação inteligente e de adaptação climática nas luzes da rua, ou até no reconhecimento dos níveis de resíduos em contentores para otimizar as rotas de recolha do lixo [8].

Com o grande aumento das fontes de dados (dispositivos IoT) ao longo do tempo, as empresas procuram formas de aproveitar os dados a fim de obterem novos conhecimentos para que possam melhorar os serviços prestados. Um exemplo disso, remete a um fornecedor de aplicações relacionadas com aptidão física, querer obter dados relativos à qualidade do ar provenientes de estações meteorológicas espalhadas pela cidade, para poderem sugerir pistas de corrida sem

poluição aos seus utilizadores. Convém salientar que estes dispositivos são propriedade de uma gama diversificada de fornecedores como indivíduos, organizações e indústrias [38].

Da mesma forma que os consumidores dos dados beneficiam a partir da utilização destes, também os próprios proprietários dos dados recebem incentivos pela partilha dos mesmos. Para que esta economia em torno dos dados seja possível, é necessário a existência de plataformas que permitam a sua negociação e troca. Assim, plataformas de comercialização de dados IoT, também denominados IoT *data marketplaces*, estão a ser desenvolvidos para ajudar cidades e comunidades na criação de inúmeras aplicações. Estes *data marketplaces* permitem que os proprietários de dispositivos IoT vendam os seus dados aos desenvolvedores de aplicações e outros clientes interessados [70].

Para Mišura et al. [63] a criação de *data marketplaces* para IoT traz diversos benefícios como a poupança de recursos por parte dos consumidores dos dados, uma vez que não necessitam de instalar novos dispositivos para determinadas aplicações. Estas podem ser criadas rapidamente pelo facto de não haver desperdício de tempo na instalação de dispositivos e também por não ser necessária a negociação diretamente com os proprietários dos dispositivos. Além disso, há uma maior flexibilidade, uma vez que os consumidores podem adquirir apenas os dados de que necessitam. Os consumidores de dados podem obter dados, que os proprietários estão dispostos a partilhar para lucrarem a partir destes.

Apesar dos benefícios que estas plataformas proporcionam, é necessário ter em consideração certos desafios associados à segurança e privacidade de todos os participantes, especialmente dos proprietários dos dados. Estes devem poder controlar os seus dados, ou seja, devem poder escolher que parte dos dados são partilhados, com quem e, em que circunstâncias [10]. Muitas das vezes os dados provenientes dos dispositivos podem estar associados a dados pessoais, que de alguma maneira possam comprometer a segurança e privacidade dos proprietários. Assim, técnicas de anonimização dos dados são essenciais antes destes serem comercializados, para que informações que possam levar à identificação pessoal sejam ocultadas [87]. A transparência na utilização dos dados é outro ponto fundamental no desenvolvimento de um *data marketplace* pois, o proprietário dos dados tem o direito de saber como e quando são utilizados, e aplicar restrições de utilização sobre eles. Além disso, devem ter garantias de que os seus dados são partilhados em conformidade com as políticas por eles estipuladas [10]. Mecanismos de autenticação tanto dos utilizadores como dos dispositivos e controlo de acesso são fundamentais para restringir o acesso a entidades não autorizadas e também para definir diferentes níveis de acesso aos dados conforme as preferências dos proprietários [78].

Nesta dissertação, desenvolvemos um *data marketplace* para a comercialização de dados provenientes de dispositivos IoT no contexto das *smart cities*, tendo em conta a superação de todos os desafios anteriormente descritos.

1.1 Objetivos

O principal objetivo desta dissertação é, portanto, o desenvolvimento de um *data marketplace* num ambiente seguro, confiável e transparente, na qual fornecedores de dados os disponibilizem para que compradores interessados os possam adquirir. Assim, a dissertação foca-se também nos seguintes objetivos:

- Identificar caraterísticas chave para o desenvolvimento de um *data marketplace*, bem como o estudo de diferentes plataformas existentes.
- Desenvolvimento de um *data marketplace* no contexto de *smart cities*.
- Desenvolvimento de um *data marketplace* com gestão do ciclo de vida dos dados e políticas de autenticação e controlo de acesso.
- Integração com um simulador de dados de IoT.

1.2 Estrutura da Dissertação

Esta dissertação é constituído por 5 capítulos, distribuídos da seguinte forma:

- **Capítulo 2 - Conceitos Fundamentais:** Neste capítulo são descritos todos os conceitos fulcrais para uma melhor compreensão do projeto desenvolvido.
- **Capítulo 3 - Estado da Arte:** Neste capítulo é apresentada a pesquisa realizada sobre *data marketplaces* existentes na atualidade. É feita uma abordagem a caraterísticas fundamentais para o desenvolvimento de um *data marketplace* e, por fim, apresentamos quais as caraterísticas presentes em cada uma destas plataformas resultantes da pesquisa. Ainda neste capítulo são analisados simuladores de dados IoT para integração com o sistema.
- **Capítulo 4 - Desenho e Desenvolvimento:** Neste capítulo é descrito todo o desenvolvimento do nosso *data marketplace*. Primeiramente é apresentada a arquitetura geral e teórica, seguida da descrição de todos os componentes bem como a sua implementação.
- **Capítulo 5 - Resultados e Análise:** Neste capítulo são apresentados e discutidos os resultados do questionário desenvolvido para avaliação das funcionalidades e usabilidade da nossa aplicação.
- **Capítulo 6 - Conclusões:** Neste capítulo são apresentadas as conclusões finais e algumas sugestões de trabalho futuro a serem consideradas.

Capítulo 2

Conceitos Fundamentais

Neste capítulo são abordados os conceitos fundamentais para a compreensão e elaboração desta dissertação. Foi feita uma abordagem ao conceito *smart cities*, às diferentes definições de *data marketplace* propostas por diferentes autores e o ecossistema de um *data marketplace*. São abordados os conceitos de sistema de mensagens *publish-subscribe* e *web services*. São também apresentadas as etapas que constituem o ciclo de vida dos dados propostas por diferentes autores. Além disto, apresentado o conceito de gestão de identidade e acesso, assim como conceitos relacionados a este, tais como autenticação e autorização. Por fim, são apresentados protocolos utilizados em processos de autenticação e autorização, bem como diferentes modelos de controlo de acesso.

2.1 *Smart Cities*

Com o aumento da população nas áreas urbanas, as cidades procuram formas de fornecer recursos e serviços, como por exemplo, água potável, energia, sistemas de esgotos, educação e serviços de saúde. Assim, estas cidades tornam-se cidades mais "inteligentes"ao usufruírem de todo o potencial da tecnologia e dos dados para prestarem os serviços essenciais aos cidadãos, a fim de aumentarem a sua qualidade de vida de forma eficiente com garantias de sustentabilidade económica, social e ambiental [56]. Posto isto, é essencial a recolha de informações para a gestão diária das atividades e planeamento do desenvolvimento a longo prazo nas cidades. Por exemplo, informações sobre transportes públicos, como localização e utilização em tempo real, ocupação de lugares de estacionamento, engarrafamentos, e outros dados como condições meteorológicas, estado de poluição atmosférica e sonora, contaminação da água, consumo de energia, devem ser recolhidos continuamente. Para tal, inúmeros dispositivos como sensores, atuadores, *smartphones* e outros aparelhos inteligentes, espalhados por toda a área urbana interligados através da *internet*, recolhem informações continuamente - *Internet of Things* (IoT) [95].

2.1.1 Sensores nas *Smart Cities*

Para cada serviço ou aplicação que se pretenda melhorar ou desenvolver, dependendo do domínio em que se enquadram, são necessários diferentes dispositivos IoT. Assim estes fazem parte de uma gama bastante heterogénea. Os dispositivos IoT controlam vastas áreas geográficas e detetam eventos e condições em tempo real do meio em que se encontram. A sua utilização é bastante diversificada, sendo que pode ser classificada em seis grupos distintos: saúde, mobilidade, água, energia, gestão do desperdício, segurança. Assim, numa *smart city* existem inúmeros dispositivos como por exemplo *smartphones*, *smartwatches*, medidores de glucose inteligente sem fio na monitorização de saúde [79]. Sensores de proximidade ultrassónicos e de medição da distância em parques de estacionamento inteligentes. Estes sistemas visam acabar com os problemas existentes nos parques de estacionamento. Controlam a chegada e a partida dos automóveis, indicando também a zona de estacionamento e o espaçamento exato do veículo a ser estacionado. Em sistemas de gestão do desperdício, sensores de nível, medem a quantidade de lixo. Quando é atingido o valor estipulado, estas informações são enviadas para o *smartphone* do condutor do camião de recolha. Sensores de temperatura, humidade, água, Dióxido de Carbono (CO2), Dióxido de Nitrogénio (NO2) são utilizados na monitorização ambiental inteligente permitindo, por exemplo, controlar a poluição no ar, como também a pureza e a disponibilidade de água [45].

2.1.2 *Smart Cities* no Mundo

Segundo a Juniper Research, uma empresa de estudos de mercado, desenvolveu uma pesquisa com o intuito de determinar as cinco cidades mais inteligentes no mundo. Para o desenvolvimento desta pesquisa foram avaliadas diferentes caraterísticas de uma *smart city* como transportes e infraestruturas, energia e iluminação, gestão e tecnologia da cidade e conetividade urbana. Os resultados da pesquisa apresentam a cidade de Shanghai no 1º lugar como a cidade mais inteligente no ano de 2022, seguida de Seoul, Barcelona, Beijing e New York [81]. De seguida iremos apresentar projetos desenvolvidos em cada cidade.

Shanghai desenvolveu uma plataforma denominada Citizen Cloud, que oferece mais de 1200 serviços públicos aos cidadãos. Estes serviços abrangem nascimento, casamento, cultura, educação, turismo, segurança social, transportes, tratamento médico e saúde serviços jurídicos, e cuidados seniores. Esta plataforma é a melhor forma de contactar as entidades governamentais da cidade possui também um único contacto telefónico para que os cidadãos não necessitem de procurar o número do departamento pretendido [5].

Seoul, capital da Coreia do Sul, desenvolveu o projeto Smart City in Daily Life pilot. Este projeto visa a proteção das crianças, prevenindo que estas desapareçam, através de *beacons* e *robots* de patrulha noturna [72].

Barcelona possui uma distribuidora metropolitana de propriedade pública, que fornece energia 100% renovável para iluminação pública, semáforos, habitações públicas, edifícios municipais e centros desportivos. A iluminação das ruas é feita com luzes *Light Emitting*

Diode (LED) [36].

Em **Beijing**, capital da China, todos os transportes públicos podem ser pagos através do *smartphone*. Nesta cidade, todos os cidadãos são portadores de um cartão virtual responsável pela gestão dos documentos de identificação [44].

New York, desenvolveu um sistema automático de leitura dos contadores, através de todos os contadores de água da cidade. Estes são equipados com dispositivos sem fios que enviam leitura da água para um sistema de faturação até quatro vezes por dia. Este sistema permite que através da conta MY DEP Account, os clientes consigam visualizar e gerir os seus consumos diários, semanais, mensais e anuais [77].

2.2 *Data Marketplaces*

A rápida evolução da tecnologia e a presença de ambientes ricos em dados, originou a necessidade de se criarem locais específicos para o comércio de dados, onde estes podem ser fornecidos e adquiridos. Um *data marketplace* pode ser visto como um local onde fornecedores e consumidores se encontram para trocarem bens [92]. A popularidade dos *data marketplaces* tem crescido ao longo dos anos, devido à necessidade de se obter a informação essencial quando é realmente necessária, em praticamente todas as áreas de negócio [93]. Estes, permitem que as empresas explorem novas oportunidades de receita, através da venda dos seus dados internos e pela aquisição de dados externos, de forma a melhorar os seus serviços e alcançarem os seus objetivos [35]. Os *data marketplaces* podem desempenhar um papel fundamental na economia, permitindo o processo comercial de dados entre organizações.

2.2.1 Definições de *Data Marketpalce*

Não existe uma definição unânime para este termo, no entanto, existem várias definições propostas. Schomm et al. [86] definem um *data marketplace* como uma plataforma na qual, qualquer pessoa, ou pelo menos um grande número de clientes possivelmente registados, podem inserir e manter conjuntos de dados. Uma outra definição, proposta por Spiekermann et al. [92], menciona um *data marketplace* como sendo uma plataforma digital, onde produtos de dados são comercializados. Segundo Koutroumpis et al. [50] um *data marketplace*, assemelha-se a uma plataforma multilateral, onde as interações entre participantes é realizada por meio de um intermediário digital. Para Lawrenz et al. [57] um *data marketplace* é uma plataforma que permite a compra e venda de produtos tal como qualquer outro *marketplace online*. No entanto, no caso específico dos *data markeplaces* o produto negociado são dados.

2.2.2 Ecossistema de um *Data Marketplace*

Relativamente ao ecossistema de um *data marketplace*, os principais participantes são vendedores, compradores, fornecedores de serviços adicionais e o proprietário do *data marketplace*. Os primeiros são entidades, que tanto podem ser empresas como indivíduos que colocam os dados disponíveis no *data marketplace*, esperando assim obter lucros com a venda destes. Os compradores de dados correspondem aos participantes do *data marketplace* que estão interessados em adquirir os conjuntos de dados de que necessitam para processos de tomada de decisão, desenvolver novos serviços, e pelos quais estão dispostos a pagar. A intervenção dos fornecedores de serviços adicionais está relacionada com a disposição de aplicações, algoritmos que acrescem valor aos dados e facilitam o acesso e a sua utilização pelos compradores. Por último, o proprietário do *data marketplace* recolhe, armazena os dados provenientes do fornecedor e torna-os disponíveis para o comprador [35, 92]. Por vezes, é o próprio proprietário que recorre à venda de dados, assumindo assim também o papel de vendedor [86].

2.3 Sistema de Mensagens *publish-subscribe*

Sucintamente, um sistema de mensagens com um padrão *publish-subscribe* permite o envio de mensagens para todas as partes que demonstrem interesse nestas. A ação de registar o interesse é designada por subscrição, pelo que o interessado é o *subscriber*. O elemento que fornece as mensagens é denominado de *publisher* e fá-lo através da publicação das mesmas. Por fim, o elemento responsável pela entrega das mensagens enviadas pelos *publishers* para os *subscribers* é designado por *broker*. Os três principais tipos de sistemas de mensagens *publish-subscribe* são: sistemas baseados em conteúdos, sistemas baseados em tipos e sistemas baseados em tópicos [42].

- **Sistemas baseados em tipos:** O *subscriber* refere o tipo de informação em que está interessado.
- **Sistemas baseados em conteúdos:** O *susbcriber* descreve o conteúdo que deseja receber na mensagem. Por exemplo, pode subscrever as mensagens que contenham leituras de temperatura inferiores a um limiar definido.
- **Sistemas baseados em tópicos:** Nestes sistemas, as mensagens são publicadas em tópicos, nos quais os *subscribers*subscrevem aqueles em que possuem interesse, a fim de receber as mensagens deste.

No nosso projeto, iremos utilizar um sistemas de mensagens *publish-subscribe* baseado em tópicos. A Fig 2.1 mostra o funcionamento deste tipo de sistemas, com dois *publishers*, um tópico para o qual são enviadas as mensagens e três *subscribers* que recebem as mensagens do tópico.

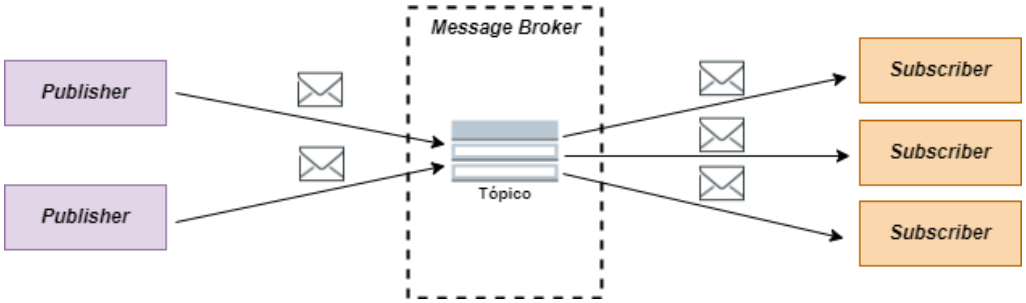


Figura 2.1: Exemplo de um sistema de mensagens do tipo *publish-subscribe*

2.4 *Web Services*

A maioria das aplicações *web*, dependem destes serviços para integração entre cliente e servidor. Estes são componentes de *software* baseados em protocolos e padrões que permitem a comunicação através da rede entre aplicações ou sistemas. Os *web services* são uma peça fundamental na integração dos sistemas e aplicações, uma vez que estes podem ser baseados em diferentes linguagens, tecnologias e plataformas [98]. É preciso ter em atenção de que os *web services* não fornecem a interface para o utilizador, mas sim o serviço que vai ser consumido através desta. Esta interface é denominada *Web Application Programming Interface* (API). Podemos dizer que uma *web* API, representa um *web service* à espera de pedidos vindos do cliente e a responder a estes [60]. Nos dias de hoje, um dos *web services* mais utilizado é *Representational State Transfer* (REST).

REST é um tipo de arquitetura que facilita a comunicação entre sistemas a partir da *web*. As interfaces REST baseiam-se unicamente em *Uniform Resource Identifier* (URI)'s para o acesso aos recursos e no protocolo *Hypertext Transfer Protocol* (HTTP) para a comunicação entre cliente-servidor, ou seja, para o envio de mensagens entre as duas partes [71]. O protocolo HTTP possui quatro métodos básicos: GET para obter um determinado recurso ou um conjunto de recursos, POST para criar um recurso novo, PUT para atualizar um recurso e DELETE para apagar um recurso. Para que estes possam ser acedidos, o pedido feito pelo cliente deve conter um caminho (*endpoint*) para o recurso pretendido. O *endpoint* consiste num URI que funciona como um identificador para o recurso. As respostas enviadas pelo servidor podem ser em formato *Extensible Markup Language* (XML) ou *JavaScript Object Notation* (JSON) [60].

É caraterizado por separar as preocupações do cliente e do servidor e por ser *stateless*. O cliente e o servidor são considerados duas partes independentes, isto é, são implementados e desenvolvidos independentemente, através de qualquer tecnologia ou linguagem. Caso haja alguma alteração no código de qualquer umas das partes, a outra nunca será afetada [60]. Este princípio permite a escalabilidade de cada um dos lados de forma independente. Outro princípio importante já mencionado é a *statelessness*. Um sistema com uma arquitetura REST é considerado *stateless* uma vez que cada pedido HTTP decorre de forma isolada. O pedido HTTP feito pelo cliente, tem de incluir as informações necessárias para que o servidor o consiga

realizar pois, este nunca se baseia em informações de pedidos anteriores [82].

2.5 Transformação dos Dados

A transformação dos dados é algo fundamental para a valorização destes, bem como para facilitar o seu aproveitamento por parte das aplicações em fase de produção. Por vezes, os dados em bruto (*raw*) são difíceis de ser processados e compreendidos pelas partes que os requerem. Como tal, a devida preparação dos mesmos soluciona este problema [41, 100]. As organizações podem vender diretamente os dados com pouca atenção e investimento na sua transformação, ou podem vender dados processados e analisados de forma a vender serviços de dados [97]. No seu ciclo de vida, os dados são sujeitos a diversas transformações que aumentam a sua valorização. Os dados em bruto são convertidos em dados tratados que permitem obter novos conhecimentos e serviços. Num *data marketplace*, o proprietário é responsável pelo controlo e processamento dos dados. Tanto a recolha dos dados, como a distribuição dos mesmos, são duas etapas cruciais em qualquer *data marketplace*, no entanto, existem outras que não são constantes e são alvo de ajustes conforme o âmbito do projeto a desenvolver. Tanto o número de etapas presentes como a sua ordenação podem sofrer alterações [13, 24].

Curry et al. [13], na sua pesquisa centrada em *value chain*, divide o ciclo de vida dos dados em cinco etapas, sendo elas:

1. ***Data Acquisition***. Este é o processo de recolha, filtragem e limpeza dos dados antes de serem colocadas no local de armazenamento para posterior análise dos mesmos.

2. ***Data Analysis***. Esta etapa é responsável por explorar e transformar os dados em bruto adquiridos, de forma a extrair a informação útil e mais relevante para que se possam tomar decisões adequadas em cada caso de utilização específico. As duas formas populares de fazer estas análises é através de *machine learning* e *data mining*.

3. ***Data Curation***. Toda a gestão da qualidade dos dados é assegurada nesta etapa, de forma a que sejam garantidos todos os requisitos para uma utilização eficaz. A etapa de *data curation* pode ser dividida em: tarefas de criação, seleção, classificação, transformação, validação e preservação.

4. ***Data Storage***. Esta é responsável por tornar os dados persistentes de uma forma escalável para satisfazer as necessidades das aplicações que necessitam de um acesso rápido aos dados. Normalmente recorre a bases de dados *Not Only SQL* (NoSQL).

5. ***Data Usage***. Esta é a última etapa que consiste na utilização dos dados com ferramentas necessárias para a atividade empresarial. A utilização de dados por estas entidades pode aumentar a competitividade através da redução de custos, do aumento do valor acrescentado ou de qualquer outro parâmetro que possa ser medido em relação aos critérios de desempenho existentes.

Bergman et al. [7] realizou um projeto no qual, através de entrevistas feitas a proprietários de *data marketplaces*, identificou seis etapas de processamento dos dados:

1. ***Data Collection***. Refere-se a esta tarefa como a recolha de dados dos vendedores. O autor menciona que a negociação dos termos e condições de utilização dos dados entre os participantes do *data marketplace* se concretiza nesta etapa. O acordo em relação aos dados durante a sua recolha influencia as tarefas que são realizados nas fases seguintes.

2. ***Data Standardization***. Esta tarefa descreveu-a como a chave da interoperabilidade dos dados. Permite que os dados recolhidos das diferentes fontes sejam convertidos num único formato de forma a sejam mais facilmente compreendidos e partilhados por todas as partes envolvidas no processo de troca dos dados.

3. ***Data Cleaning***. É responsável por toda a gestão da qualidade dos dados, verificando a consistência e o conteúdo dos mesmos.

4. ***Data Storage***. Em relação ao armazenamento dos dados, é mencionado que estes necessitam de ser armazenados num local seguro e escalável e, podem ser armazenados numa plataforma centralizada ou descentralizada.

5. ***Data Analysis***. Esta etapa, abrange a agregação e análise de dados de forma a uma fácil leitura, compreensão e obtenção de novos conhecimentos a partir destes .

6. ***Data Distribution***. Refere-se a esta tarefa como sendo fundamental num *data marketplace* assim como *data collection*. Os dados são recolhidos dos vendedores e distribuídos aos seus compradores. São distribuídos aos compradores conforme as condições previamente acordadas durante a recolha de dados.

2.5.1 Valorização dos Dados num *Data Marketplace*

Em relação à valorização dos dados num *data marketplace*, este pode desempenhar vários papéis. Na sua forma simplista, funcionam como *raw-data trader*, ou seja, são intermediários de dados em bruto, não processados. Numa outra forma, estes atuam como *data normalizer*, definindo modelos padrão, formatos e atributos aos dados para serem comercializados. Um *data marketplace* com o papel de *data agregator* valoriza os dados através da agregação destes em pacotes lógicos, como por exemplo, todos os dados de uma determinada região são agrupados e oferecidos a fornecedores de serviços. Ao assumirem o papel de *quality assurer*, verificam o conteúdo, a consistência e a qualidade dos dados [19].

2.5.2 Agregação de Dados

A agregação de dados tem sido aplicada no processamento e gestão de dados nos sistemas de informação modernos, a fim de economizar espaço de armazenamento, diminuir a largura de

banda e os custos energéticos, encontrar padrões pouco habituais e ganhar novos conhecimentos a partir da informação. Os processos de agregação de dados têm-se tornado cada vez mais populares em domínios que lidam com a recolha de grandes quantidades de dados de dispositivos que são posteriormente analisados, como é o caso da IoT e da *cloud computing*. Um exemplo prático que podemos ter em consideração para uma melhor compreensão dos processos de agregação, consiste no controlo de uma casa através de uma aplicação de vigilância que agrega os dados provenientes de várias câmaras e sensores. Os dados agregados de cada casa, podem ser novamente agregados na *cloud* para que seja possível analisar a segurança da região [9].

Os processos de agregação de dados são considerados uma sequência de três etapas, sendo estas [9]:

1. **Preparação dos dados em bruto**. Nesta etapa é realizada a preparação necessária dos dados em bruto como localização, extração, transporte e normalização. Esta é realizada desde que os dados saem da fonte de dados até que chegam ao elemento responsável pela agregação, o agregador.

2. **Agregação dos dados em bruto**. Nesta etapa, o agregador aplica funções de agregação aos dados em bruto de forma a transformá-los em dados agregados.

3. **Posterior tratamento dos dados agregados**. Nesta etapa os dados agregados são tratados pelo agregador que os pode armazenar persistentemente ou disponibilizá-los para outros processos.

As **funções de agregação** são amplamente aplicadas em inúmeros domínios, como é o caso da economia, finanças, estatística e ciências da computação. Geralmente estas funções são definidas e utilizadas para tornar um conjunto de valores numéricos num só, de forma a que o resultado da agregação tenha em consideração todos os valores individuais. Consistem em funções matemáticas para cálculos estatísticos. As cinco funções de agregação básicas são: SUM, MAX, MIN, AVERAGE e COUNT [59].

A agregação de dados é aplicada nas redes de sensores sem fios, como é o caso da IoT. As redes de sensores sem fios, geralmente, são constituídas por conjuntos organizados de nós que monitorizam condições físicas como movimentos, luz, humidade e temperatura. Processam os dados em bruto, enviam a informação resultante para a estação base, também denominada *sink* [12]. Neste âmbito, a agregação de dados permite que cada nó agregue os dados gerados por si e também aqueles que recebe, encaminhando apenas os dados agregados. Assim, a quantidade de dados enviados na rede é consideravelmente reduzida o que, consequentemente, leva à diminuição do consumo de energia e da largura de banda, prolongando assim a vida útil da rede [53].

De uma forma sintetizada, a agregação é o processo no qual os dados em bruto são recolhidos e transformados num formato sumariado para análise estatística. É um processo que demonstra uma enorme utilidade na progressão das *smart cities* pois, pelos conhecimentos que se podem obter a partir dos resultados destes processos é possível melhorar diversos serviços e também as

atividades diárias dos indivíduos. Por exemplo, o nível máximo ou média das concentrações de pólen e poluição atmosférica podem ser do interesse dos indivíduos para que possam planear as atividades ao ar livre. O valor máximo da temperatura numa região e num determinado período de tempo, pode ser benéfico para prevenir o risco de incêndios. A quantidade média de exercício físico diário que os indivíduos praticam, medida por sensores de movimento, pode ser útil para inferir as condições de saúde pública [58].

2.6 Gestão de Identidade e Acesso

Nos dias de hoje, quando se disponibilizam dados e recursos através da rede existe uma necessidade de identificação dos utilizadores e controlo sobre os serviços aos quais estes estão autorizados a utilizar [46]. São necessários mecanismos que previnam o risco de acesso e uso indevido de dados por indivíduos não autorizados. Como tal, a *Identity and Access Management* (IAM) vem colmatar esses problemas de segurança. A IAM assegura que apenas as pessoas corretas têm acesso aos recursos certos. Controla o acesso aos recursos através da verificação do utilizador e da autorização, no tempo certo e pelas razões certas [65]. A IAM é composta por dois elementos, a gestão de identidade e a gestão de acesso.

O conceito identidade refere-se ao conjunto de caraterísticas únicas de uma entidade que tanto podem ser indivíduos, *software*, como dispositivos. Uma identidade utilizada para fins de identificação é denominada de identificador [4, 88]. A gestão de identidade é responsável por gerir o ciclo de vida das identidades (criação, gestão, atualização, eliminação). Está relacionada com o aprovisionamento das identidades, isto é, trata da criação de contas e identificadores dos utilizadores para permitir o acesso aos recursos por parte do utilizador, de outra forma, também restringe a conta quando já não é necessário o acesso aos recursos. A gestão de acesso é o segundo elemento da IAM e assegura a autenticação, autorização e gestão de políticas [88].

2.6.1 Autenticação

A **autenticação** é o processo através do qual é validada a identidade de um utilizador que requer o acesso a determinadas informações restritas, aplicações ou outros serviços. Para uma melhor compreensão do processo de autenticação, este pode ser classificado em quatro fatores, também conhecidos como os quatro Ss. [14, 48]:

1. ***Something you know***. Este fator refere-se a **algo que o utilizador sabe** como palavra-passe, *Personal Identification Number* (PIN), *passphrases*.
2. ***Something you have***. Este fator inclui **algo que o utilizador possui** como *smart cards*, *pendrive*, telemóveis, entre outros aparelhos físicos.
3. ***Something you are***. Este fator refere-se a **algo que carateriza o utilizador** como dados biométricos que podem ser impressões digitais, *scan* da retina, reconhecimento facial

e também padrões comportamentais como reconhecimento de voz, entre outros.

4. ***Someplace you are***. Este fator está relacionado com **algum local onde o utilizador se encontra** ou determinado processo. Inclui localizações geográficas e endereços IP.

O tipo de autenticação pode também ser classificada tendo por base o número de fatores, acima mencionados, os quais são utilizados durante o seu processo [76]:

- **Autenticação de fator simples**. Este tipo de autenticação recorre a um único fator e tem como princípio "aquilo que o utilizador sabe". A técnica de autenticação mais utilizada é a autenticação por palavra-passe que combina o nome de utilizador com uma palavra-passe.
- **Autenticação multi-fator**. Este tipo de autenticação recorre a dois ou mais fatores. Normalmente, inclui dados biométricos do utilizador. Por exemplo, o levantamento de dinheiro numa *Automated Teller Machine* (ATM), requer um *token* físico, o cartão e um código PIN. É a combinação de "algo que o utilizador possui", o cartão, com "algo que o utilizador sabe", o código PIN. É possível tornar esta ação mais segura, ao adicionar mecanismos biométricos.
  - **Autenticação de dois fatores**. Este tipo de autenticação e o mais comum da autenticação multi-fator e recorre a dois fatores. Geralmente é utilizada a combinação nome de utilizador/palavra-passe com outro fator, com "algo que o utilizador possui", como por exemplo, um token utilizando One-Time Password (OTP) [78].

A **autorização**, também denominada de **controlo de acesso**, é o processo que sucede após a autenticação ser realizada com sucesso. É o responsável pela mediação dos pedidos feitos aos recursos e dados presentes num sistema, determinando se o pedido deve ser concedido ou negado [84].

2.6.2 Protocolos de Autenticação e Autorização

Quando pretendemos desenvolver um sistema de autenticação e autorização é necessário compreender como são geridas as identidades e também como é feita a gestão de acesso aos recursos protegidos. Assim, a existência de protocolos que permitem especificar como essas ações podem ser realizadas é fundamental. Antes de apresentarmos alguns dos padrões e protocolos existentes, é necessário esclarecer o conceito *Single Sign-On* (SSO). O SSO é um processo que permite a um utilizador autenticar-se apenas uma vez num sistema no qual as credenciais da sessão são confiadas entre diferentes aplicações dentro de um domínio seguro [1]. Iremos agora apresentar alguns padrões e protocolos:

- **OAuth2.0**: O OAuth2.0 é um protocolo de autorização que permite a um utilizador conceder acesso aos seus recursos a outros *web sites*. Dessa forma, não necessita de expor

constantemente as suas credenciais cada vez que pretende aceder a um *site* ou aplicação. Este protocolo, é utilizado por diversos fornecedores de identidade como o Facebook, Google, Microsoft e muitos outros [26].

- **OpenID**: é um padrão aberto baseado no protocolo OAuth2.0. Permite a realização da autenticação dos utilizadores for meio de fornecedores de identidade como é o caso da Microsoft, Facebook, Google, entre outros [11].

- ***JSON Web Token* (JWT)**: JWT é um padrão aberto que define uma forma compacta, independente e segura de transmitir informações como sendo objetos JSON, entre duas partes. Estes *tokens* são utilizados em diferentes cenários como processos de autorização e de troca de informações. Os processos de autorização são aqueles onde são mais utilizados JWT. Estes processos que ocorrem após o utilizador estar autenticado, cada pedido realizado por este inclui um JWT, permitindo ao utilizador aceder a serviços e recursos protegidos. São também bastante utilizados no processo SSO. Os JWT são uma boa opção na troca segura de informações entre duas partes, uma vez que podem ser assinados, por exemplo, através de pares de chaves públicas/privadas, permitindo assim, verificar a identidade do utilizador que envia a informação. A assinatura permite inferir que apenas o utilizador na posse da chave privada é que assinou a informação [75].

- ***Security Assertion Markup Language* (SAML)**: O SAML é um padrão aberto baseado no formato XML que permite a troca de informações de autenticação e autorização entre aplicações parceiras de negócios. Tal como o JWT, também este padrão é muitas vezes utilizado no processo SSO [54].

2.6.3 Modelos de Controlo de Acesso

As políticas de controlo de acesso correspondem a requisitos de alto nível que descrevem como é realizada a gestão de acesso, assim como quem pode ter acesso à informação e em que circunstâncias. Estas são implementadas por meio de um mecanismo responsável pelo pedido de acesso feito por um utilizador. Os modelos de controlo de acesso estabelecem a ponte entre as políticas e os mecanismos e descrevem as caraterísticas de segurança de um sistema de controlo de acesso [55]. De seguida, serão apresentados alguns dos modelos de controlo de acesso existentes:

- ***Discretionary Access Control* (DAC)**: este modelo de controlo de acesso baseia-se na identidade do utilizador que requer o acesso e em regras que ditam quem pode ou não exercer ações e sobre quais recursos. Os utilizadores podem conceder e também revogar os seus privilégios a outros utilizadores [84].

- ***Mandatory Access Control* (MAC)**: este modelo de controlo de acesso é baseado na classificação dos *subjects* e objetos. Os objetos são as entidades passivas que armazenam as informações, enquanto que os *subjects* são as entidades ativas que solicitam o acesso aos objetos. Ao contrário do modelo DAC no qual os utilizadores são humanos, aqui as partes

ativas correspondem a processos, ou seja, são programas em execução. A classificação é baseada numa hierarquia de níveis de segurança, na qual as entidades ativas só têm acesso a objetos que tenham um nível igual ou inferior ao seu [84].

- ***Role Based Access Control* (RBAC)**: neste modelo, o controlo de acesso é baseado em *roles*. As *roles* estão associadas a conjuntos de permissões e por sua vez, os utilizadores estão associados a *roles*, adquirindo assim as permissões. Numa organização as *roles* são concebidas para as diferentes funções a desempenhar e são atribuídas aos utilizadores tendo por base as suas qualificações e responsabilidades. Os utilizadores podem facilmente mudar de *roles* [85].

- ***Attribute Based Access Control* (ABAC)**: este modelo de controlo de acesso baseia-se nos atributos das entidades ativas e passivas, ou seja, nas suas caraterísticas. Através deste modelo, o controlo de acesso aos objetos é realizado em função da avaliação de regras baseadas nos atributos dos *subjects*, objetos e do ambiente em que o pedido de acesso foi feito. Definem, através de regras, as combinações de atributos autorizadas a aceder a um determinado objeto [40]. Os atributos dos *subjects* podem ser por exemplo, a sua identidade, idade *role*, código postal, enquanto que os atributos dos objetos correspondem a a sua identidade, localização, tamanho, valor. Por último, os atributos do ambiente podem ser a hora do dia, estado do sistema, data [89].

Capítulo 3

Estado da Arte

Neste capítulo é apresentada a pesquisa realizada sobre todo o contexto do trabalho, principalmente dos *data marketplaces* existentes na atualidade. Além disso, é feita uma abordagem das caraterísticas essenciais ao desenvolvimento de um *data marketplace* e, por fim, apresentamos quais as caraterísticas presentes em cada um destes sistemas resultantes dessa mesma pesquisa. Na última secção, em 3.2, são apresentados e discutidos os simuladores de dados que foram estudados para uma posterior integração com o nosso sistema.

3.1 Data Marketplaces

Após ter sido efetuada uma abordagem em torno do conceito de *data marketplace* por vários autores na secção 2.2 foi então realizada uma pesquisa sobre alguns *data marketplaces*. Foram tidas em conta algumas caraterísticas consideradas fundamentais para o desenvolvimento do projeto. Tanto a arquitetura como a transformação dos dados são duas caraterísticas inseridas por Spiekermann et al. [92] no seu esquema de classificação de *data marketplaces*. A **arquitetura** foi eleita para o estudo do estado da arte pois, esta caraterística define a infraestrutura organizacional e tecnológica utilizada para fornecer serviços e produtos aos clientes interessados. Assim, a arquitetura é a propriedade fundamental de qualquer plataforma de comercialização de dados. A **transformação dos dados** foi também escolhida uma vez que indica o nível de processamento a que os dados são submetidos no *data marketplace* e como são oferecidos aos clientes.

Por sua vez, a caraterística **tipo de dados** é abordada no estado da arte pelo facto de indicar a presença de dados pessoais ou não pessoais, de forma a perceber que cuidados devem ser tidos no desenvolvimento de um *data marketplace* conforme os dados que nele são oferecidos. As caraterísticas **mecanismos de autenticação** e **controlo de acesso** foram escolhidas para que seja possível compreender de que forma se previne que indivíduos e dispositivos não autorizados tenham acesso ao *data marketplace* e como são atribuídas as permissões de acesso aos recursos da plataforma aos utilizadores, respetivamente.

A forma como o controlo sobre os dados é concedido ao proprietário destes é também algo

fundamental a ter em consideração quando o assunto são *data marketplaces*. Dessa forma, ter conhecimento de como estas plataformas atuam para permitir aos utilizadores controlar para que finalidade os seus seus dados são utilizados, como lhes é concedido o direito de escolher os dados que pretendem partilhar e com quem, é fundamental. A possibilidade de estarem sempre atualizados de atividades passadas e presentes sobre os seus dados é também essencial. Como tal, para os cenários descritos, as caraterísticas **transparência dos dados**, **granularidade dos dados**, **políticas dos vendedores** e **registo de transações** foram escolhidas para o estudo do estado da arte.

É preciso salientar que alguns dos *data marketplaces* estudados ainda se encontram em fase de estudo/produção, nomeadamente o I3 - Intelligent IoT Integrator e o IOTA. De seguida, iremos detalhar todas as caraterísticas escolhidas para o estudo do estado da arte da dissertação.

3.1.1 Arquitetura

A arquitetura de um *data marketplace* pode ser de dois tipos: centralizada e descentralizada. Numa plataforma centralizada, os dados são oferecidos e armazenados por meio de um local central, por exemplo, uma infraestrutura *cloud*. Numa plataforma descentralizada os dados permanecem do lado do fornecedor [92]. A comercialização dos dados é realizada diretamente entre compradores e vendedores sem intervenção de um intermediário [49].

Wibson é um *data marketplace* que apresenta uma arquitetura descentralizada baseada na tecnologia *blockchain* [73]. Devido à sua infraestrutura descentralizada, cada vendedor, guarda os seus próprios dados, encriptados nos seus dispositivos [99]. O IOTA é outro exemplo de uma plataforma descentralizada, no entanto, devido aos custos das transações da *blockchain*, desenvolveram uma *Distributed Ledger Technology* (DLT), própria sem taxas de transação, designada Tangle [7, 94]. O I3 é um *data marketplace* que inicialmente foi desenvolvido com uma infraestrutura centralizada, mas que se encontra numa fase de migração para a descentralização [52]. O Dawex é um *data marketplace* constituído por uma infraestrutura centralizada onde os dados são guardados encriptados [35].

3.1.2 Tipos de Dados

Esta caraterística está relacionada com o domínio dos dados comercializados, assim como o tipo de dados, por exemplo, dados pessoais, ou dados produzidos por sensores.

Wibson é um *data marketplace* no qual o produto comercializado são dados pessoais. Podemos dizer que esta plataforma é de domínio específico, com um papel ativo na indústria de dados pessoais. Alguns exemplos de dados disponíveis nesta plataforma são *mobile Advertising identifier*, o histórico de pesquisas, ou até de transações bancárias de indivíduos [25] .

Ao contrário do Wibson, o IOTA e o Dawex são dois *cross-domain data marketplaces*. As indústrias alvo destes *data marketplaces* passam por *smart cities*, energética, cuidados de saúde,

automóvel, ambiente, entre outras [18, 32]. O IOTA permite a partilha de dados produzidos em tempo real por sensores. [32] O I3 é um *data marketplace*, concebido para o domínio das *smart cities*, que tal como o IOTA permite a troca de dados produzidos em tempo real entre sensores e aplicações [52].

3.1.3 Transformação dos Dados

Para descrever esta caraterística, vamos considerar a definição de transformação dos dados proposta por Spiekerman et al. [92]. Segundo o autor, a transformação dos dados está relacionada com o grau de verificação e preparação sintática e semântica dos dados no *data marketplace*. Estes podem ser comercializados sem qualquer tipo de processamento, serem submetidos a processos de normalização, agregação e de verificação da qualidade.

Quanto à transformação dos dados, podemos dizer que o Wibson fornece os dados numa forma agregada, de acordo com as preferências do comprador. Tendo em conta um dos exemplos referidos pelo autor, um comprador pode solicitar todas as transações bancárias, realizadas num dia especifico, por todos os indivíduos do sexo feminino, com idade superior a uma determinada idade, rendimentos superiores a um determinado valor, residentes num determinado pais [99]. O IOTA e o Dawex, são dois dos *data marketplaces* analisados por Spiekermann et al. [92]. Segundo a análise do autor, tanto o IOTA como o Dawex comercializam os dados na plataforma, sem qualquer processamento. No I3 há a comercialização de dados não processados [52].

3.1.4 Mecanismos de Autenticação

Esta caraterística refere-se aos mecanismos utilizados para a realização da autenticação dos utilizadores e dispositivos no *data marketplace*.

Do *data marketplace* Wibson, fazem parte três participantes, o vendedor dos dados pessoais, o comprador e uma entidade à qual o autor denominou de notário. Este último tem como função a verificação da informação dos participantes e da qualidade e autenticidade dos dados. A mediação em caso de conflito entre vendedores e compradores, tudo isto, apenas quando solicitado. Este possui os seus próprios ficheiros com as informações relativas aos vendedores. O notário pode verificar a identidade do vendedor, solicitando-lhe informações que de alguma forma o autentiquem [99].

No IOTA a autenticação dos utilizadores é baseada no protocolo OAuth através da entidade federada Google [33]. No I3 a autenticação dos dispositivos é realizada por meio de autenticação baseada em palavra-passe, ou criptografia de chave assimétrica [102]. Por sua vez, a autenticação dos utilizadores é realizada através do tradicional método de e-mail e palavra-passe [43]. No Dawex a autenticação das contas dos utilizadores é gerida unicamente por meio de canais encriptados, utilizando algoritmos seguros e chaves de alta segurança. Oferece uma autenticação de dois fatores pelo método e-mail e palavra-passe com adicional envio de mensagem com código

temporário para inserir na plataforma [15].

3.1.5 Controlo de Acesso

De uma forma sucinta, esta caraterística está relacionada com os processos utilizados na gestão dos acessos concedidos a um utilizador. Este conceito foi abordado anteriormente em 2.6.

Wibson não realiza controlo de acesso. Não possui uma autoridade central que conceda ou negue permissões de acesso no *data marketplace* [99]. No I3 o *broker* que favorece a troca de dados entre vendedores e compradores é integrado com um *plugin* de autenticação-autorização e uma função de verificação de *JSON Web Token* (JWT) [102]. No IOTA o controlo de acesso aos recursos armazenados na Tangle é realizado através do protocolo *Masked Authenticated Messaging* (MAM) [30]. No Dawex o controlo de acesso é assegurado por protocolos de encriptação e *Access Control List* (ACL) [6, 15].

3.1.6 Transparência dos Dados

A transparência dos dados refere-se ao controlo sobre os dados por parte do proprietário dos mesmos. Se este tem a capacidade de saber para que finalidades os dados foram adquiridos. A transparência sobre a utilização dos dados, é a visibilidade sobre a forma como os dados são utilizados pelo comprador, por parte do vendedor [99].

Os *data marketplaces* Wibson e Dawex, foram construídos baseados na tecnologia *blockchain*, que lhes oferece transparência na utilização dos dados [99]. O mesmo acontece com o IOTA graças à tecnologia Tangle [34]. Estas plataformas utilizam *smart contracts* para estabelecer os acordos entre os vendedores e compradores [103]. No protótipo inicial do I3 são estabelecidos termos e condições sobre os quais os dados podem ser utilizados [51].

3.1.7 Granularidade dos Dados

Esta caraterística está relacionada com o nível de detalhe da informação oferecida pelo proprietário dos dados. Este tem o poder de decidir que partes dos seus dados pretende partilhar. Quanto mais informação este partilhar, maior o nível de granularidade.

No *data marketplace* Dawex, os utilizadores têm o poder de decidir o nível de informação partilhada [15]. O protótipo inicial do I3 é integrado com um mecanismo de segurança que permite fornecer diferentes versões de dados para diferentes níveis de autorização [51]. Relativamente aos restantes *data marketplaces* não foram encontradas informações.

3.1.8 Políticas dos Vendedores

A caraterística políticas dos vendedores diz respeito às preferências de partilha dos seus dados. Os vendedores devem ter o poder de definir com que entidades estão dispostos a partilhar os seus dados bem como com as que não pretendem que os seus dados sejam partilhados.

No Dawex os utilizadores têm total controlo sobre os dados, eles decidem com quem querem partilhá-los. É permitido aos utilizadores escolher quais as áreas de negócio que terão acesso aos dados, como também quais os territórios [16]. O protótipo inicial do *data marketplace* I3 permite aos utilizadores limitar o acesso e aprovar a utilização dos seus dados a aplicações específicas [51]. Relativamente aos restantes *data marketplaces* não foram encontradas informações.

3.1.9 Registo de Transações

O registo de transações refere-se ao registo da atividade de um utilizador na plataforma. Esta caraterística permite ter um certo controlo sobre as ações que cada utilizador realiza, guardando-as. É possível saber a que informações um determinado utilizador teve acesso.

No Wibson e no Dawex, o registo de transações é realizado na *blockchain* [99]. O Dawex permite registar e aceder a todas as comunicações e histórico de transações em qualquer momento [17]. Por sua vez, no IOTA o registo das transações é feito na Tangle [31]. O protótipo inicial do I3 utiliza um registo de transações centralizado, no entanto, tecnologias distribuídas de registo estão a ser exploradas [52]. Neste *data marketplace* informações relevantes dos utilizadores, sensores disponíveis, condições e termos de uso, preços, transações em curso, completas e pagamentos são armazenadas numa base de dados à qual denominaram de registo [51].

A tabela que se segue reúne todos os resultados da pesquisa realizada.

Data Marketplace	Arquitetura	Mecanismos de Autenticação	Controlo de Acesso	Transformação dos Dados	Tipos de Dados	Transparência dos Dados	Granularidade dos Dados	Políticas dos Vendedores	Registo de Transações
I3 [102]	Centralizada	●	●	*Raw*	Sensores	●	●	●	●
IOTA [32]	Descentralizada	●	●	*Raw*	Sensores	●	○	○	●
Dawex [16]	Centralizada	●	●	*Raw*	Ambos	●	●	●	●
Wibson [99]	Descentralizada	●	○	Agregação	Pessoais	●	○	○	●
Nossa solução	**Centralizada**	●	●	*Raw*/Agregação	Sensores	○	◐	●	●

● - caraterística presente ○ - caraterística ausente ◐ - caraterística semi-presente

Tabela 3.1: Comparação entre *data marketplaces*

3.2 Simuladores de Dados IoT

Atualmente, as aplicações de *Internet of Things* (IoT) estão presentes em inúmeras áreas de negócio e no nosso quotidiano. Durante o desenvolvimento destas aplicações, é necessário ter em consideração certos desafios, como é o caso de controlar um elevado número de dispositivos e também recolher e processar uma enormidade de dados heterogéneos. O desenvolvimento e

consequente testagem destas aplicações pode ser realizada por meio de determinadas ferramentas, tais como simuladores de dados de IoT. Estes simuladores, têm como função, produzir dados em tempo real semelhantes a dados gerados por sensores de IoT, como por exemplo, sensores de temperatura, humidade, proximidade, pressão, entre outros. Estas ferramentas são uma mais valia durante a fase de desenvolvimento destes projetos, uma vez que tornam o processo mais rápido, mais barato e menos propício a falhas [90].

Nesta secção são apresentados alguns simuladores de dados de IoT resultantes de uma pesquisa realizada, com o objetivo de selecionar aquele que melhor se adapta ao nosso projeto. Estes simuladores, *open source* são: IoT-Data-Simulator[1], iot-device-simulator[2], IoT-generator-mongodb[3] e IoT-data-simulator[4]. Para que estes fossem devidamente comparados, foram definidas algumas características, como é possível visualizar na tabela 3.2.

3.2.1 IoT-Data-Simulator

IoT-Data-Simulator é uma aplicação desenvolvida em Python, que permite simular cenários reais de sistemas IoT [2]. Neste caso, foi simulada a recolha de dados de sensores na casa de mil utilizadores. Estes sensores recolhem valores de temperatura e humidade no interior e exterior de um quarto. O resultado desta simulação consiste num conjunto de dados que segue a seguinte estrutura [2]:

- **first_name:** Primeiro nome de cada utilizador.
- **last_name:** Último nome de cada utilizador.
- **age:** Idade de cada utilizador.
- **gender:** Género de cada utilizador.
- **username:** Nome de utilizador de cada utilizador.
- **address:** Morada de cada utilizador.
- **email:** E-mail de cada utilizador.
- **sensor_record:** Registos dos sensores para cada utilizador.

O parâmetro **sensor_record** corresponde a mil registos de sensores para cada um dos utilizadores e é constituído por [2]:

- **date:** Data em que os dados foram gerados.

[1] https://github.com/adrianmuino/IoT-Data-Simulator
[2] https://github.com/Temmermans/iot-device-simulator
[3] https://github.com/Wilpapa/IoT-generator-mongodb
[4] https://github.com/IBA-Group-IT/IoT-data-simulator

- **time:** Hora em que os dados foram gerados.
- **outside_temp:** Temperatura no exterior do quarto.
- **outside_hum:** Humidade no exterior do quarto.
- **room_temp:** Temperatura no interior do quarto.
- **room_hum:** Humidade no interior do quarto.

Após a produção dos conjuntos de dados é possível escolher se estes serão guardados num ficheiro em formato *JavaScript Object Notation* (JSON) ou *Comma Separated Value* (CSV). Para além das funcionalidades já mencionadas, esta ferramenta apresenta as seguintes [2]:

1. Cálculo de valores estatísticos para os dados recolhidos (média, desvio-padrão, percentis, entre outros).
2. Visualização de quatro histogramas (um por cada bloco de seis horas) relativos à frequência de valores da temperatura exterior do quarto.
3. Visualização de um gráfico de linhas relativo à evolução da temperatura exterior do quarto comparativamente à temperatura no interior do quarto.
4. Visualização de quatro histogramas que reúnem a informação de todos os utilizadores para cada uma das métricas.

3.2.2 iot-device-simulator

iot-device-simulator é uma aplicação desenvolvida em JavaScript, geralmente utilizada para simular sensores IoT, assim como em projetos de *Smart Data Streaming* (SDS) como forma de produção de fluxos de eventos [96]. É possível simular qualquer tipo de sensor IoT, até um máximo de vinte sensores. Os conjuntos de dados gerados por cada sensor são transmitidos continuamente para um *endpoint* específico em formato JSON e também para uma base de dados não relacional denominada MongoDB. No entanto, estes dados não são persistentes, pois ao parar a transmissão a base de dados é restaurada de forma a evitar a sua sobrecarga no caso de se proceder a transmissões de conjuntos de dados regularmente [96]. Estes conjuntos de dados são constituídos pelos seguintes campos:

- ***ESP_OPS***[5]**:** Tipo da operação do evento. Os valores possíveis são i: insert, u: update, d: delete, s: safedelete e p: upsert.
- **attributeName:** Atributo de cada sensor.
- **dataType:** Tipo de cada atributo (*string*, *number*, *boolean*, *GPS-string*).

[5]Relevante se o simulador for utilizado num projeto SDS.

- **deviceName:** Nome do sensor.
- **timestamp:** Momento em que os valores são gerados.
- **value:** Valor gerado para o atributo.
- **fixedValue:** Valor fixo atribuído pelo utilizador ao atributo.
- **min:** Limite inferior para os valores gerados do atributo.
- **max:** Limite superior para os valores gerados do atributo.

3.2.3 IoT-generator-mongodb

IoT-generator-mongodb é uma ferramenta desenvolvida em Python que gera valores de medições de temperatura de sensores IoT. A recolha das medições pode ser realizada, utilizando um máximo de dezoito sensores e decorre durante um período de dias estipulado pelo utilizador. Os dados gerados são guardados na base de dados não relacional MongoDB. O autor deste projeto utiliza um método de organização de dados designado *time bucketing* segundo o qual, cada objeto dos conjuntos de dados gerado corresponde a uma hora de medições de temperatura para cada sensor. Cada sensor regista uma medição por minuto, porém, algumas medições podem ser perdidas, assim, cada objeto possui no máximo sessenta medições [61]. Os objetos dos conjuntos de dados recolhidos seguem a seguinte estrutura [61]:

- **id:** Identificação do sensor.
- **measureDate:** Dia e hora da realização da recolha dos dados.
- **measureUnit:** Unidade de medida (°C).
- **periodAvg:** Média das medições ocorridas no período de uma hora.
- **periodMax:** Valor máximo que a medição pode tomar.
- **periodMin:** Valor mínimo que a medição pode tomar.
- **missedMeasures:** Número de medições perdidas.
- **recordedMeasures:** Número de medições realizadas com sucesso.
- **values:** Conjunto de valores das medições realizadas.
  - **measureMinute:** Minuto em que ocorre a medição.
  - **measuredValue:** Valor da medição.

Esta estrutura apresentada tem diversas vantagens, tais como a redução da cardinalidade, número de objetos no conjunto, devido à utilização do método *time bucketing*. Outras vantagens prendem-se com o facto de certos valores estatísticos serem calculados na criação do conjunto de dados e o rápido acesso aos dados gerados em intervalos de uma hora [61].

3.2.4 IoT-data-simulator

IoT-data-simulator é um simulador genérico de sensores IoT desenvolvido em JavaScript numa plataforma *open source* designada Docker. Possibilita a utilização de conjuntos de dados previamente criados pelo utilizador, através do *upload* dos mesmos em formato JSON ou CSV. Permite, também, a produção de dados *on the fly*, isto é, produzir conjuntos de dados e reconfigurá-los sem a necessidade de reiniciar o processo [37]. Este é um simulador heterogéneo, permite simular qualquer tipo de sensor IoT e, para isso, oferece uma grande variedade de tipos de dados, tais como *integer*, *long*, *timestamp*, *array* e *object*. No que diz respeito a tecnologias utilizadas na comunicação entre cliente e servidor o utilizador tem a liberdade de escolher entre as seguintes [37]:

1. *Advanced Message Queuing Protocol* (AMQP).
2. *Message Queuing Telemetry Transport* (MQTT).
3. *Apache Kafka*.
4. *Representational State Transfer (REST) web services*.

O **tipo de dados** que é possível gerar por cada simulador é um ponto bastante importante a abordar nesta pesquisa. Uma vez que este projeto se enquadra no âmbito da IoT e das *smart cities*, quanto maior a heterogeneidade que um simulador apresenta, mais tipos de dados diferentes é permitido gerar, tornando-se assim mais útil. A permissão de **dados de localização**, isto é, a possibilidade de simular coordenadas geográficas é também essencial quando se trata da simulação de sensores IoT. Também o facto destes serem integrados com uma **interface gráfica** é uma mais valia pela razão de facilitar o seu manuseamento e a sua utilização. Outro ponto fundamental para esta pesquisa e uma vez que projetos relacionados com a IoT requerem a produção de grandes quantidades de dados em tempo real, a integração destes simuladores com uma plataforma de transmissão de fluxos de dados é necessária. Assim, a possibilidade de uma fácil integração com a tecnologia **Kafka** é considerada.

Caraterísticas / Simuladores	Tipos de Dados	Dados de Localização	Interface Gráfica	Integração com Kafka
adrianmuino/ IoT-Data-Simulator	Temperatura, humidade	Não	Sim	Não
Temmermans/ iot-device-simulator	Genérico	Sim	Sim	Não
Wilpapa/ IoT-generator-mongodb	Temperatura	Não	Não	Não
IBA-Group-IT/ IoT-data-simulator	Genérico	Sim	Sim	Sim

Tabela 3.2: Comparação entre os diferentes simuladores

Capítulo 4

Desenho e Desenvolvimento

4.1 Arquitetura do Sistema

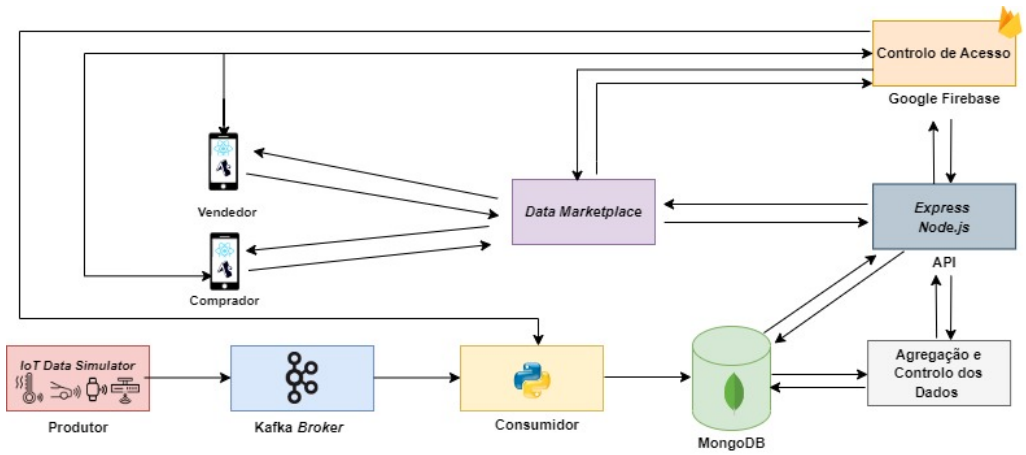


Figura 4.1: Arquitetura do *data marketplace* prosposto

A arquitetura da nossa proposta é apresentada na Fig. 4.1. Esta, pode ser dividida em duas etapas: simulação de dados de sensores de *Internet of Things* (IoT) reais e o desenvolvimento de um *data marketplace* para a compra e venda dos dados de IoT. A primeira etapa é constituída por quatro componentes: simulador de dados de IoT, *message broker*, consumidor do sistema de mensagens e a base de dados. Os primeiros três componentes referidos fazem parte de um sistema de mensagens do tipo *publish-subscribe*. Quanto à segunda etapa, consiste tal como já foi referido, no desenvolvimento de um *data marketplace* que tem como principal objetivo a transação de dados IoT entre os dois tipos de clientes do mesmo: vendedores e compradores. Fazem parte desta etapa alguns componentes fulcrais para conseguirmos cumprir os objetivos propostos neste projeto, sendo eles o *data marketplace*, a *Application Programming Interface* (API) o controlo de acesso e a agregação e controlo dos dados.

Outro ponto importante de salientar na arquitetura deste sistema é o fluxo de dados, isto é, o

fluxo de informação circulante entre componentes. Podemos de igual forma dividir a explicação destes fluxos nas mesmas duas etapas referidas anteriormente: simulação de dados de IoT reais e desenvolvimento de um *data marketplace*.

Ainda sobre a primeira etapa, poderíamos utilizar conjuntos de dados reais recolhidos por sensores reais mas para simplificar e facilitar o processo de desenvolvimento, utilizámos um simulador de dados. Os dados recolhidos/gerados são enviados em tempo real para um *message broker*, que recebe os conjuntos de dados de todos os sensores a ele ligados. De seguida, o componente consumidor é responsável por receber os conjuntos de dados do *message broker*, processá-los e torná-los persistentes ao guardá-los numa base de dados.

Na fase de desenvolvimento do *data marketplace*, o *back-end* é constituído por uma API responsável pela realização de *querys* à base de dados e também ao componente de agregação e controlo dos dados. Também é incumbido de realizar pedidos ao componente de controlo de acesso. O componente de agregação e controlo dos dados tem como função aplicar funções de agregação como médias, máximos e mínimos aos dados provenientes do simulador. Por sua vez, o componente de controlo de acesso é responsável pelos processos de autenticação e autorização dos múltiplos utilizadores do *data marketplace*, sendo eles os compradores e vendedores. Realiza toda a gestão das informações dos utilizadores, armazenado-as. Em relação ao *data marketplace*, este corresponde à plataforma onde os dados são publicados e vendidos. Nas secções seguintes serão descritos todos os componentes pertencentes a cada uma das etapas mencionadas anteriormente.

Relativamente ao ciclo de vida dos dados e como o *data marketplace* processa os dados de forma a valorizá-los e a criar novos conhecimentos a partir destes, desenvolvemos um estudo no qual identificámos várias etapas de processamento de dados. Como já foi referido em 2.5 estas etapas variam conforme a natureza e contexto do projeto a desenvolver. Várias etapas são definidas de autor para autor. No âmbito do nosso projeto, decidimos escolher as seguintes etapas: geração de dados, aquisição de dados, processamento de dados, armazenamento de dados, agregação de dados e distribuição de dados.

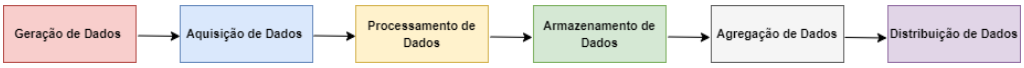


Figura 4.2: Etapas do ciclo de vida dos dados relativas ao nosso projeto

1. **Geração de Dados**. Nesta etapa decidimos proceder à geração dos dados por meio de um simulador de dados. Este processo poderia ser realizado através de sensores reais, no entanto, sai um pouco do domínio do nosso trabalho. Como tal, optámos pelo simulador de dados que também nos permitiu uma maior facilidade e rapidez no desenvolvimento. Todo o processo de geração de dados através do simulador será descrito em 4.2.0.1.

2. **Aquisição de Dados**. Após a geração de dados, estes são enviados em tempo real para um local de armazenamento, neste caso, os tópicos existentes no Kafka *broker* para depois serem processados pelo consumidor do Kafka.

3. **Processamento de Dados**. Esta etapa é realizada no componente consumidor do Kafka.

Aqui agrupamos as medições que cada sensor realiza em *buckets* de forma a que as medições fiquem agrupadas por sensor como será explicado em 4.2.0.3, e também agregamos todas as medições realizadas por cada tipo de sensor. Estes processos tornam o acesso aos dados e as consultas que são necessárias realizar à base de dados na etapa de agregação, mais simples.

4. **Armazenamento de Dados**. Nesta etapa os dados são tornados persistentes num local de armazenamento. Como o nosso projeto requer a geração de dados em tempo-real, ou seja, grandes quantidades de dados são gerados, optámos pela utilização de uma base de dados *Not Only SQL* (NoSQL), denominada MongoDB.

5. **Agregação de Dados**. Nesta etapa realizamos o cálculo de todos os valores estatísticos necessários através de consultas feitas à base de dados. Calculamos os valores de média, máximo e mínimo tanto para todos os sensores e de cada tipo, como para o conjunto dos sensores de cada tipo.

6. **Distribuição de Dados**. Nesta etapa os dados relativos aos sensores são colocados no *data marketplace* para que, utilizadores interessados os possam adquirir a fim de criar novos projetos, melhorar alguns já existentes e tomar melhores decisões.

4.2 Simulação de dados de sensores IoT reais

Pensar na conceção dos produtos, neste caso, dados de sensores IoT, que iriam ser comercializados na plataforma, foi o primeiro passo a tomar na produção do nosso projeto. Como tal, a produção desses dados através de um simulador foi o processo que nos pareceu mais vantajoso. Também tivemos de pensar na forma como os dados seriam transmitidos do simulador para serem devidamente armazenados. Assim sendo, precisávamos de um sistema que nos permitisse enviar fluxos de dados continuamente, de forma a simular fluxos de dados de sensores reais. Como tal, achámos que a utilização de um sistema de mensagens com um padrão *publish-subscribe* (2.3) seria a melhor opção, uma vez que estes são caraterizados pela transmissão de mensagens de forma assíncrona para as diferentes partes interessadas nestas mensagens [21].

De seguida, iremos abordar os três componentes pertencentes a este tipo de sistema de mensagens e também as tecnologias utilizadas para a sua implementação. Iremos também apresentar o componente responsável pelo armazenamento dos dados produzidos, a base de dados.

4.2.0.1 Simulador de Dados IoT

Relativamente à fase de geração de dados, optámos por utilizar um simulador de dados de IoT ao invés de sensores reais, por diversas razões. Primeiramente pela facilidade de integração do simulador no nosso projeto que, consequentemente, levou a um menor dispêndio de tempo na

tarefa de produção de dados. Outro motivo que nos levou à integração de um simulador de dados, foi o fato de necessitarmos da geração contínua de grandes quantidades de dados sem a preocupação de ocorrência de falhas e perda de dados. O último motivo prende-se com o fato de podermos adaptar a criação dos dados de acordo com sua necessidade ao longo da realização do projeto.

Através da pesquisa que realizámos em 3.2, para encontrarmos um simulador de dados que melhor se adequasse aos nossos objetivos, o simulador IoT-data-simulator, apresentado em 3.2.4, foi o escolhido por nós. Este simulador permitiu-nos uma facilidade extrema na conceção dos conjuntos de dados de que precisávamos, devido à vasta gama de tipos de variáveis que oferece, como também, pela sua fácil manipulação graças à interface gráfica que o integra e que tornou a sua utilização muito intuitiva. O fato de ter mecanismos que permitem a integração com um *message broker* também foi um ponto fundamental na escolha deste simulador.

No nosso trabalho, uma vez que este se insere na área das *smart cities*, decidimos simular sensores de monitorização de condições ambientais, como é o caso de sensores de temperatura, humidade relativa e pressão atmosférica. Os dados produzidos por este sensores podem ser utilizados no desenvolvimento de serviços de informação meteorológica, que as cidades podem oferecer aos seus cidadãos. Para além dos sensores referidos, optámos também por simular sensores dinâmicos, por exemplo, os dispositivos *Global Positioning System* (GPS) incorporados em trotinetas elétricas, que indicam o caminho percorrido por estas. Estes dados podem ser úteis para que empresas especializadas na área dos transportes ecológicos tenham conhecimento das zonas com maior afluência de trotinetas elétricas, no caso de pretenderem desenvolver novas aplicações.

No desenvolvimento deste componente, começámos por criar, através da interface gráfica quatro sessões. Podemos compreender cada sessão como o tipo de sensor que pretendemos simular, por exemplo, todos os sensores de um certo tipo são criados na sessão correspondente a esse tipo. No nosso caso, criámos uma sessão correspondente a sensores de temperatura, uma outra sessão relativa a sensores de humidade relativa, uma terceira de sensores de pressão atmosférica e uma sessão para os sensores dinâmicos.

Na Fig 4.3 são apresentadas as sessões criadas na interface gráfica do simulador. A primeira corresponde à sessão para sensores de temperatura, a segunda corresponde à sessão para sensores de humidade relativa, a terceira a sessão de sensores de pressão atmosférica e, por fim, a sessão de sensores dinâmicos.

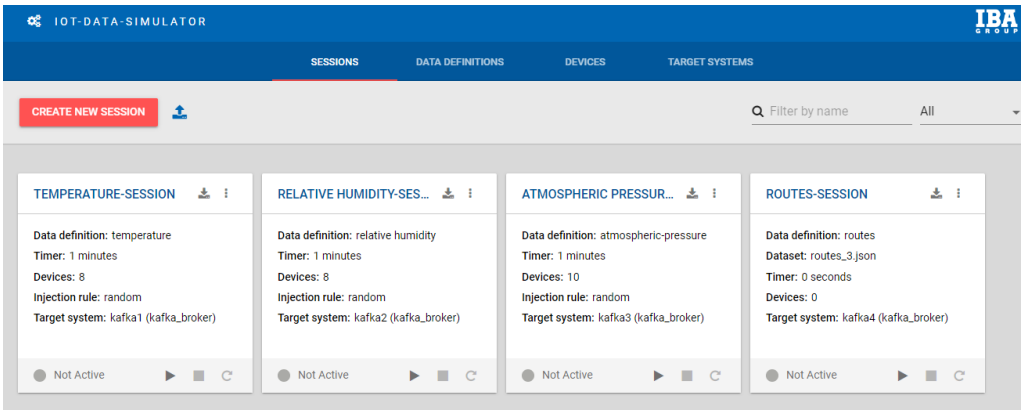


Figura 4.3: Sessões criadas na interface gráfica do simulador

De forma a que os sensores que pretendemos simular e consequentemente os dados produzidos por estes sejam o mais coerente possível com sensores de IoT reais, procedemos à seleção de atributos que nos pareceram ser os mais adequados para cada tipo de sensor. Na criação de sessões, este simulador possui uma secção da qual faz parte um esquema em formato *JavaScript Object Notation* (JSON), que corresponde ao conjunto de propriedades que irão satisfazer cada tipo de sensor. Cada propriedade é constituída por três campos: tipo, nome e propriedade. Esta última corresponde à identificação da propriedade no conjunto de dados final. O campo descrição é opcional. Relativamente ao esquema na criação das sessões de temperatura, humidade relativa e pressão atmosférica, pode ser observado nas Fig 4.4, 4.5 e 4.6, respetivamente. De seguida iremos apresentar bem como descrever os atributos que selecionámos para cada sessão.

A *Sensores de Temperatura*

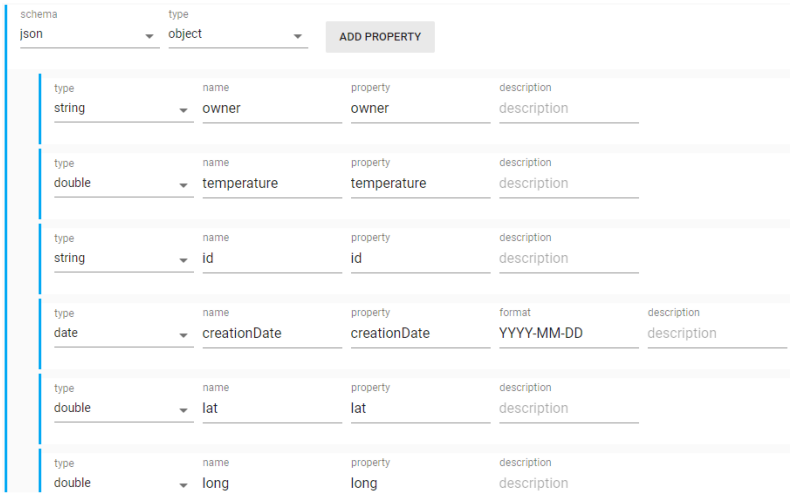


Figura 4.4: Esquema dos conjuntos de dados de sensores de temperatura

- **owner:** propriedade do tipo *string* que representa o proprietário do sensor.
- **temperature:** propriedade do tipo *double* que representa valores de temperatura medidos pelo sensor.

- **id:** propriedade do tipo *string* que representa a identificação do sensor.
- **creationDate:** propriedade do tipo *date* que representa a data de criação dos dados do sensor.
- **lat:** propriedade do tipo *double* que corresponde à coordenada latitude do sensor.
- **long:** propriedade do tipo *double* que corresponde à coordenada longitude do sensor.

B *Sensores de Humidade Relativa*

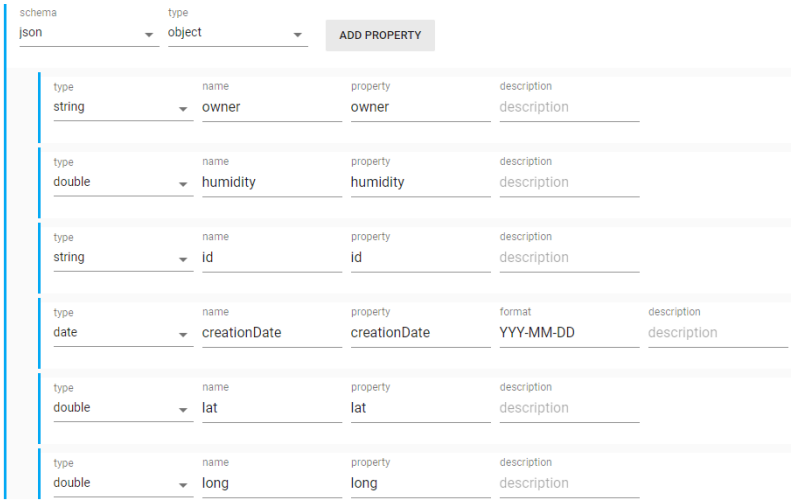


Figura 4.5: Esquema dos conjuntos de dados de sensores de humidade relativa

- **owner:** propriedade do tipo *string* que representa o proprietário do sensor.
- **humidity:** propriedade do tipo *double* que representam níveis de humidade no ar medidos pelo sensor.
- **id:** propriedade do tipo *string* que representa a identificação do sensor.
- **creationDate:** propriedade do tipo *date* que representa a data de criação dos dados do sensor.
- **lat:** propriedade do tipo *double* que corresponde à coordenada latitude do sensor.
- **long:** propriedade do tipo *double* que corresponde à coordenada longitude do sensor.

C *Sensores de Pressão Atmosférica*

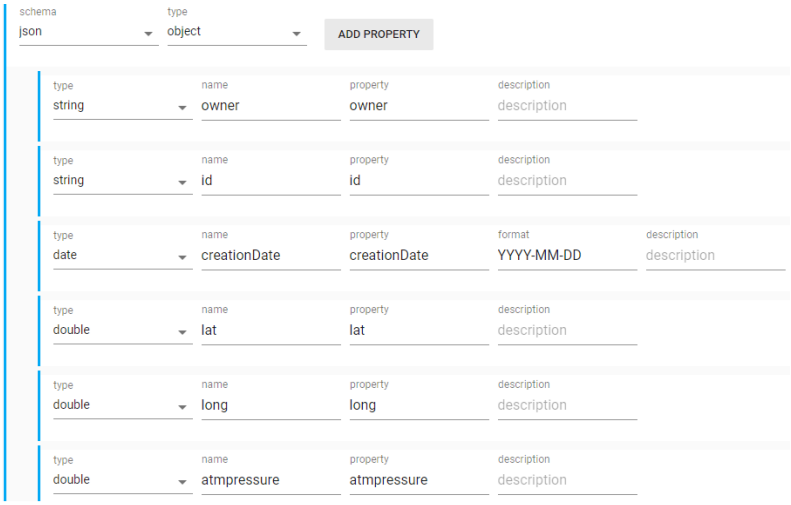


Figura 4.6: Esquema dos conjuntos de dados de sensores pressão atmosférica

- **owner:** propriedade do tipo *string* que representa o proprietário do sensor.
- **id:** propriedade do tipo *string* que representa a identificação do sensor.
- **creationDate:** propriedade do tipo *date* que representa a data de criação dos dados do sensor.
- **lat:** propriedade do tipo *double* que corresponde à coordenada latitude do sensor.
- **long:** propriedade do tipo *double* que corresponde à coordenada longitude do sensor.
- **atmpressure:** propriedade do tipo *double* que representa valores de pressão atmosférica medidos pelo sensor.

D *Sensores Dinâmicos*

Quanto a esta categoria de sensores, o procedimento foi diferente dos restantes. Uma vez que com estes sensores o pretendido seria simular percursos realizados por trotinetas elétricas de um determinado ponto A até um ponto B, o primeiro passo a tomar foi a obtenção das coordenadas dos percursos. Para tal, recorremos a uma ferramenta do Google Maps que permite desenhar rotas específicas e exportar todos os dados relativos a cada rota para um ficheiro *Keyhole Markup Language* (KML). Estes ficheiros contêm informações como o nome da rua onde se situa o ponto inicial, assim como o nome da rua do ponto final do percurso. Também é constituído pelo conjunto de coordenadas de latitude e longitude do percurso.

Sendo que uma das *features* deste simulador é a possibilidade de *upload* de dados previamente criados em formato JSON ou *Comma Separated Value* (CSV), colocámos as coordenadas dos percursos pretendidos num ficheiro JSON, juntamente com informações como o identificador de cada percurso, o proprietário e a data em que foram realizados. Decidimos que todos os percursos iriam pertencer a apenas a um proprietário, sendo este uma empresa especializada na área. Ao realizarmos o *upload* do ficheiro, o esquema JSON do simulador fica automaticamente preenchido com campos presentes no ficheiro. A representação do

esquema é apresentada na Fig 4.7. Na Fig 4.8 temos a primeira posição do *array* das coordenadas do percurso, seguida da descrição dos atributos que o constituem.

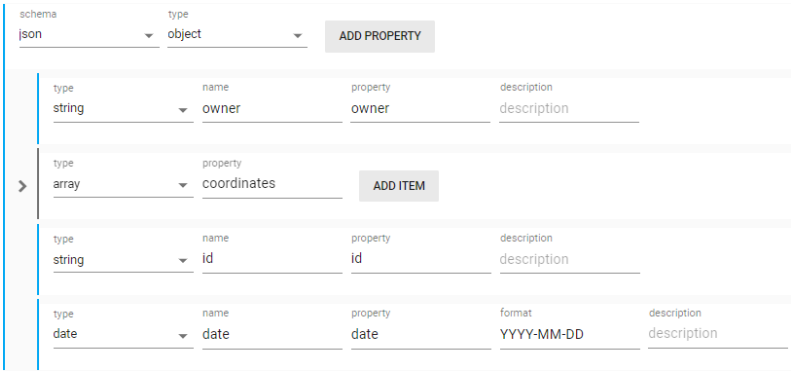


Figura 4.7: Esquema dos conjuntos de dados de sensores dinâmicos

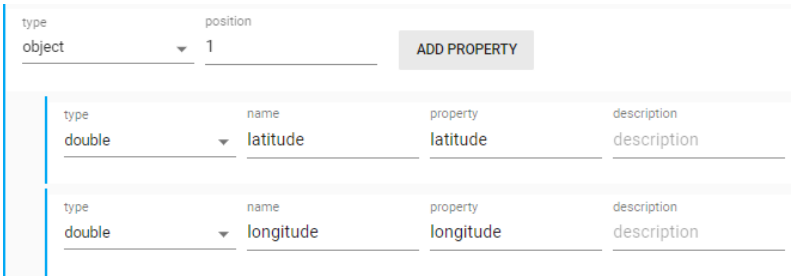


Figura 4.8: Elementos do *array* coordinates

- **owner:** propriedade do tipo *string* que representa o proprietário do percurso.
- **coordinates:** propriedade do tipo *array* de objetos que contem o conjunto de todas as coordenadas do percurso.
- **latitude:** propriedade do tipo *double* que corresponde à coordenada latitude de cada um dos pontos que constituem o percurso.
- **longitude:** propriedade do tipo *double* que corresponde à coordenada longitude de cada um dos pontos que constituem o percurso.
- **id:** propriedade do tipo *string* que representa a identificação do percurso.
- **date:** propriedade do tipo *date* que representa a data de realização do percurso.

Após uma explicação de como são criadas as sessões, bem como os conjuntos de propriedades de cada uma, vamos abordar a conceção dos sensores em si. Dentro de cada sessão é possível criar uma série de sensores que seguem as propriedades anteriormente atribuídas à sessão. A cada um dos sensores, atribuímos propriedades únicas como a sua identificação, o seu proprietário e coordenadas de latitude e longitude. Decidimos criar oito sensores na sessão de temperatura, oito sensores na sessão de humidade relativa e dez sensores na sessão de pressão atmosférica. Quanto à sessão de sensores dinâmicos não procedemos à criação de sensores pela facto dos conjuntos de dados desta sessão serem previamente criados e carregados no simulador. Como já

foi referido anteriormente, todos estes sensores têm coordenadas diferentes correspondentes a locais da cidade do Porto. Na Fig 4.9 podemos observar como é criado um sensor, neste caso um sensor de temperatura.

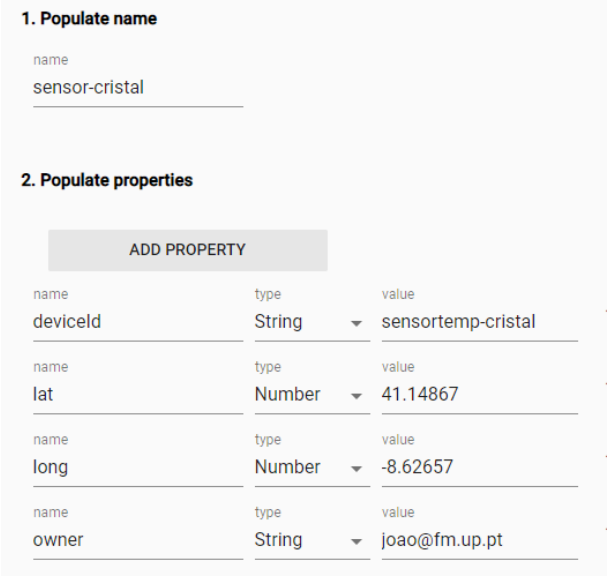


Figura 4.9: Exemplo de propriedades atribuídas a um sensor da sessão de temperatura.

Uma vez criados os sensores pretendidos, é necessário associar as suas propriedades às propriedades inicialmente criadas para cada sessão, de forma a que seja possível produzir os conjuntos de dados necessários. Para tal, a cada propriedade da sessão, é feita a correspondência da respetiva propriedade dos sensores, à exceção dos valores que queremos que sejam gerados aleatoriamente, como a temperatura, humidade relativa e pressão atmosférica. Na Fig 4.10 podemos observar que a todas as propriedades atribuímos as respetivas propriedades dos sensores, à exceção do campo temperature, com valores compreendidos entre 15 e 20 e do campo creationDate que corresponde à data em que os conjuntos de dados são criados. Este procedimento foi igual para todas as sessões. No entanto, escolhemos valores entre 20 e 40 para sensores de humidade relativa e 990 e 1010 para sensores de pressão atmosférica. Optámos por estes intervalos de valores para que a discrepância dos valores gerados não fosse tão acentuada e fossem mais próximos do real possível. No que diz respeito à associação das propriedades dos sensores dinâmicos com a respetiva sessão, esta é realizada automaticamente aquando do *upload* dos conjuntos de dados no simulador.

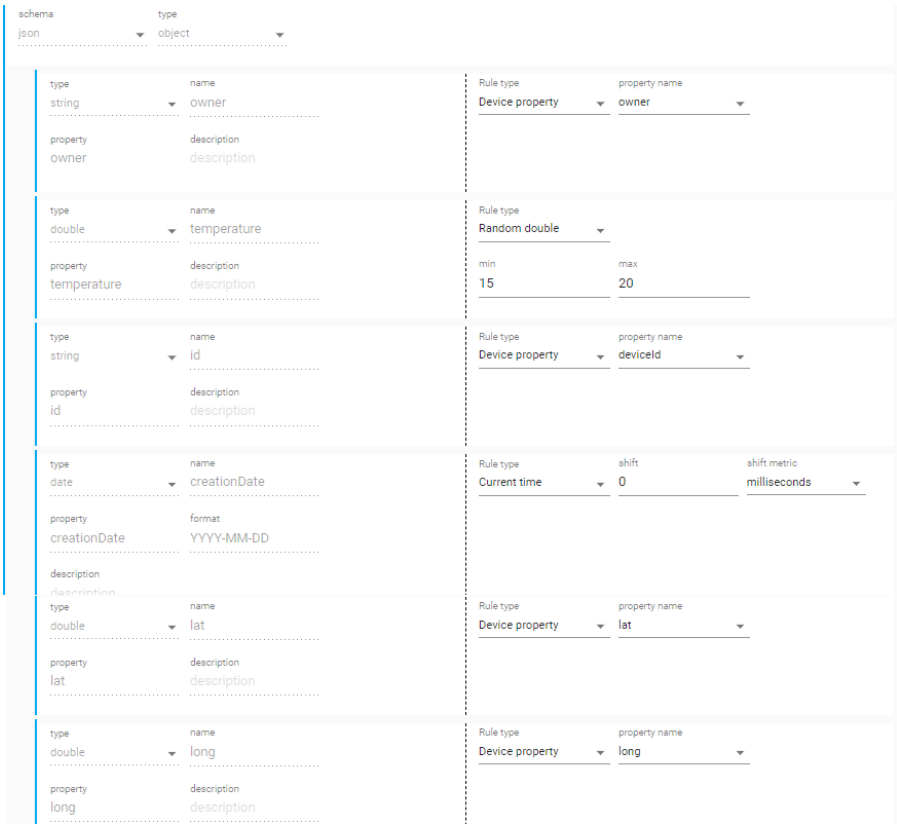


Figura 4.10: Valores atribuídos às propriedades da sessão de temperatura

Por fim, este simulador tem uma *feature* muito útil que nos ajudou na escolha do mesmo, para o desenvolvimento do nosso projeto. Possibilitou-nos o envio de mensagens com os conjuntos de dados gerados para os tópicos de um sistema de mensagens *publish-subscribe*, posteriormente criados. Estes tópicos, serão enumerados e descritos na secção seguinte. Na etapa final de configuração de cada sessão, existe uma secção na qual escolhemos a tecnologia que desejamos utilizar. Nessa secção, há um campo no qual indicamos o *Uniform Resource Locator* (URL) com endereço *Internet Protocol* (IP), número da porta e também outro campo onde colocamos o nome do tópico para onde pretendemos enviar os dados.

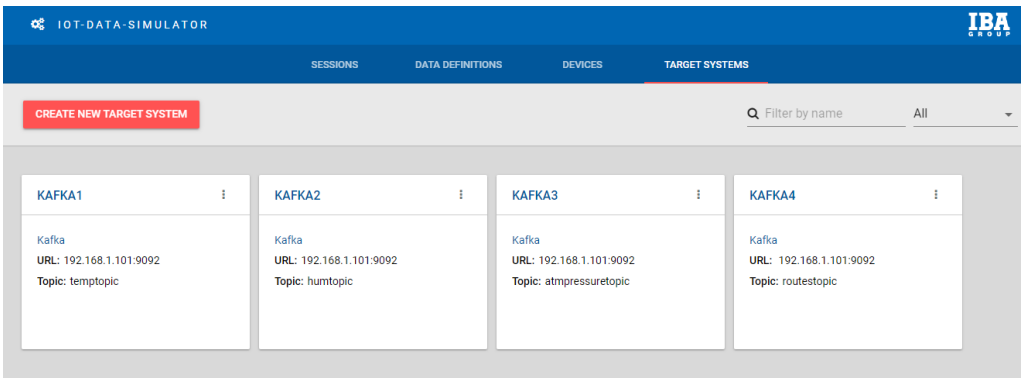


Figura 4.11: Integração do simulador com o *message broker*

4.2.0.2 *Message Broker*

Quando falamos num sistema de mensagens *publish-subscribe*, podemos dizer que o *message broker* é o elemento intermediário entre as restantes partes. Por outras palavras, é aquele que estabelece a comunicação e permite a troca de mensagens entre os produtores (*publishers*) e os consumidores (*subscribers).* Mais especificamente, no nosso projeto, este componente facilita a troca de conjuntos de dados entre o simulador de dados e o componente consumidor que será abordado na secção seguinte. Recebe os dados provenientes do simulador, armazena-os nos respetivos tópicos que criámos a partir dele e encaminha-os para o consumidor.

Quanto à tecnologia utilizada para desenvolver este componente, do conjunto de tecnologias oferecidas pelo simulador e tendo em conta apenas as que são sistemas de mensagens *publish-subscribe*, escolhemos o Kafka. O Kafka, é uma plataforma de transmissão de eventos. Basicamente, é uma plataforma responsável pela recolha de fluxos de dados em tempo-real a partir de milhares de fontes como por exemplo, sensores, bases de dados, dispositivos móveis e serviços de *cloud*. Também é responsável pelo armazenamento dos mesmos para posterior utilização e pelo processamento tanto dos anteriormente guardados, ou seja, dos mais antigos, como dos recolhidos em tempo-real. Por último, pode-se dizer que também é responsável pelo envio destes fluxos de dados para as diferentes partes interessadas [47]. Uma caraterística chave do Kafka é o fato dos produtores e consumidores não comunicarem diretamente entre si, o que torna esta tecnologia altamente escalável. Relativamente ao Kafka *cluster*, este pode ter apenas um ou múltiplos *brokers*. Quanto ao tópicos, são divididos em partições num único *broker* ou podem ser distribuídas por vários *brokers* [47]. Optámos então pela utilização desta tecnologia por diversos motivos: primeiro porque é uma das tecnologias com maior utilização em casos de uso semelhante; segundo porque é extremamente eficaz em termos de *performance*, o que era algo que nós prezamos num sistema do género; e por último, porque já detínhamos um pouco de conhecimento anterior relativo a esta tecnologia.

No nosso projeto temos um Kafka *broker* no qual criámos quatro tópicos com uma partição, denominados temptopic, humtopic, atmpressuretopic e routestopic. Para cada um deles são publicados e armazenados os conjuntos de dados provenientes da respetiva sessão criada no simulador de dados. No primeiro tópico referido são publicados todos os dados referentes aos sensores de temperatura enquanto que no segundo, todos os dados de sensores de humidade relativa, no terceiro tópico os dados de pressão atmosférica e por fim, no quarto tópico os dados relativos aos percursos realizados por trotinetas elétricas.

4.2.0.3 Consumidor

O consumidor é o terceiro e último elemento do sistema de mensagens *publish-subscribe* e, dessa forma, é o responsável por subscrever os tópicos e consumir os dados de cada um deles. Este componente tem como principal objetivo o processamento e a persistência dos conjuntos de dados na base de dados. Neste componente, decidimos a estrutura que queríamos que os dados

armazenados na base de dados apresentassem. Através dele, manipulámos os dados criados pelo simulador, modificando nomes de variáveis e adicionando novos campos necessários. Para além destas caraterísticas, o consumidor também estabelece a comunicação com o componente de controlo de acesso para conseguir recuperar informações armazenadas na base de dados deste último, fundamentais em passos futuros do nosso projeto.

Escolhemos a linguagem Python para desenvolver este componente uma vez que é uma linguagem de programação bastante simples, de fácil compreensão e também por ter vários módulos que nos ajudaram na interação entre o consumidor e as diferentes partes do sistema. Para a implementação do consumidor, utilizámos os módulos PyMongo para podermos interagir com a base de dados, o kafka-python para podermos aceder aos diferentes tópicos do kafka e por último o firebase_admin para a interação com o componente de controlo de acesso como administrador.

Quanto ao processamento dos dados provenientes do *message broker*, estruturámos os dados da maneira que achámos mais conveniente. Alterámos os nomes de algumas variáveis para serem mais percetíveis e criámos novos campos que achámos convenientes. Neste componente, agrupámos os dados provenientes de cada tópico em coleções, formando, assim, uma coleção na base de dados para cada tipo de sensor. Dentro das coleções de sensores ambientais, os conjuntos de dados são inseridos sob uma estrutura em *buckets*. Esta estrutura pode ser vista como baldes de informação em que cada balde está associado a um sensor e todas as medições realizadas pelo sensor vão para o balde correspondente. Mais concretamente, cada vez que uma medição é gerada para um determinado sensor, vai ser agrupada juntamente com as anteriores.

Decidimos também, agrupar de uma forma global os dados de cada tipo de sensor, isto é, agregámos as medições geradas por todos os sensores de determinado tipo. De uma forma simplista, agrupámos todas as medições de temperatura, independentemente do seu sensor de origem. Este processo é semelhante para todos os sensores ambientais. O objetivo pelo qual decidimos proceder a este processo de agregar os dados, foi para que conseguíssemos oferecer informações estatísticas de uma determinada região, no nosso caso, de toda a região do Porto.

Inicialmente, o principal objetivo deste componente era consumir os dados provenientes do simulador. No entanto, tivemos de lhe incumbir a realização de tarefas para a segunda etapa deste projeto, ou melhor dizendo, a construção da plataforma de comercialização. No momento de decidirmos como iríamos atribuir as permissões de acesso aos dados dos sensores por parte utilizadores, achámos que através das organizações aos quais os utilizadores pertencem poderia ser uma boa abordagem. Todas as informações relativas a um utilizador são conseguidas no momento do seu registo na plataforma e são armazenadas na base de dados do componente de controlo de acesso. Assim, tivemos de estabelecer a comunicação entre o consumidor e o componente de controlo de acesso, para conseguirmos recuperar essas informações, como é o caso das organizações.

Criámos o campo organizations para cada sensor, com o intuito de atribuir permissões de acesso aos dados no *data marketplace*, baseadas na organização à qual pertencem. Da mesma,

forma decidimos criar os campos allowLocal e allowInfo. O campo allowLocal foi criado para questões de permissão de acesso à localização exata do sensor após aquisição dos dados do mesmo. Por sua vez, criámos o campo allowInfo para que o proprietário do sensor pudesse conceder o acesso a dados estatísticos ou dados não processados às organizações. No consumidor criámos também outro campo denominado buyUser para guardar os compradores dos dados do sensor. Quanto aos sensores dinâmicos, apenas foi criado o campo organizations e buyUser. Os restantes campos não foram criados pelo facto de estes sensores apenas oferecerem um tipo de dados e após a sua compra, os compradores terem acesso ao conjunto de coordenadas que constituem o percurso. Na secção 4.2.1, iremos analisar com mais detalhe a estrutura final dos dados resultante de todo o processamento a que foram sujeitos neste componente e na secção 4.3.1 iremos apresentar detalhadamente todo o processo de controlo de acesso baseado nestes campos.

No bloco de código A.1 podemos ver como são recuperadas as organizações contidas na base de dados do componente de controlo de acesso. Em A.2 temos a modificação de algumas variáveis. A criação dos novos campos e inserção dos dados de temperatura na base de dados. O processo é igual para os restantes tipos de sensores ambientais. No bloco de código A.3 temos a inserção dos dados de trotinetas elétricas na base de dados.

4.2.1 Base de Dados

Para tornar os conjuntos de dados gerados persistentes, utilizámos uma base de dados. Como prezamos pela *performance* do sistema, optámos por utilizar uma base de dados não relacional, também denominada NoSQL. Outra razão para esta escolha prendem-se com o facto de parte do nosso projeto depender da geração de dados em tempo real e estas bases de dados serem caracterizadas por suportar e lidar com grandes quantidades de dados. A principal diferença entre estas e as bases de dados relacionais é o facto de não usarem tabelas como estrutura para armazenar os dados, não seguirem um esquema pré-definido e serem muito eficientes com dados não estruturados, como por exemplo, documentos word e pdf, imagens, ficheiros de vídeo [64].

As bases de dados NoSQL são uma melhor opção quando é necessário suportar grandes quantidades de dados pelo facto de terem um sistema distribuído no qual se adicionam facilmente mais servidores. Nas bases de dados relacionais, as grandes quantidades de dados, podem-se tornar num problema a nível económico, por serem necessários mais recursos de *hardware*, o que é bastante dispendioso. A nível de desempenho, uma vez que está diretamente relacionado com a complexidade das tabelas. Isto é, quanto maior a quantidade de dados mais tabelas são utilizadas e mais informação nelas contidas, o que leva a um aumento no tempo de execução das *querys* [69].

No nosso trabalho, utilizámos uma base de dados não relacional baseada em documentos. Estas bases de dados são caraterizadas por armazenarem os dados em documentos em formato JSON, *Binary JavaScript Object Notation* (BSON) ou *Extensible Markup Language* (XML). Relativamente aos documentos, estes são guardados em coleções. Uma das vantagens destas bases

de dados não relacionais, e como já foi anteriormente referido, é o facto de não ser necessário seguirem uma estrutura específica. Quer isto dizer que, os documentos de uma mesma coleção não têm necessariamente de ter todos os mesmos campos e da mesma forma, os campos podem ter tipos de dados também diferentes [67].

A base de dados que escolhemos para o desenvolvimento do nosso projeto foi o MongoDB. Um dos motivos que nos levou a escolher o MongoDB para armazenamento dos dados, foi o facto de ser uma tecnologia bastante utilizada nos dias de hoje e, dessa forma, haver muita documentação sobre o seu funcionamento. Outro motivo que nos levou à escolha desta base de dados foi pelo facto de correr numa plataforma em nuvem, denominada MongoDB Atlas. Esta plataforma funciona como uma base de dados e fornece serviços em nuvem de forma a simplificar a gestão dos nossos dados, como é o caso do Database Triggers que iremos abordar mais à frente [66].

A nossa base de dados começou por ser composta por cinco coleções. Quatro delas referentes a cada tipo de sensor, ou seja, temperatura, humidade relativa, pressão atmosférica e percursos realizados por trotinetas elétricas e, por último, a coleção que contem as medições agregadas independentemente do seu sensor de origem, por tipo de sensor. No entanto, com o decorrer do projeto, precisámos de criar uma sexta coleção para podermos guardar as transações que ocorrem no *data marketplace*. Isto é, a atividade do utilizador na plataforma, como ações exercidas sobre os dados dos sensores na plataforma de comercialização, como ações de compra e venda. Depois de uma breve explicação de como está distribuída a base de dados, vamos abordar de seguida, como representámos os conjuntos de dados de cada coleção.

4.2.1.1 Estrutura dos dados produzidos pelo simulador

Relativamente à representação dos conjuntos de dados, e como já foi mencionado em 4.2.0.3, achámos que seria uma boa ideia agrupar os dados provenientes do simulador em *buckets*. Cada documento tem os conjuntos de dados relativos a um sensor e cada vez que novas medições são produzidas, são inseridas num *array* nesse mesmo documento, prevenindo que novos documentos para um mesmo sensor sejam gerados cada vez que uma medição é produzida. Decidimos recorrer a este método por questões de organização dos dados, pois assim, não temos informações de cada sensor "espalhadas"pela base de dados e a realização de *querys* também se torna mais simples, assim como realizar cálculos estatísticos.

Inicialmente e ainda na fase de produção dos conjuntos de dados ambientais por parte do simulador, escolhemos propriedades que achámos que seriam as que representavam de forma mais real os dados, como é o caso de um campo para identificação do sensor, as coordenadas de latitude e longitude para efeitos de localização, a data de geração dos dados e também um campo que indicasse o proprietário do sensor. Posteriormente, no consumidor, criámos o campo samples, correspondente a um *array*, onde são inseridas as medições à medida que são produzidas. Contudo, à medida que o nosso trabalho se foi desenvolvendo, foi necessário criar novos campos. Criámos um *array* para guardar a identificação dos compradores, uma vez que, os dados do sensor podem ser adquiridos por vários compradores. Criámos também um *array* de objetos

designado organizations, em que a chave corresponde a cada uma das organizações provenientes do componente controlo de acesso, e que o valor é um booleano, em que *true* significa que os utilizadores pertencentes a essa mesma organização tem permissões para aceder aos dados do sensor e *false* caso contrário. A mesma lógica foi pensada para o *array* allowLocal, no entanto, o valor *true* representa o acesso à localização exata do sensor após a aquisição dos seus dados e *false* para a negação do acesso. Tal como estes dois *arrays*, também o allowInfo segue o mesmo padrão, com a distinção de que aqui há uma escolha no tipo de informação que o sensor oferece. No caso do valor ser *true*, os utilizadores das respetivas organizações têm acesso às medições produzidas pelo sensor, enquanto que se o valor for *false*, o acesso é centrado em informações estatísticas.

Tipo	**Nome da variável**	**Função**
Object	allowLocal	Permissões de acesso à localização do sensor após a compra dos dados.
Object	allowInfo	Permissões para o tipo de informações que o sensor oferece.
Array	buyUser	Compradores dos dados do sensor.
String	creationDate	Data de criação dos dados do sensor.
String	deviceId	Identificação do sensor.
Object	organizations	Permissões de acesso aos dados do sensor.
String	owner	Proprietário do sensor.
Array	samples	Medições realizadas.
Double	latitude	Coordenada de latitude da localização do sensor.
Double	longitude	Coordenada de longitude da localização do sensor.

Tabela 4.1: Estrutura dos dados de sensores ambientais

Os conjuntos de dados que decidimos agrupar por região têm um estrutura muito simples. Atribuímos-lhe um identificador para conseguirmos identificar a que tipo de sensor pertencem. Como estes conjuntos de dados foram concebidos para a oferta de informações estatísticas por região, não temos apenas um sensor de origem, mas sim os múltiplos sensores que os originaram. Dessa forma, teríamos de conseguir uma maneira de os representar e a que nos pareceu mais correta, foi através de um marcador representativo centrado em coordenadas de latitude e longitude aleatórias dentro da região do Porto. Criámos também um campo samples onde são inseridas as medições dos sensores à medida que são produzidas e um campo creationDate que corresponde à data de geração dos dados. Tal como todos os outros conjuntos de dados, também a estes tivemos de lhe atribuir um campo buyUser.

Tipo	**Nome da variável**	**Função**
String	id	Identificação do tipo de dados.
Array	buyUser	Compradores dos dados.
String	creationDate	Data de criação dos dados.
Array	samples	Medições realizadas.
Double	latitude	Coordenada de latitude onde se situa o marcador.
Double	longitude	Coordenada de longitude onde se situa o marcador.

Tabela 4.2: Estrutura de dados estatísticos por região

Quanto aos dados dos sensores dinâmicos, o identificador do percurso, data de realização, proprietário e as coordenadas que constituem cada um, foram criados e inseridos no simulador. No entanto no consumidor procedemos à criação do campo buyUser e organizations.

Tipo	**Nome da variável**	**Função**
String	id	Identificação do percurso.
String	date	Data de realização do percurso.
String	owner	Proprietário do percurso.
Array	buyUser	Compradores dos dados do percurso.
Array	organizations	Permissões de acesso aos dados do percurso.
Array	coordinates	Conjunto das coordenadas de latitude e longitude que constituem o percurso.

Tabela 4.3: Estrutura dos dados de sensores dinâmicos

4.2.1.2 Estrutura das transações

Uma vez que as transações correspondem às ações desempenhadas pelos utilizadores na plataforma, a estrutura destas varia. A informação relativa à venda de dados de um sensor é apresentada para todos os utilizadores com permissão de acesso ao sensor, enquanto que informação de compra de dados é apresentada para o proprietário do sensor e para o comprador. Para o registo da compra e venda de dados de um sensor, bem como para registo da quantia transferida para o vendedor, a transação segue a seguinte estrutura:

Tipo	Nome da variável	Função
String	sensor	Identificação do sensor.
String	type	Tipo do sensor.
String	seller	Identificação do proprietário do sensor.
String	buyer	Identificação do comprador do sensor.
String	buystring	Registo de compra de dados do sensor visível para o vendedor.
String	sellstring	Registo de venda de dados do sensor visível para os utilizadores com acesso ao sensor.
String	amountstring	Registo de quantia transferida visível para o vendedor.
String	buyerstring	Registo de compra de dados do sensor visível para o comprador.
String	sellerstring	Registo de venda de dados do sensor visível para o vendedor.

Tabela 4.4: Estrutura do registo de ações de compra, venda e quantia transferida

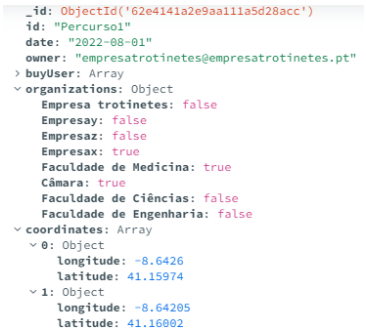
Quando um comprador exporta os dados do sensor adquiridos, também esta ação é registada e apresentada para o vendedor e comprador. A estrutura desta transação é um objeto composto por quatro parâmetros.

Tipo	Nome da variável	Função
String	sensor	Identificação do sensor.
String	type	Tipo do sensor.
String	seller	Identificação do proprietário do sensor.
String	buyer	Identificação do comprador do sensor.
String	exportsellerstring	Registo de exportação dos dados do sensor visível para o vendedor.
String	exportbuyerstring	Registo de exportação dos dados do sensor visível para o comprador.

Tabela 4.5: Estrutura de dados do registo de exportação de dados do sensor

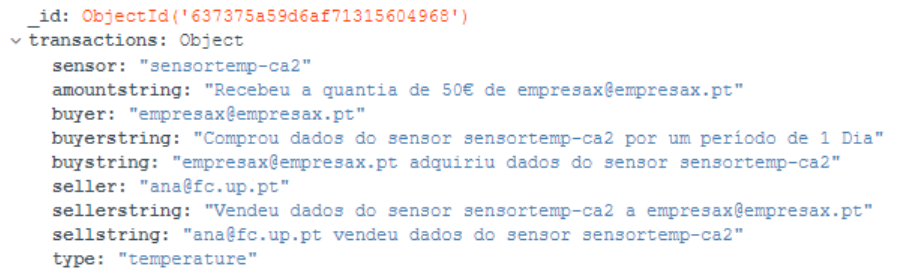
O MongoDB apresenta uma interface gráfica, o MongoDB Compass, que nos permite visualizar todo o conteúdo da base de dados. Nas Fig 4.12a e 4.12b estão representadas as estruturas de dados de um sensor de humidade relativa e de um percurso realizado por trotinetas elétricas, respetivamente. Todos os sensores de todas as coleções, exceto da coleção dos percursos, seguem a mesma estrutura, alterando apenas o *array* de *samples* consoante o tipo de dados recolhido pelo sensor. Na Fig 4.13 temos o exemplo da diferentes estruturas dos registos.

(a) Estrutura dos dados de um sensor de humidade relativa

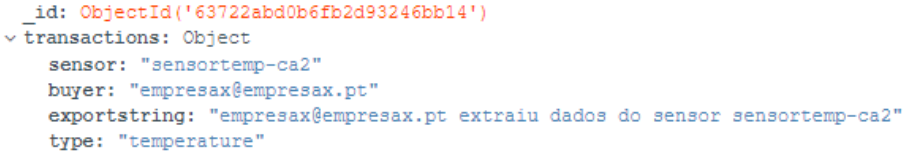


(b) Estrutura dos dados de um percurso de trotinetas elétricas

Figura 4.12: Representação da estrutura dos dados na base de dados.



(a) Exemplo da estrutura de registo de compra, venda e quantia transferida



(b) Exemplo da estrutura de registo de exportação de dados

Figura 4.13: Exemplos de registo de transações

Voltando agora aos serviços de gestão de dados do MongoDB Atlas, o Database Triggers foi uma peça fundamental no nosso trabalho. Este é uma ferramenta poderosa que permite a execução de funções quando certos eventos ocorrem na base de dados, por exemplo, quando um documento é inserido, atualizado ou eliminado. Podem ainda ser programados para executar código em momentos específicos ou num intervalo definido [68]. No nosso projeto, utilizámos este serviço para conseguirmos atualizar um certo campo dos documentos em função de uma condição temporal. Isto é, uma das caraterísticas do nosso *data marketplace* é a compra dos dados dos sensores por períodos de tempo. Cada utilizador interessado nos dados do sensor, pode comprá-los por uma hora, um dia ou uma semana e, uma vez expirado esse tempo, este deixa de ter acesso aos mesmos. Como tal, é necessário que o *array* dos compradores dos dados dos sensores seja atualizado e o respetivo comprador seja eliminado para que perca o acesso aos dados no fim do tempo estipulado. Para isso, programámos um trigger no MongoDB Atlas que corre de minuto em minuto a função com a condição de retirar o respetivo comprador do *array*

com base no tempo de subscrição dos dados escolhido por ele. Na Fig 4.14 temos um exemplo de como é atualizado o campo buyUser após a aquisição dos dados do sensor pelo comprador. É adicionado ao *array* um objeto com a identificação do comprador e respetivo tempo de acesso aos dados em *timestamps*. Os *timestamps* apresentados na figura correspondem a aquisição por um período de um dia. Após expiração do tempo, o objeto é eliminado do array. Na Fig 4.15 temos um exemplo do intervalo de execução bem como a função executada para retirar o comprador do campo buyUser.

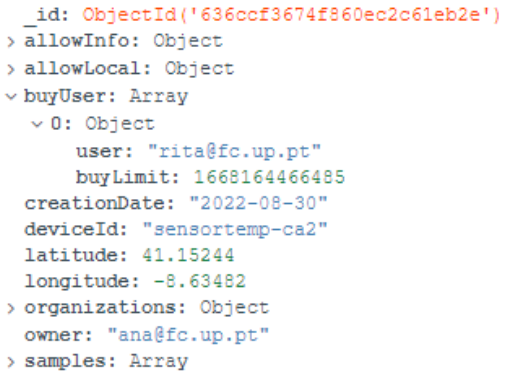
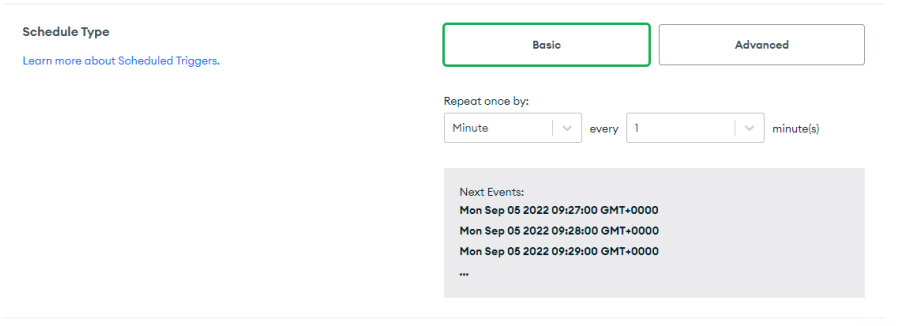
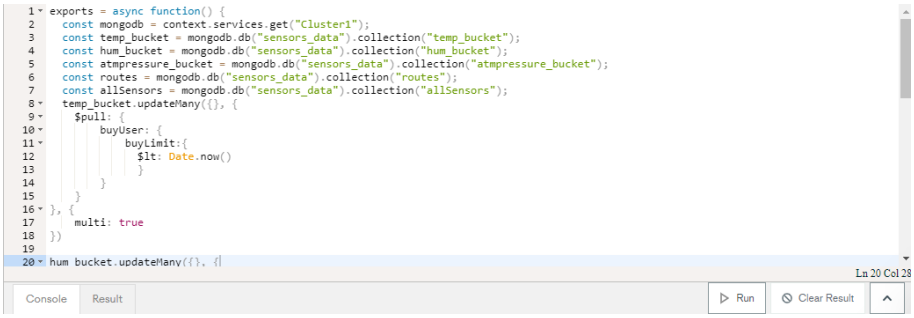


Figura 4.14: Exemplo do campo buyUser após aquisição dos dados do sensor



(a) Intervalo de tempo da execução da função



(b) Função para atualização do campo buyUser

Figura 4.15: *Trigger* para atualização do campo buyUser

4.3 Desenvolvimento de um *Data Marketplace*

Após todo o processo de produção dos dados dos sensores IoT, temos o desenvolvimento da plataforma onde são comercializados. Para que esta fosse devidamente concebida, registos e formas de autenticação dos utilizadores tiveram de ser pensadas, assim como estabelecer regras para atribuir permissões de acesso aos utilizadores. Também tivemos de pensar em algumas técnicas que de alguma forma ocultassem os dados do utilizador a um nível da interface do utilizador. Outro ponto fundamental foi como seriam feitas as comunicações entre os diferentes componentes do nosso sistema. Para isso, recorremos a *web services* anteriormente abordados em 2.4.

De seguida iremos apresentar bem como descrever todos os componentes pertencentes a esta etapa, bem como as tecnologias utilizadas para o seu desenvolvimento. O componente de controlo de acesso é apresentado em 4.3.1, por sua vez, a API na subsecção 4.3.2, o componente de agregação e controlo de acesso em 4.3.3 e, por fim, o *data marketplace* na subsecção 4.3.4.

4.3.1 Controlo de Acesso

O componente de controlo de acesso é o responsável pela autenticação e autorização dos utilizadores. Trata do aprovisionamento das contas dos utilizadores, ou seja, o registo destes na plataforma. No processo de autenticação, este verifica a identidade do utilizador comparando os dados fornecidos com os dados armazenados. Por outras palavras, através da autenticação um utilizador prova ser quem realmente é. Este componente também é responsável pela autorização e controlo de acesso. Assegura que o utilizador tem permissão para aceder aos recursos que pretende. Saber quais são as intenções dos utilizadores é fundamental para que lhes possam ser concedidas apenas as permissões necessárias.

4.3.1.1 Google Firebase

Para a implementação do nosso componente de controlo de acesso optámos por utilizar a Google Firebase, pelo facto de oferecer um serviço de autenticação e autorização bastante intuitivo. Permitiu-nos implementar o nosso sistema de controlo de acesso de uma forma muito simples e rápida. Esta plataforma desenvolvida pela Google, fornece também dois tipos de bases de dados não relacionais. A Cloud Firestore e a Realtime Database. A Realtime Database é uma base de dados na qual os dados são armazenados numa árvore JSON enquanto que a Cloud Firestore é baseada em documentos [27]. Este foi o principal motivo que nos levou a utilizar esta Cloud Firestore pelo facto de já estarmos familiarizados com a estrutura de coleções e documentos, uma vez que utilizámos também esta estrutura no MongoDB para armazenamento de dados dos sensores.

Quanto ao Firebase Authentication, este oferece diversas formas de autenticação, que tanto

podem ser através do tradicional e-mail e palavra-passe, como por número de telemóvel e por meio de diversos fornecedores de identidade como Google, Facebook, GitHub, etc [28]. No nosso caso, o processo de autenticação é feito por meio de autenticação com e-mail e palavra-passe. Quando um utilizador se regista na aplicação, o seu e-mail fica guardado na Firebase Authentication juntamente com um identificador único e informações adicionais como data de criação e última vez que efetuou *login*. O Firebase Authentication providencia também a gestão de *tokens* permitindo a geração de *tokens JSON Web Token* (JWT), mencionados em 2.6.2. Após a autenticação, estes podem ser utilizados para atribuir ou negar o acesso a um determinado recurso ao utilizador [29].

Podemos dizer que este processo é bastante simples. As credenciais, neste caso e-mail e palavra-passe do utilizador, são enviadas para o componente de controlo de acesso, sendo que, este verifica se estão corretas e coincidem realmente com o utilizador. Caso isto se verifique, o utilizador é autenticado com sucesso e é enviado um *token* JWT de acesso como resposta, caso contrário é enviado um aviso de que as credenciais não estão corretas e, dessa forma, não se pode concluir a autenticação. O processo de autorização decorre quando o utilizador tenta aceder a algum recurso da aplicação. No nosso caso, o acesso a esses recursos é feito por meio de uma API, que será abordada mais à frente. Neste processo é enviado o *token* JWT recebido na autenticação, no cabeçalho de todos os pedidos à API, de forma a que possa ser verificada a identidade o utilizador e se tem, ou não, permissões de acesso àquele recurso.

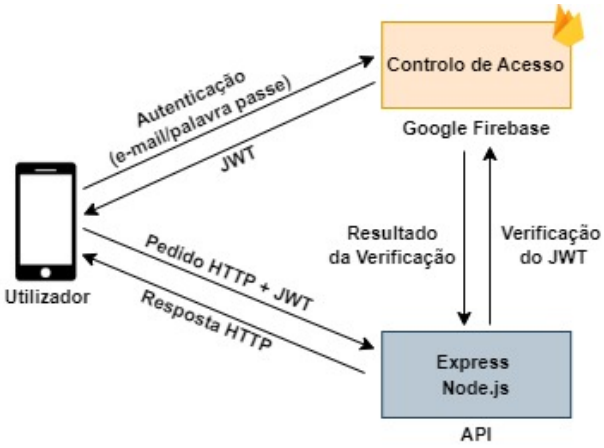


Figura 4.16: Autenticação/Autorização.

4.3.1.2 Aprovisionamento

Na fase de aprovisionamento, atribuímos determinadas caraterísticas ao utilizadores para que, uma vez autenticados na plataforma, lhes sejam concedidas as permissões e acessos necessários. Dessa forma, no momento de registo do utilizador na plataforma, desenvolvemos parâmetros, que este é obrigado a fornecer. Primeiramente, para efeitos de autenticação e, uma vez que a identificação é o primeiro passo do controlo de acesso, é pedido ao utilizador um e-mail e a criação de uma palavra-passe. O objetivo do seu registo no *data marketplace*, ou seja, se a sua participação na plataforma se destina a comprar, vender dados ou ambos também é requerido.

Outra informação que pretendemos obter dos utilizadores, é a organização à qual pertencem. Estas informações ficam todas armazenadas na base de dados do controlo de acesso. Esta base de dados é a Cloud Firestore. No que diz respeito à forma como estruturámos esta base de dados, temos uma coleção *users* que contém documentos relativos a cada um dos utilizadores. O identificador de cada documento é precisamente o identificador único do respetivo utilizador. A estrutura de dados dos documentos varia caso o utilizador registado seja particular ou empresa como será demonstrado de seguida. Na Fig 4.17 temos um documento relativo a um utilizador particular, que contém os seguintes campos:

- **buySell:** objetivo do utilizador no *data marketplace*. Vender, comprar ou ambos.
- **e-mail:** e-mail do utilizador.
- **firstName:** Primeiro nome do utilizador.
- **lastName:** Último nome do utilizador.
- ***organization*:** Nome da organização à qual o utilizador pertence.

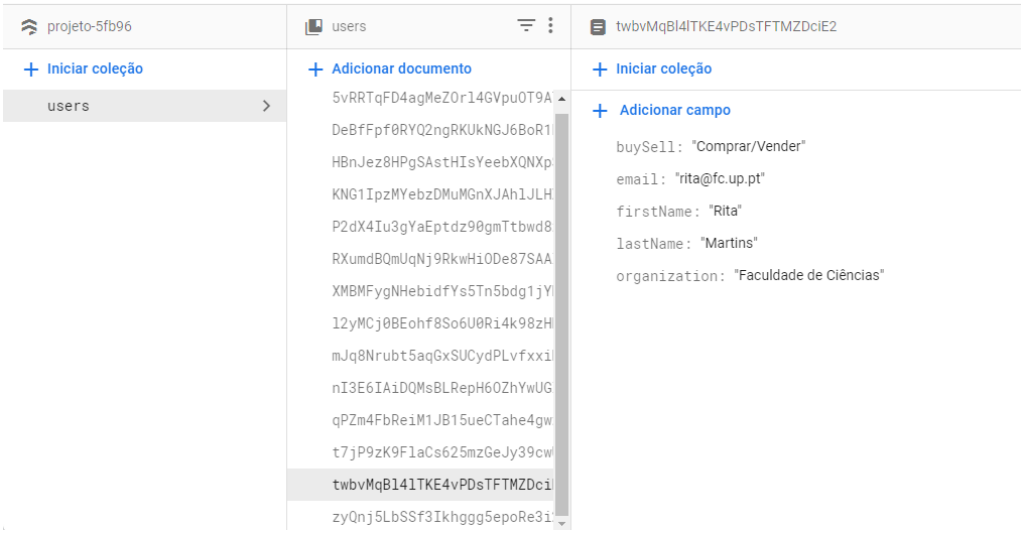


Figura 4.17: Exemplo da estrutura de dados de um documento de um utilizador particular.

Na Fig 4.18 temos um documento relativo a uma empresa e como é possível visualizar apenas tem os campos buySell, e-mail e *organization*.

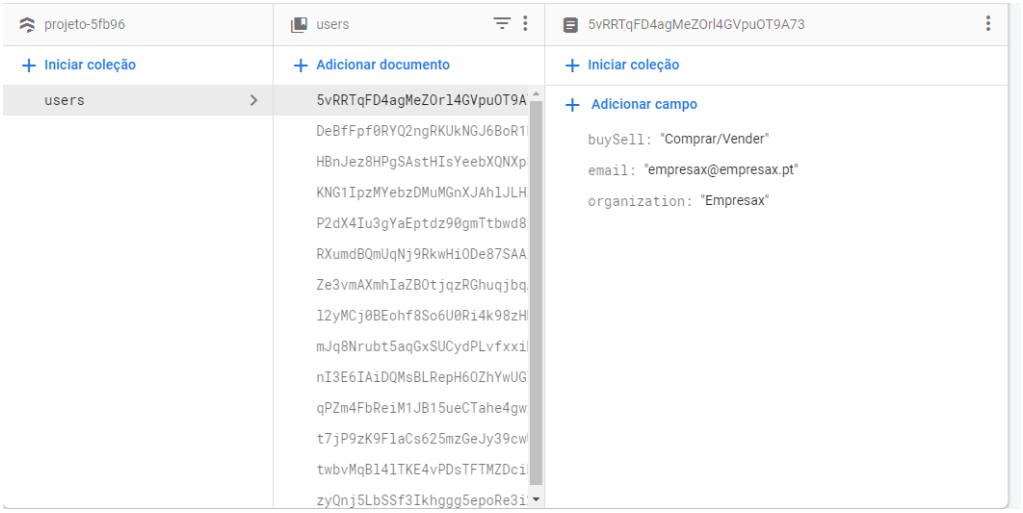


Figura 4.18: Exemplo da estrutura de dados de um documento de uma empresa.

4.3.1.3 Permissões e Acesso aos Dados do *Data Marketplace*

As permissões de acesso aos dados dos sensores são atribuídas a conjuntos de utilizadores, não a utilizadores individualmente. Isto é, quando um utilizador se regista na aplicação, é-lhe requerido a organização à qual pertence e assim que este é autenticado passa a pertencer a essa organização no contexto da aplicação. À medida que novos utilizadores se registam na aplicação com uma organização, herdam as permissões de acesso dessa mesma organização. Assim, as permissões são dadas por organização, ou seja, todos os utilizadores que pertencerem a essa organização têm as mesmas permissões de acesso. Quem estipula quem tem acesso ou não aos dados de um sensor, é o próprio proprietário do mesmo. Os campos organizations, allowLocal e allowInfo foram criados e atribuídos a cada um do sensores para que as permissões de acesso pudessem ser atribuídas. A representação destes campos foi anteriormente descrita em 4.2.1.1 quando foi descrita a estrutura dos conjuntos de dados inseridos na base de dados.

A função que os utilizadores estão dispostos a exercer no *data marketplace* como vendedores, compradores ou ambos também influencia nas permissões de acesso que lhe são atribuídas. Se um utilizador for vendedor, tem acesso a determinadas páginas da aplicação que um comprador não tem e vice-versa. Um vendedor também não tem permissão para adquirir dados de outros utilizadores, estas permissões apenas são dadas se a intenção de um utilizador for comprar ou vender/comprar.

4.3.2 API

De forma a que os diferentes componentes do nosso sistema possam comunicar entre si, mais especificamente, para que o *front-end* possa recuperar dados da base de dados e também para que possa haver verificação de credencias por parte do componente de controlo de acesso, seguimos o principio dos *web services* mencionados em 4.3.

Esta API foi criada para diversos propósitos. Primeiro, para que pudéssemos recuperar informações relacionadas com os diferentes sensores presentes na base de dados, para vários fins no *front-end*. Para tal, recorremos a pedidos *Hypertext Transfer Protocol* (HTTP) GET realizados do lado do cliente. Do lado do servidor, definimos funções que satisfizessem os pedidos do cliente. Também para atualizações que fossem necessárias fazer na base de dados como é o caso de atualizar o campo buyUser, sempre que um utilizador compra os dados relativos a um sensor ou percurso. O mesmo acontece com os campos organizations, allowLocal e allowInfo, quando o proprietário deseja colocar para venda determinado sensor para determinadas organizações ou até para todas, permitir ou negar a visibilidade da localização exata do sensor após os dados adquiridos ou escolher o tipo de informação que desejam que seja apresentada para os utilizadores de cada organização. Nestes cenários apresentados, realizámos pedidos HTTP PATCH do lado do cliente, com as respetivas funções no servidor.

No caso do registo das transações, cada vez que uma transação ocorre, seja relativa à compra ou à venda de dados de um sensor, também a coleção correspondente é atualizada, ou melhor dizendo, são criadas novas entradas na coleção. Para este cenário fizemos pedidos HTTP POST e tal como nos restantes casos descritos, com as respetivas funções na API.

Outra funcionalidade muito importante que tivemos de implementar nesta API, foi a integração com o componente de controlo de acesso para que a partir deste último se pudesse proceder à verificação dos *tokens* de acesso enviados em cada pedido. Para este processo, recorremos ao método verifyIdToken, pré-definido do componente de controlo de acesso. Esta API também está ligada ao componente de agregação e controlo dos dados, para que, quando pedido específicos de cálculos estatísticos vindos do *front-end*, estes possam ser devidamente recuperados.

4.3.2.1 Express

No que toca ao desenvolvimento desta API, decidimos recorrer a uma *framework* concebida para o node.js denominada express. Esta *framework* conhecida por ser minimalista, suporta JavaScript e é constituída por um conjunto de *features* que nos permitiram desenvolver o nosso componente rápido, facilmente sem a necessidade de muitas linhas de código. Oferece um sistema de *routing* que basicamente descreve como o servidor reage ao pedido feito pelo cliente relativo a um *endpoint*. Este processo é feito através da combinação de um dos métodos HTTP com o URL do recurso. Para além do sistema de *routing*, disponibiliza um mecanismo *middleware* que consiste nas funções que vão ser executadas entre o pedido do cliente e a resposta do servidor [22, 23]. A escolha desta *framework* também advém do facto de esta possuir mecanismos pré-definidos para a integração com a base de dados MongoDB. No bloco de código A.4 podemos observar como é realizada a recuperação dos dados dos sensores de temperatura com recurso ao método HTTP GET. Em A.5 temos a inserção de entradas com dados das transações na coleção *transactions*, através de HTTP POST. No bloco de código A.6 podemos verificar como é realizada a atualização do campo buyUser através do método HTTP PATCH. Em A.7 temos a função para verificação dos *tokens* de acesso enviados em cada pedido feito pelo cliente. Por último no bloco de código

A.8 é apresentado um pedido realizado ao componente de agregação e controlo dos dados para obtenção da média de temperatura de cada sensor. Este processo é feito por meio do método HTTP GET, com o URL específico para a média de cada sensor de temperatura. O processo é igual para todos os tipos de sensores e também para os conjuntos de dados por região.

4.3.3 Agregação e Controlo dos Dados

O componente de agregação e controlo dos dados é o responsável pela aplicação de funções de agregação aos dados dos sensores ambientais para que estes possam ser transformados numa forma sumariada e seja possível a oferta de informações estatísticas no *data marketplace*, como anteriormente descrito em 2.5.2 . Neste componente, as medições realizadas por cada sensor de um determinado tipo na região do Porto, são utilizadas para calcular o valor médio, máximo e mínimo resultante de todos os sensores desse tipo. Como descrito em 4.2.0.3, no componente consumidor as medições realizadas, por exemplo, por cada sensor de temperatura, são todas agrupadas independentemente do seu sensor de origem para que neste componente, possamos calcular o valor médio da temperatura assim como os valores máximo e mínimo registados em toda a região do Porto. O mesmo acontece para os restantes tipos de sensores de humidade relativa e sensores de pressão atmosférica.

Para procedermos à realização dos cálculos estatísticos, desenvolvemos uma *Representational State Transfer* (REST) API baseada na *framework* Express. Esta API estabelece a comunicação entre a base de dados e a API principal para que, quando recebe os pedidos provenientes desta última, possa aceder à base de dados e através de *querys* possamos obter todos os valores estatísticos de que necessitamos. De seguida iremos apresentar alguns exemplos de como são realizadas as consultas à base de dados para obtermos os valores estatísticos. No bloco de código A.9 podemos ver como é realizada a *query* para obtenção do valor médio de humidade relativa na região do Porto. Em A.10 temos a *query* para o valor máximo de temperatura registado na região do Porto e no bloco de código A.11 temos a *query* para o valor mínimo de pressão atmosférica registado na região do Porto.

No que concerne ao controlo dos dados por parte do proprietário do sensor, isto é, para que ele possa controlar quais os dados do sensor que serão apresentados ao comprador quando estes são adquiridos pelo mesmo, decidimos também recorrer a cálculos estatísticos. Dessa forma, o proprietário tem a opção de escolher se os dados que são apresentados pelo sensor são as medições geradas consecutivamente ou os dados processados a partir de métodos estatísticos. Aqui, o processo de obtenção dos valores estatísticos também foi realizado através de consultas feitas pela API concebida para esse propósito. Nos blocos de código A.12, A.13 e A.14 temos representado como são realizadas as *querys* à base de dados para obtermos a média da temperatura, o valor máximo de humidade relativa e o valor mínimo de pressão atmosférica, para cada sensor, respetivamente.

Desta forma, a diversidade de informações que se podem adquirir na plataforma é maior, podendo ir de dados não processados a dados estatísticos de sensores, como também informações

estatísticas de toda a região do Porto.

4.3.4 *Data Marketplace*

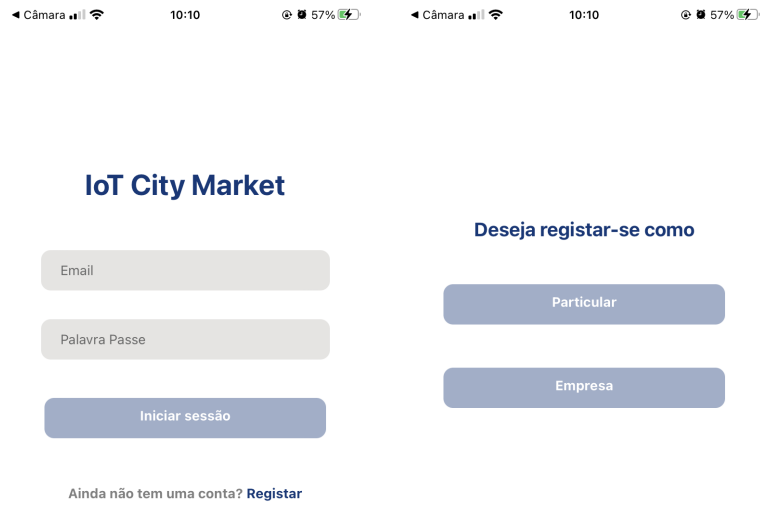
O componente *data marketplace* é o responsável por toda a interface do utilizador, ou seja, corresponde ao *front-end* do nosso projeto. O objetivo deste componente é atuar como uma plataforma na qual ocorre a comercialização dos dados provenientes dos sensores. Este é o local onde utilizadores publicam as suas ofertas para que os clientes interessados, possam explorar e caso tenham interesse, então adquirir. Uma vez que o intuito do nosso projeto recai sobre o desenvolvimento de uma aplicação para comercialização de produtos, neste caso dados IoT, e tal como qualquer outra aplicação *e-commerce*, é necessário o registo e *login* do utilizador para que este possa ter acesso à plataforma e realizar as tarefas que deseja. Estes utilizadores podem desempenhar o papel de vendedores, compradores ou ambos, e tanto podem ser indivíduos particulares como empresas.

Quanto à comercialização dos dados, um mesmo utilizador pode ser proprietário de um ou até vários sensores independentemente do tipo de sensor. No caso dos percursos de trotinetas elétricas estes pertencem a uma empresa. Tanto os dados dos sensores como dos percursos, quando colocados à venda na plataforma podem ser adquiridos por diversos compradores interessados. Um outro aspeto importante é o fato dos dados de um determinado sensor poderem ser comprados por períodos de tempo específicos, como por exemplo, por uma hora, um dia ou até uma semana e em que a cada período de tempo tem um preço associado. O tipo de dados que o sensor oferece também tem influencia no preço. Os dados relativos aos percursos são vendidos em conjuntos de percursos, não individualmente, no entanto, também por períodos de uma hora, um dia ou uma semana com preços definidos. Outra *feature* interessante, é a forma como o comprador tem acesso aos dados após os adquirir. Estes podem ser exportados em formato CSV.

Para criarmos a nossa aplicação móvel, optámos por utilizar a *framework* React Native. Esta é uma *framework* baseada na linguagem JavaScript. A escolha do React Native para o desenvolvimento do nosso projeto, advém do fato de precisarmos de desenvolver uma aplicação móvel capaz de funcionar tanto em Android como IOS. Como tal, a escolha do React Native foi imediata, uma vez que nos permite a possibilidade de *cross-platform*, através da integração com uma outra *framework* denominada Expo. Para uma melhor compreensão da implementação da plataforma, vamos dividir a explicação pelas diferentes páginas criadas.

4.3.4.1 *Login* e Registo

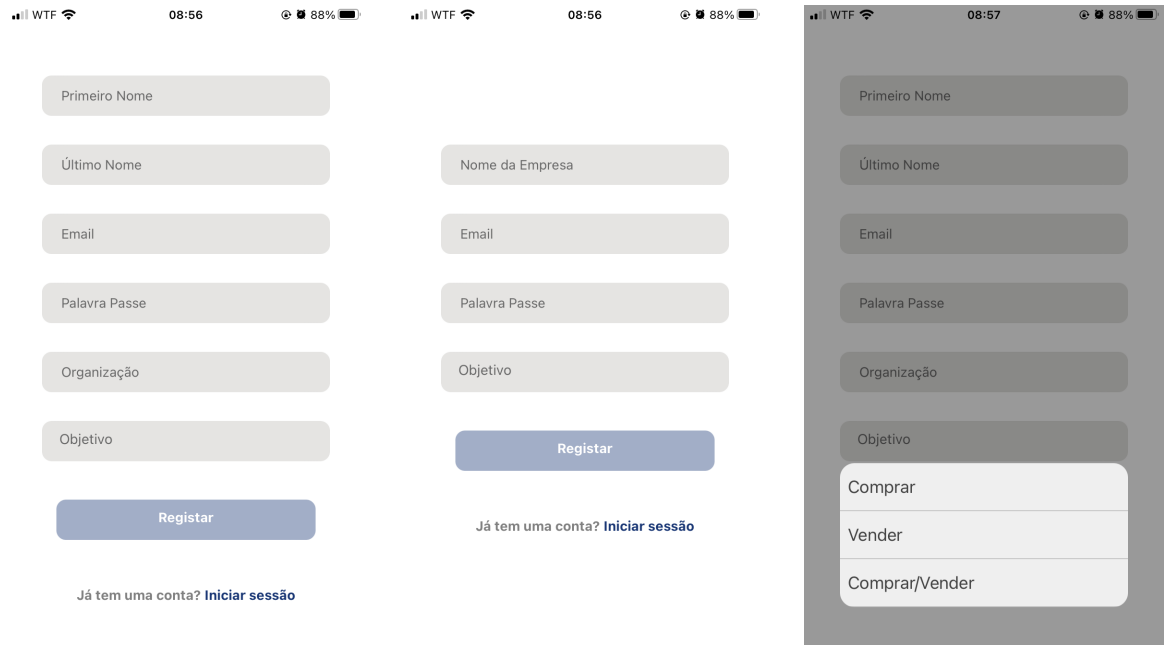
O primeiro passo tomado por nós no desenvolvimento deste componente, foi a criação de uma página de *login* na qual os utilizadores inserem as suas credenciais, e-mail e palavra-passe, para terem acesso à plataforma (4.19a). Para o caso dos utilizadores ainda não se encontrarem registados na aplicação, temos uma página para que o utilizador possa escolher se se deseja registar como particular ou empresa, como podemos ver na Fig 4.19b.



(a) Página de *login* (b) Opção de registo

Figura 4.19: Página de *Login* e Opção de registo.

Foram criadas duas páginas de registo. No caso da página de registo de um utilizador particular (4.20a), os campos exigidos são primeiro e último nome do utilizador, e-mail, palavra-passe, organização à qual pertence e também qual o seu objetivo na plataforma. Quanto à página de registo de empresas e outras entidades que desejem participar nesta plataforma (4.20b), é pedido o nome da empresa, e-mail, palavra-passe e também qual o seu objetivo. Neste caso, o nome da empresa desempenha a mesma função que o campo organização. Todas estas informações ficam armazenadas na base de dados do componente de controlo de acesso como mencionado em 4.3.1, para que as devidas permissões sejam concedidas aos utilizadores. O campo "Objetivo"corresponde ao que o utilizador pretende efetuar na plataforma, ou seja, comprar dados, vender ou ambas as hipóteses. Conforme o objetivo do utilizador, este tem acesso a diferentes páginas da aplicação como será explicado na secção 4.3.4.2. Já o campo Organização irá servir para que, no caso do proprietário de um sensor, este possa conceder diferentes níveis de acesso aos seus dados às organizações registadas na plataforma. Na Fig 4.20 podemos ver a página de registo para um utilizador particular, para uma empresa, organização ou outra entidade e também as opções do parâmetro Objetivo.

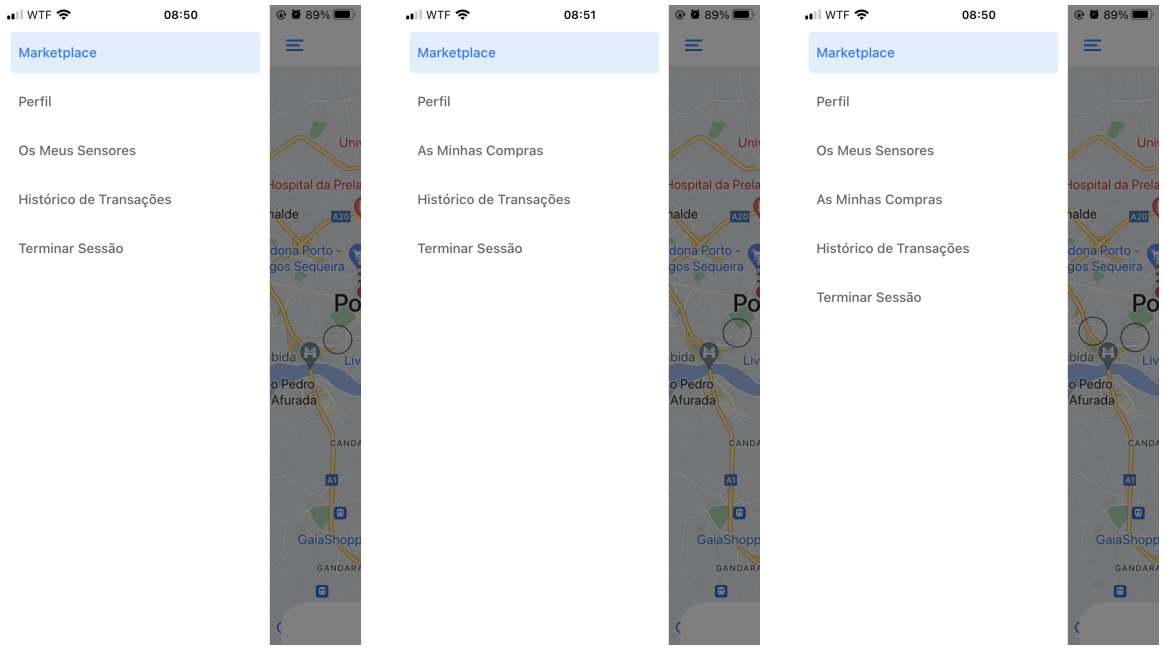


(a) Registo como particular (b) Registo como empresa (c) Parâmetro Objetivo

Figura 4.20: Páginas de registo e objetivo do utilizador no *data marketplace*

4.3.4.2 *Menu*

Após o registo ser realizado com sucesso, o *menu* do utilizador varia conforme o interesse deste no *data marketplace*. Se o utilizador estiver interessado em comprar dados, o *menu* apresenta uma opção "As Minhas Compras". Caso o interesse do utilizador seja apenas colocar os seus dados à venda, então apresenta apenas uma opção "Os Meus Dados", no caso de ambos, as duas opções estão visíveis no *menu*. O *menu* dispõe também uma opção "Perfil", "Marketplace", "Histórico de Transações"e por último, o botão "Terminar Sessão", caso o utilizador queira terminar a sua atividade na plataforma. Na Fig 4.21 podemos ver os diferentes tipos de *menus* conforme o papel do utilizador no *data marketplace*.



(a) *Menu* de um vendedor (b) *Menu* de um comprador (c) Menu de um utilizador com ambos os objetivos.

Figura 4.21: Diferentes tipos de *Menu*.

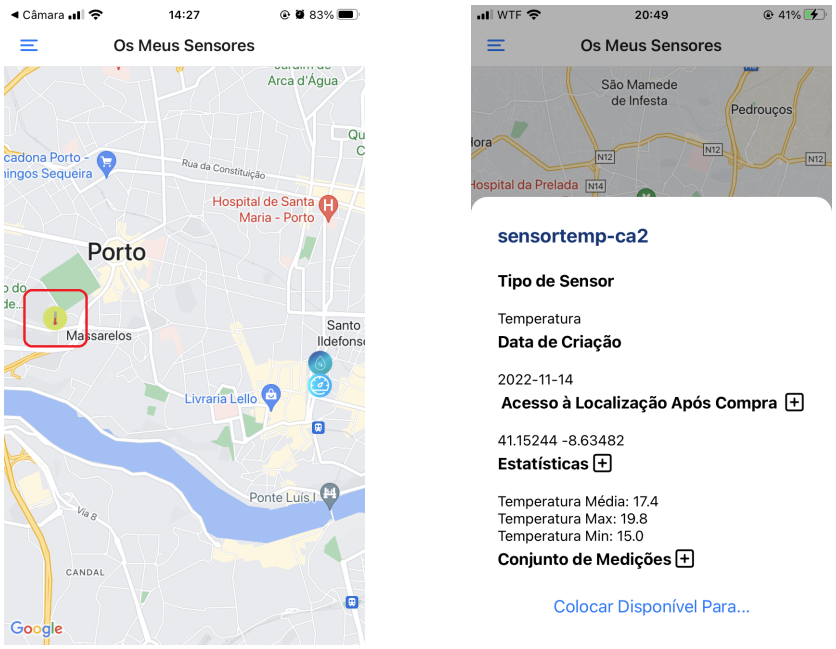
4.3.4.3 Perfil

A página de "Perfil", apresenta todas as informações pessoais relativas ao utilizador como o seu e-mail, organização, a sua função no *data marketplace* e um botão "Eliminar Conta", se este decidir apagar a sua conta na aplicação, como mostrado na Fig 4.22. Estas informações foram recuperadas da Cloud Firestore da Firebase onde estão armazenadas como explicado em 4.3.1.1. Caso o utilizador decida eliminar a conta, estas informações são automaticamente eliminadas do local onde estão armazenadas.

Figura 4.22: Exemplo da página Perfil de um utilizador

4.3.4.4 Os Meus Sensores

Para a implementação da página "Os Meus Sensores", recorremos ao pacote react-native-maps do React Native que nos permitiu integrar Google Maps na nossa página bem como para colocar os sensores no mapa. Neste mapa, estão dispostos os sensores do utilizador para que os possa colocar à venda se assim o desejar. Para a explicação do funcionamento desta página assim como dos respetivos sensores, iremos seguir o exemplo do sensor de temperatura sensortemp-ca2 visível na Fig 4.23a.

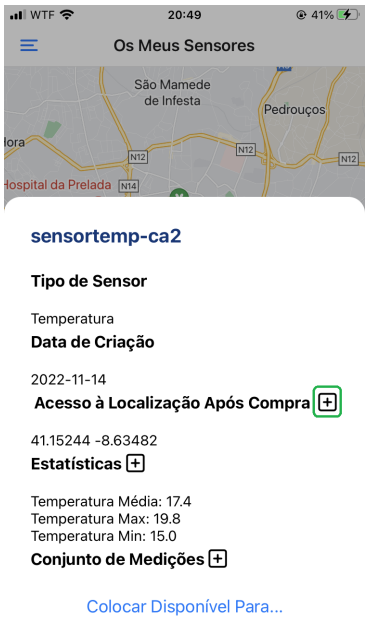


(a) Página Os Meus Sensores.

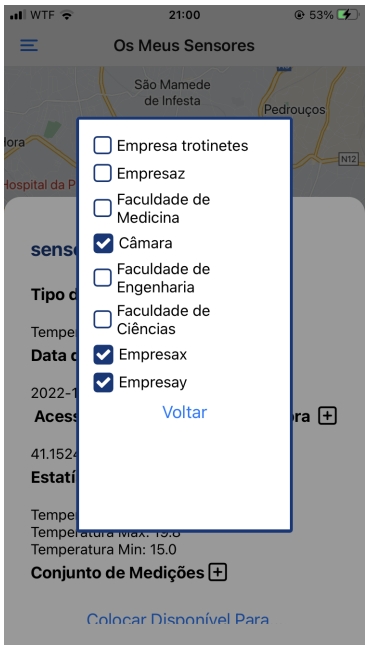
(b) Representação do sensor de temperatura

Figura 4.23: Página Os Meus Sensores e respetivo sensor de temperatura

É dada ao proprietário do sensor a possibilidade de permitir o acesso à localização exata do sensor após aquisição dos dados do mesmo (Fig 4.24). Este pode também escolher para quais organizações o sensor apresentará dados estatísticos (Fig 4.25) e para quais apresentará os dados sem processamento (Fig 4.26). Em cada uma das opções, após o utilizador clicar no botão correspondente, é-lhe apresentada uma lista com todas as organizações registadas na plataforma até ao momento, para que este possa escolher para quais deseja disponibilizar a localização exata do sensor após compra, dados estatísticos do sensor ou os conjuntos de medições (4.24b, 4.25b, 4.26b). É de salientar que cada organização só pode ter acesso a um tipo de dados.



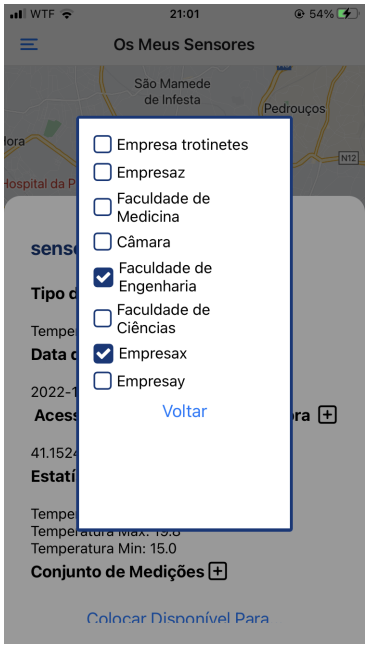
(a) Botão de permissão de acesso à localização após compra



(b) Organizações com acesso à localização

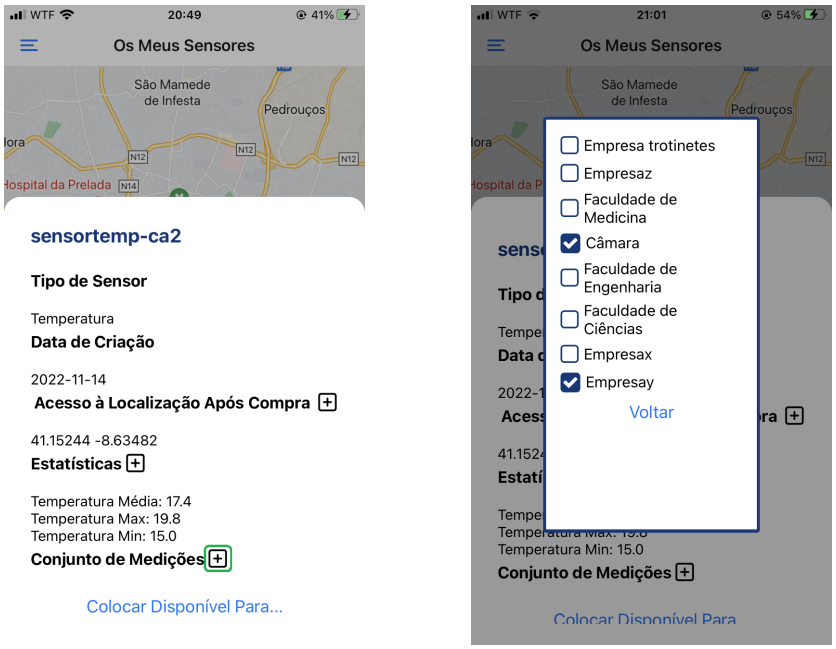
Figura 4.24: Permissão de acesso à localização após compra

(a) Botão de permissão de dados estatísticos



(b) Organizações com acesso a dados estatísticos

Figura 4.25: Permissão de acesso a dados estatísticos



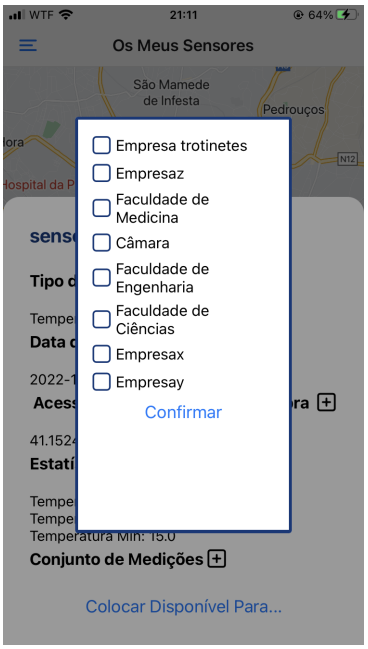
(a) Botão de permissão dos conjuntos de medições

(b) Organizações com acesso aos conjuntos de medições

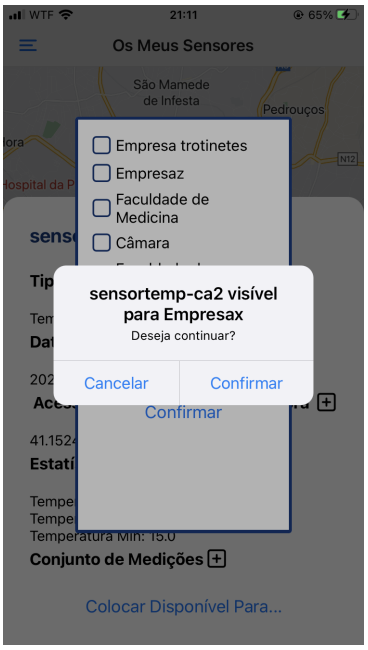
Figura 4.26: Permissão de acesso aos conjuntos de medições

Por fim, o utilizador seleciona para quais organizações o sensor ficará disponível para compra. Quando o utilizador seleciona uma organização é-lhe apresentado um aviso para confirmação da ação (Fig 4.27c). Podemos ver como é realizado o processo na Fig 4.27. Na Fig 4.27d temos o conjunto das organizações com acesso aos dados do sensor. O utilizador também pode cancelar o acesso aos dados do sensor a qualquer momento clicando na organização pretendida. Também lhe é apresentado um aviso para confirmação da ação, como mostrado na Fig 4.28.

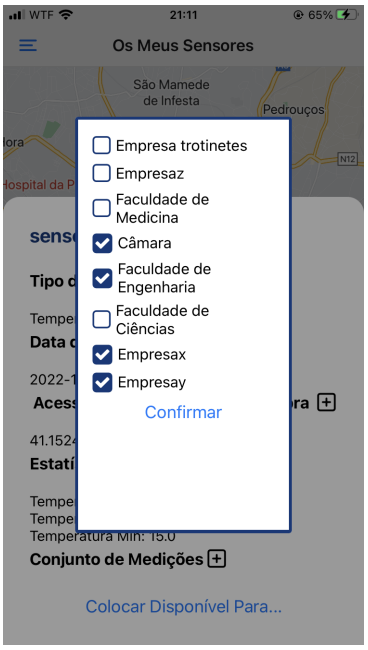
(a) Botão colocar os sensores disponíveis para compra



(b) Organizações registadas na plataforma

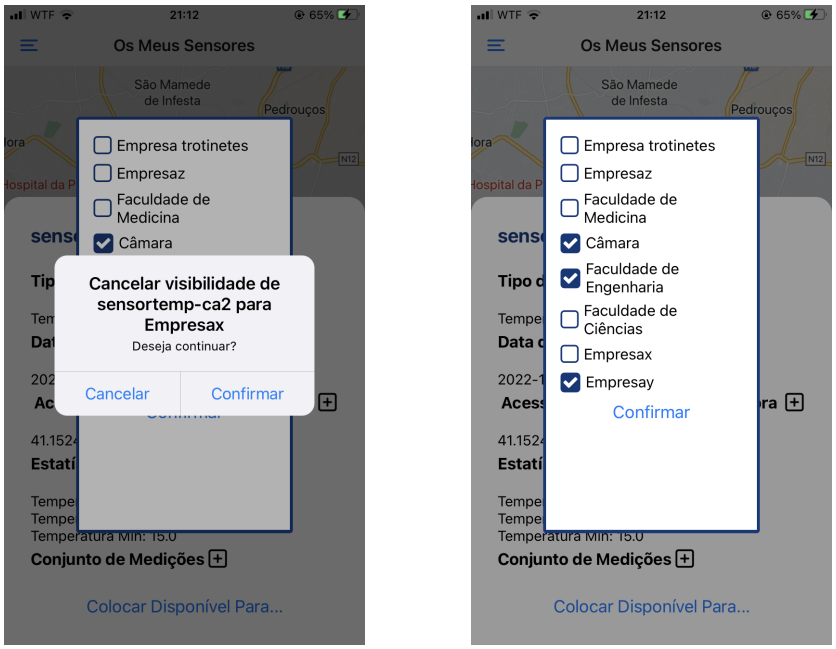


(c) Aviso de confirmação de colocar sensor disponível para a organização



(d) Organizações com acesso ao sensor

Figura 4.27: Processo para tornar o sensor disponível para compra

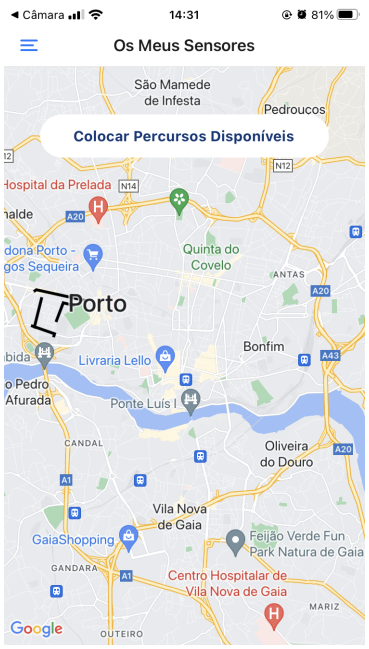


(a) Aviso de confirmação de retirar acesso ao sensor à organização

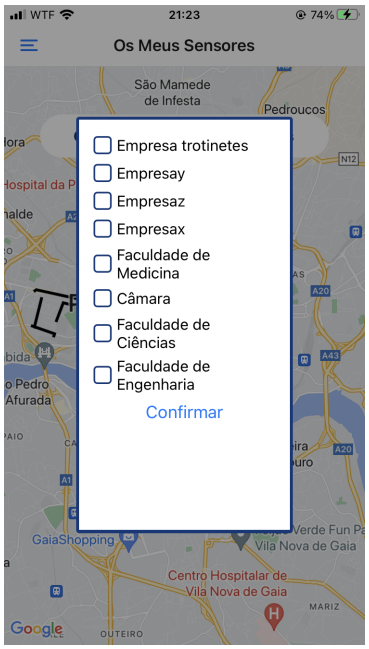
(b) Organizações com acesso ao sensor após cancelamento

Figura 4.28: Cancelamento do acesso ao sensor à organização

Para os percursos realizados por trotinetas elétricas, o processo de colocação para venda é ligeiramente diferente, pois os percursos não são vendidos individualmente, mas sim agrupados. Assim, o proprietário destes dados, dispõe de um botão "Colocar Percursos Disponíveis", na sua página "Os Meus Sensores"(Fig 4.29a), que lhe permite colocar os dados dos percursos à venda para as organizações desejadas (Fig 4.29d). Ao clicar no botão é-lhe apresentada, tal como para os sensores ambientais, uma lista com todas as organizações registadas na plataforma até ao momento, para que este possa selecionar as pretendidas (4.29b). Ao clicar numa organização é também apresentado um aviso para confirmação da intenção de colocar os percursos disponíveis para a organização (4.29c). Também aqui o proprietário tem o poder para cancelar o acesso aos dados a qualquer momento. Na Fig 4.29 podemos ver como se realiza o processo de colocar os percursos de trotinetas elétricas à venda.

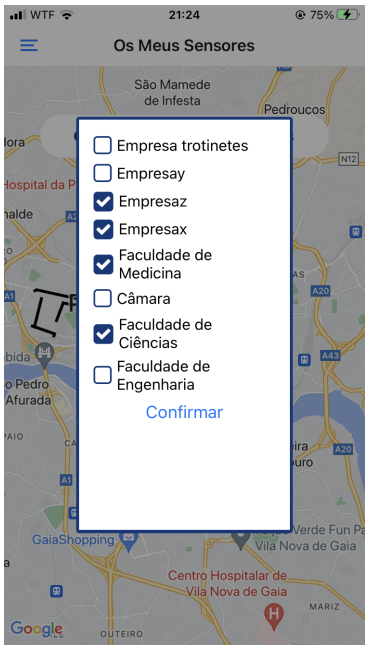


(a) Botão para colocar os percursos disponíveis para venda



(b) Organizações registadas na plataforma

(c) Aviso de confirmação de colocar os percursos disponíveis para a organização



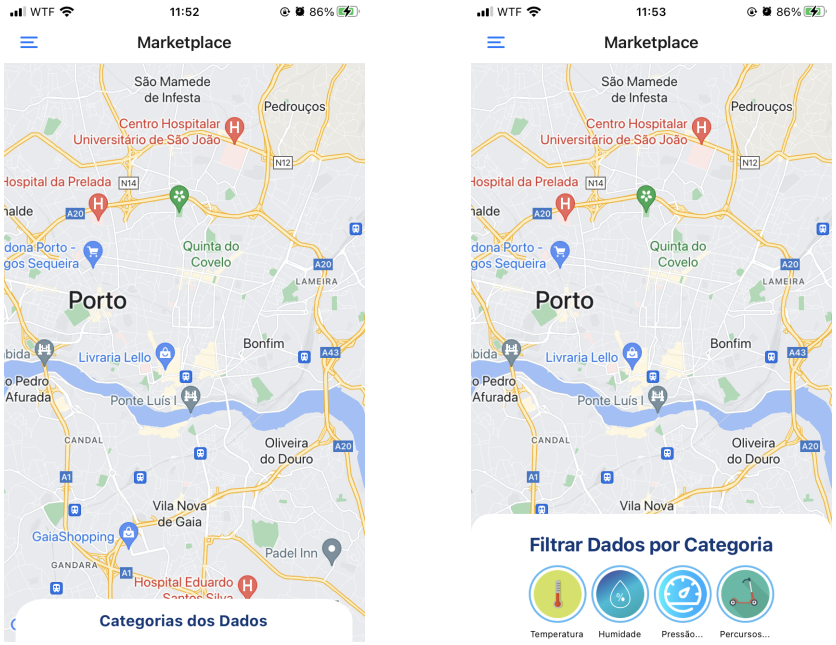
(d) Organizações com acesso aos dados dos percursos

Figura 4.29: Processo para colocar os percursos disponíveis para compra

4.3.4.5 *Marketplace*

Esta é a página de comercialização de dados, tal como a página "Os Meus Sensores", esta também é constituída por um mapa no qual estão apresentados todos os sensores que cada

utilizador colocou à venda. A título de exemplo, iremos continuar com sensores de temperatura, incluindo o sensor de temperatura sensortemp-ca2, para seguirmos um fluxo lógico. Nesta página, começámos por criar um botão "Categoria dos Dados", que apresenta um filtro para que o utilizador possa navegar sobre os diferentes tipos de dados de que a plataforma dispõe. Este filtro é constituído pelas opção "Temperatura", "Humidade Relativa", "Pressão Atmosférica"e "Percursos de Trotinetas Elétricas", como mostrado na Fig 4.30.

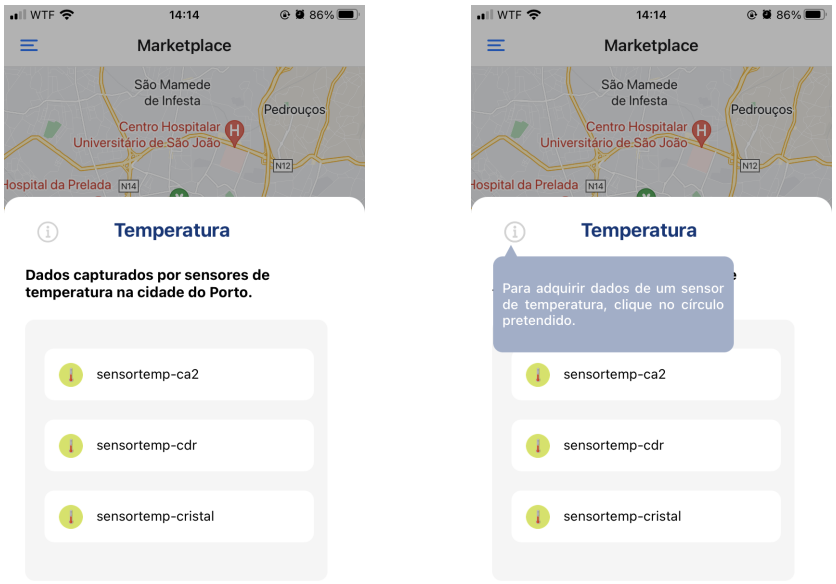


(a) Botão Categoria dos Dados

(b) Opção para filtrar dados por categoria

Figura 4.30: *Menu* para filtragem do tipo de dados oferecidos no *data marketplace*

Ao clicar em cada uma das categorias é possível, visualizar quais são os sensores que têm dados à venda até ao momento na plataforma. Na Fig 4.31a temos os sensores de temperatura disponíveis bem como um botão de ajuda que dá indicações ao utilizador de como adquirir dados dos sensores (Fig 4.31b).



(a) Lista dos sensores de temperatura disponíveis para compra

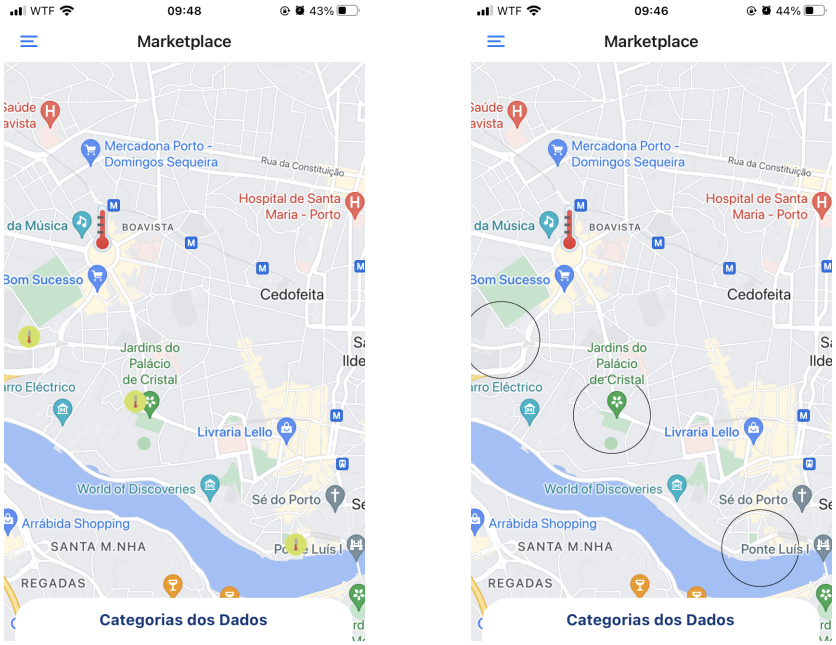
(b) Botão de ajuda com a respetiva informação

Figura 4.31: Sensores disponíveis e instruções para compra

Nesta plataforma, todos os sensores são comercializados sem qualquer indicação da sua localização. Para tal, utilizámos uma técnica que consiste na criação de um círculo com um determinado raio R e, em que o centro é gerado aleatoriamente com base nas coordenadas exatas do sensor. Assim, mostramos que o sensor se encontra dentro dessa região, na região do círculo, mas sem mostrar a sua localização exata. Simulámos um círculo de raio R, com centro nas coordenadas exatas do sensor e, de seguida, gerámos um ponto aleatório dentro desse círculo. Esse seria o centro do novo círculo, com o mesmo raio, que iria abranger a localização exata do sensor mas sem divulgar a mesma. A função utilizada para a criação do círculo é apresentada de seguida e na Fig 4.32 temos representado o antes e depois do sensores de temperatura disponíveis na plataforma após aplicação da técnica de ocultação da localização.

$$(x, y) = (x_0 + (\cos(r2\pi) \times (\sqrt{r}R)), y_0 + (\sin(r2\pi) \times (\sqrt{r}R))), \tag{4.1}$$

onde (x,y) representam as coordenadas do centro do círculo, r representa um número aleatório, R o raio do círculo e (x_0, y_0) representam as coordenadas do sensor.

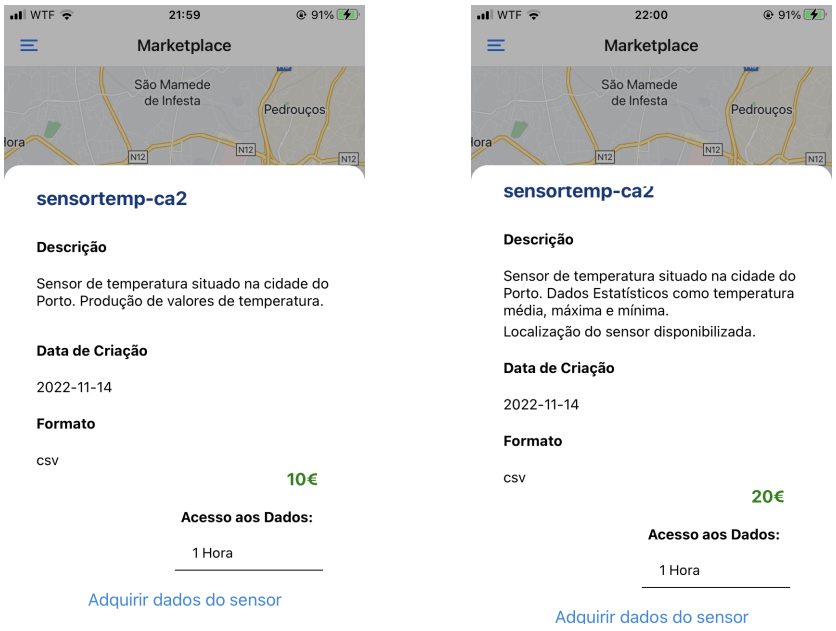


(a) Apresentação da localização de vários sensores de temperatura

(b) Localização dos sensores de temperatura oculta

Figura 4.32: Ocultação da localização aplicadas a vários sensores de temperatura

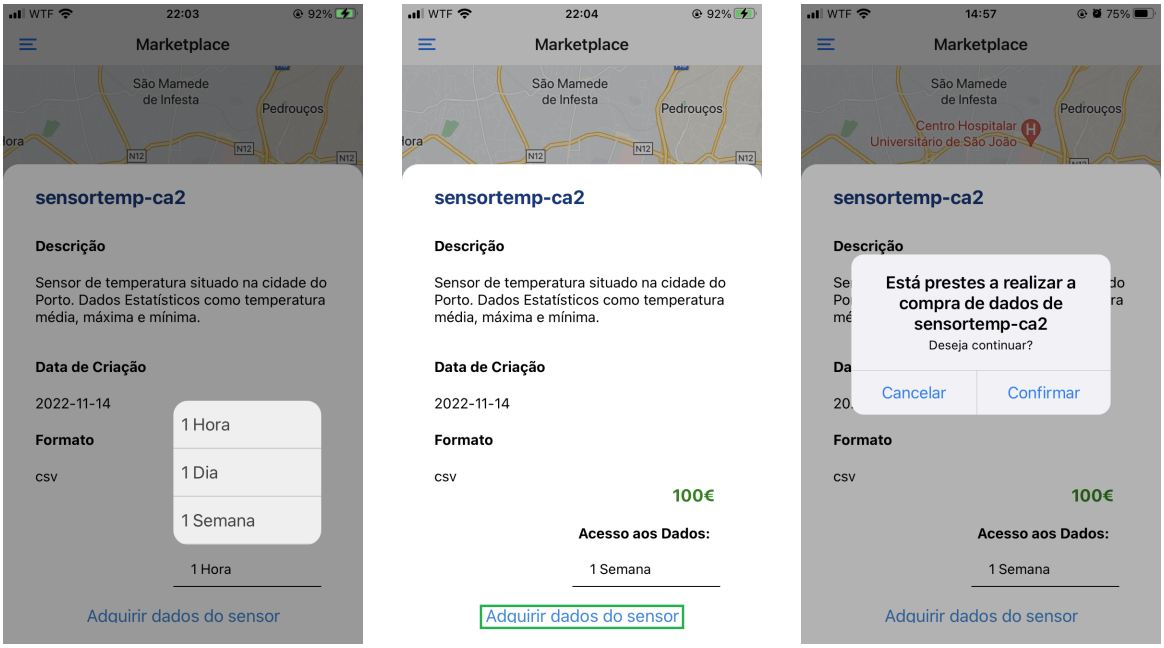
Para a explicação de como são apresentados os dados dos sensores nesta página, iremos voltar a utilizar apenas o sensortemp-ca2. Para todos os sensores é disponibilizada uma descrição dos dados que o sensor oferece, a data de quando começaram a ser produzidos, o formato do ficheiro de exportação, preço e o tempo pelo qual o utilizador deseja adquirir os dados. Se a organização à qual o utilizador autenticado pertence, tiver permissão para aceder ao dados não processados do sensor, a informação é representada como mostra a Fig 4.33a, no caso de ter permissão para aceder a dados estatísticos e também à localização exata do sensor, temos a informação representada na Fig 4.33b. Os dados do sensor podem ser adquiridos por períodos de uma hora, um dia ou uma semana (Fig 4.34a). Após o utilizador escolher o período de tempo pelo qual deseja adquirir os dados, este clica no botão "Adquirir dados do sensor" (4.34b), que lhe apresentará um aviso para confirmação de compra (4.34c). Na Fig 4.34 podemos ter uma perceção de como se realiza a compra dos dados do sensor por um período de uma semana.



(a) Representação da oferta dos dados não processados

(b) Representação da oferta de dados estatísticos com localização

Figura 4.33: Diferentes tipos de informação que um sensor pode oferecer



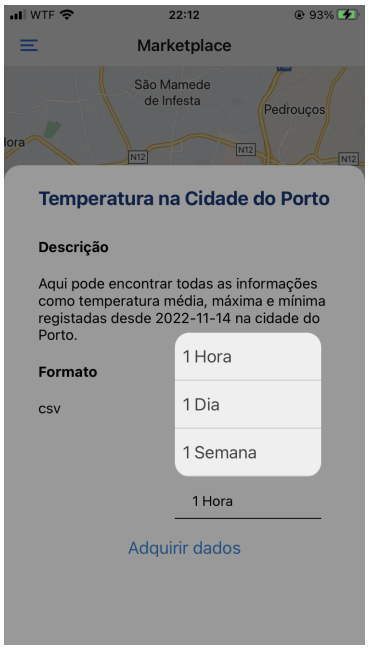
(a) Opções de acesso aos dados (b) Botão Adquirir dados do sensor (c) Confirmação para compra

Figura 4.34: Realização de compra do sensor sensortemp-ca2 por 1 semana

Também nas categorias de sensores de temperatura, humidade relativa e pressão atmosférica, temos um marcador representativo, através do qual é possível adquirir informações estatísticas da região do Porto, para cada tipo de dados. Na Fig 4.35a temos um marcador que permite adquirir

informações estatísticas de temperatura da região do Porto. A informação disponibilizada por este marcador possui uma descrição dos dados que oferece, bem como o formato do ficheiro de exportação dos mesmos (4.35c). Estes dados podem também ser adquiridos por períodos de uma hora, um dia e uma semana (4.35b). Após o utilizador escolher o tempo pelo qual pretende adquirir os dados e clicar no botão "Adquirir dados", é apresentado um aviso de confirmação de compra (4.35d). Na Fig 4.35, podemos ver como estes dados são apresentados bem como é efetuada a aquisição dos mesmos.

(a) Marcador representativo dos dados de temperatura da região do Porto



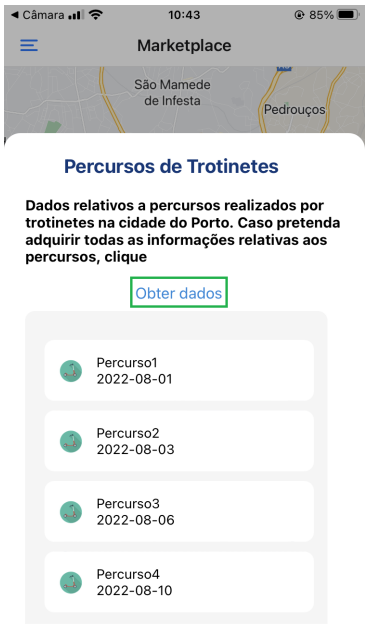
(b) Opções de acesso aos dados

(c) Botão Adquirir dados

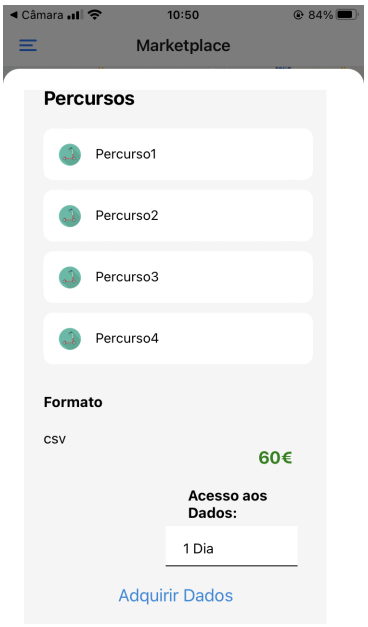
(d) Confirmação de compra

Figura 4.35: Processo de compra de informações de temperatura da região do Porto

Na categoria de percursos de trotinetas elétricas, temos os percursos realizados até ao momento e que podem ser adquiridos no *modal* da respetiva categoria através do botão "Obter dados"(Fig 4.36a). O restante processo é realizado igual aos dos sensores como é possível verificar na Fig 4.36b.



(a) Botão Obter dados



(b) *Modal* para compra de dados de percursos de trotinetas elétricas

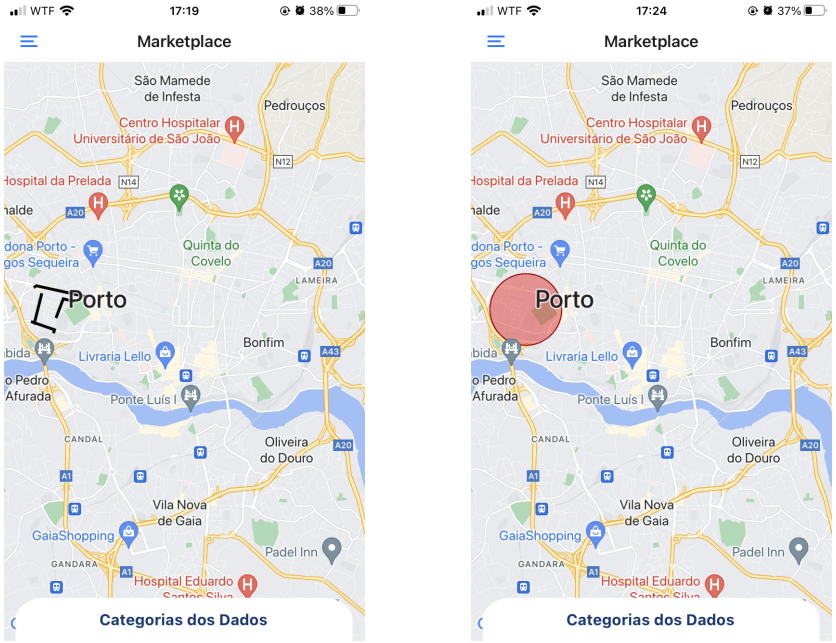
Figura 4.36: Compra dos dados dos percursos de trotinetas elétricas

Também estes dados são comercializados sem que cada percurso seja divulgado. Uma vez que estes dados pertencem a uma empresa desse domínio, apenas é vantajoso para ela que os dados relativos aos percursos sejam unicamente divulgados após a sua aquisição. Para tal, todos os percursos realizados numa determinada região, como por exemplo, na região da Boavista são representados por um círculo que abrange todos estes percursos. Com este procedimento, é possível saber que os percursos se encontram na zona do círculo, mas não explicitamente os percursos em si. A função para a criação do círculo é apresentada de seguida:

$$(x, y) = (x_0 + (\cos(r2\pi) \times (\sqrt{r}R)), y_0 + (\sin(r2\pi) \times (\sqrt{r}R))), \tag{4.2}$$

$$(x_0, y_0) = (\frac{x_{max} + x_{min}}{2}, \frac{y_{max} + y_{min}}{2}) \tag{4.3}$$

onde (x,y) representam as coordenadas do centro do círculo, r representa um número aleatório, R o raio do círculo (a distância do ponto mais longínquo a (x_0, y_0)) e (x_0, y_0) representam respetivamente o ponto médio entre o ponto mais a oeste e o ponto mais a este, e o ponto médio entre o ponto mais a norte e o ponto mais a sul. Na Fig 4.37 é possível visualizar o resultado do procedimento descrito.



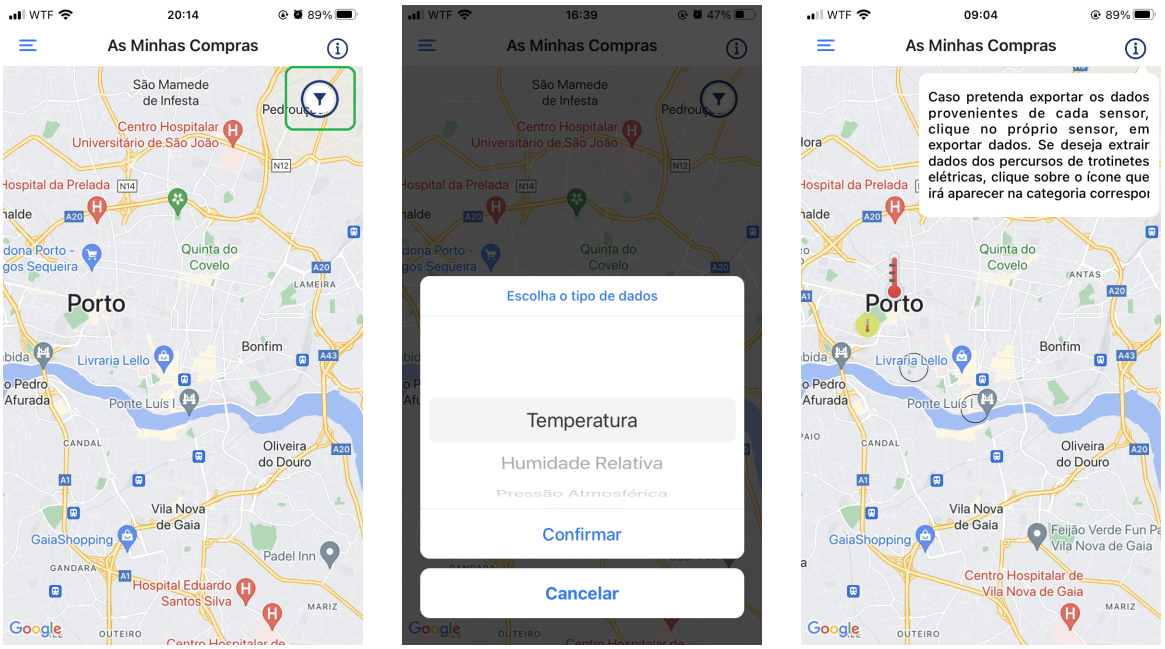
(a) Percursos sem localização exata oculta

(b) Percursos com localização exata oculta

Figura 4.37: Ocultação da localização exata dos percursos de trotinetas elétricas

4.3.4.6 As Minhas Compras

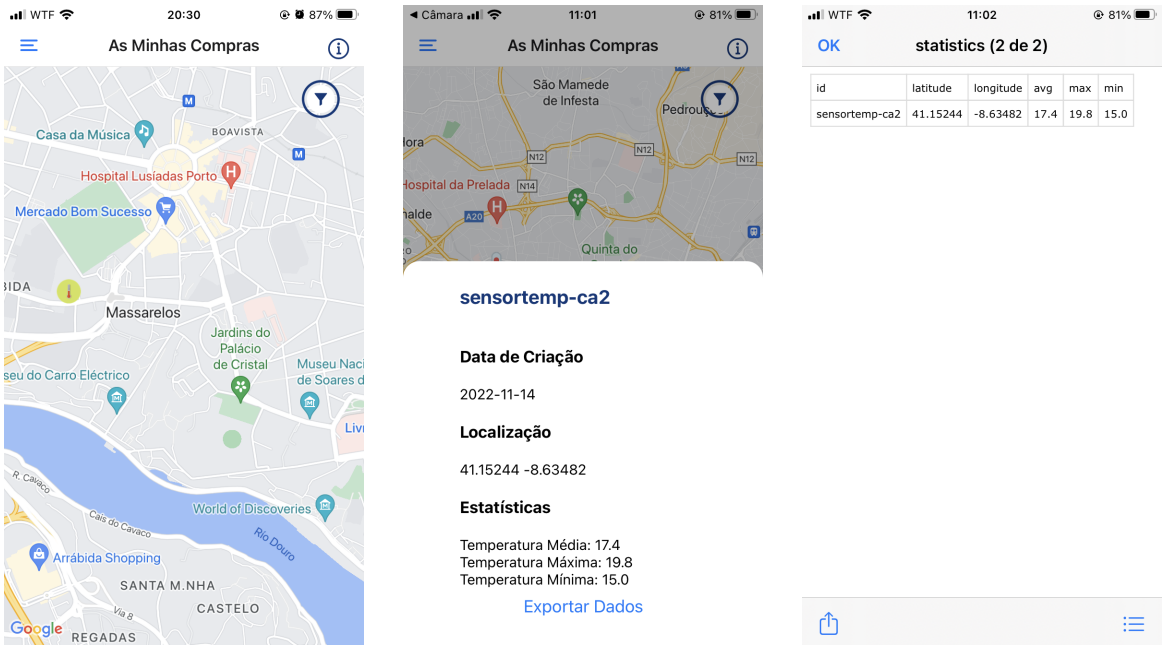
Na página "As Minhas Compras", temos um mapa onde ficam dispostos os sensores dos quais o utilizador adquiriu dados. Também para a descrição desta página vamos recorrer ao sensortemp-ca2. Por questões de organização, também nesta página optámos por criar um filtro (Fig 4.38b), de forma a que o utilizador possa navegar sobre os diferentes tipos dados que adquiriu, se for o caso. Temos também um botão de ajuda (Fig 4.38c) que indica como o utilizador deve proceder para exportar os dados dos sensores e dos percursos.



(a) Botão de filtragem (b) Filtro dos dados (c) Botão de ajuda e respetiva informação

Figura 4.38: Página As Minhas Compras com filtro de dados e informações de exportação

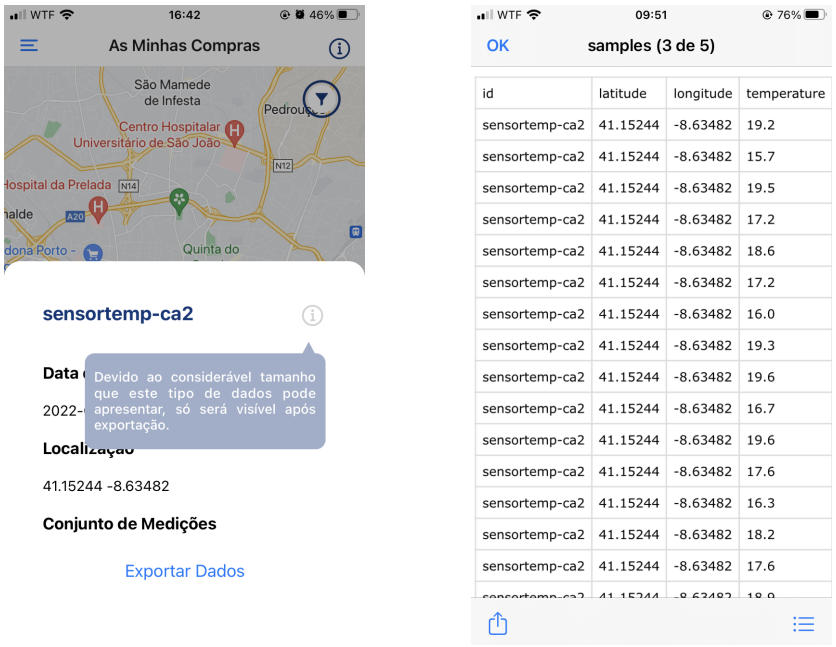
Na Fig 4.39a, temos o sensortemp-ca2 do qual o utilizador adquiriu dados. Podemos concluir que este utilizador faz parte de uma organização à qual o proprietário deu permissão de acesso à localização exata do sensor após compra, como explicado em 4.3.4.4. Na Fig 4.39b podemos ver como são representados os dados estatísticos com acesso à localização adquiridos. É também apresentado um botão "Exportar Dados"que permite ao utilizador exportar os dados durante o período pelo qual os adquiriu. Ao clicar no botão, os dados serão exportados num ficheiro em formato CSV com a identificação do sensor, coordenadas de latitude e longitude, temperatura média, máxima e mínima. (4.39c).



(a) Sensor com localização exata (b) Dados do sensor com estatísticas e localização exata (c) Ficheiro csv com dados estatísticos e localização exata

Figura 4.39: Dados do sensor adquiridos com informações estatísticas e localização exata

No caso da aquisição de conjuntos de medições, ou seja, dados sem processamento é apresentado um botão de ajuda (Fig 4.40a). Este, indica ao utilizador que este tipo de dados só poderá ser visível após exportação dos mesmos devido ao tamanho que estes podem apresentar. O ficheiro CSV exportado contem a identificação do sensor, as coordenadas de latitude e longitude e as medições de temperatura geradas (4.40b).

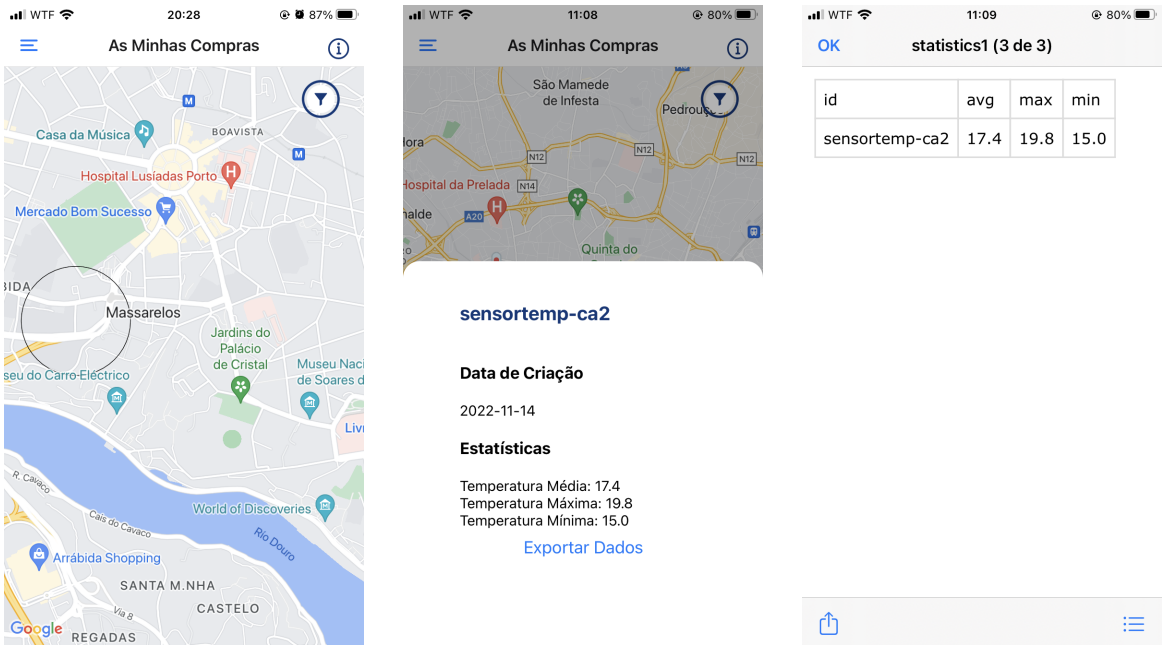


(a) Sensor com localização exata e conjuntos de medições

(b) Ficheiro csv com conjuntos de medições e localização

Figura 4.40: Dados adquiridos com acesso à localização exata e conjuntos de medições

Na Fig 4.41a é possível verificar que o utilizador não tem permissão para aceder à localização exata do sensor, dessa forma, podemos ver na Fig 4.41b que as coordenadas do sensor não são apresentadas. Também na mesma figura temos o caso de dados estatísticos adquiridos, na qual é apresentado a data de criação dos dados e a temperatura média, máxima e mínima. A exportação dos dados é realizada através do botão "Exportar Dados". O ficheiro CSV exportado é composto pela identificação do sensor, valor médio, máximo e mínimo (4.41c). A representação para conjuntos de medições seria igual ao representado na Fig 4.40a à exceção de visualização das coordenadas. O ficheiro CSV seria igual ao apresentado na Fig 4.40b, também sem as coordenadas.



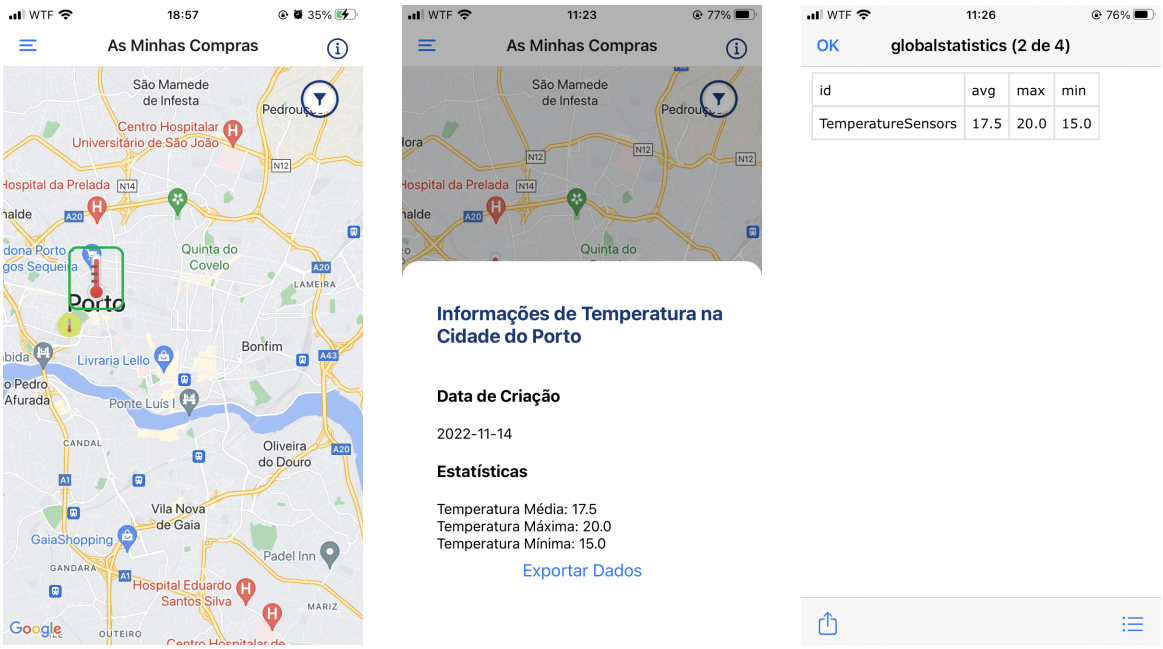
(a) Sensor sem acesso à localização exata após compra

(b) Dados estatísticos adquiridos sem acesso à localização do sensor

(c) Ficheiro csv apenas com estatísticas

Figura 4.41: Sensor adquirido com informações estatísticas

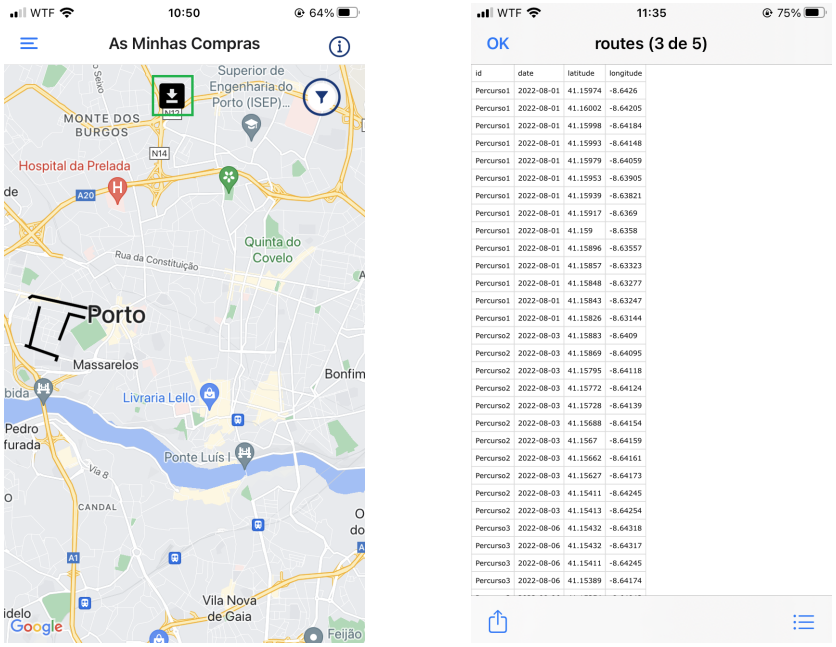
No caso das informações de temperatura da região do Porto, o marcador representativo é disposto sobre o mapa do utilizador, tal como ocorre para os outros sensores (Fig 4.42a). A informação disponibilizada por este marcador dispõe de data de criação dos dados e temperatura máxima, média e mínima. A informação é representada como mostrado na Fig 4.42b. O ficheiro CSV segue a estrutura que pode ser visualizada na Fig 4.42c.



(a) Marcador representativo de temperatura na região do Porto (b) Representação dos dados adquiridos (c) Ficheiro csv com dados adquiridos

Figura 4.42: Informações adquiridas de temperatura da região do Porto

Para o caso dos percursos realizados por trotinetas elétricas, podemos observar na Fig 4.43a que o utilizador que adquiriu os dados tem agora acesso aos percursos exatos. Ao clicar no botão apresentado no centro do mapa, o utilizador exporta todos os dados relativos aos percursos, também em formato CSV. Este ficheiro é constituído pela identificação de cada percurso, a data em que cada um deles foi realizado e todos os conjuntos de coordenadas que os constituem 4.43b.



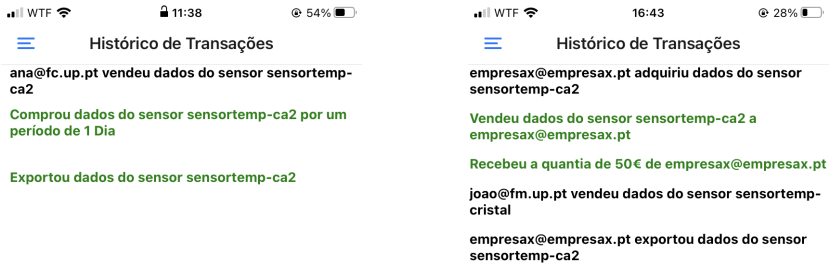
(a) Percursos de trotinetas elétricas adquiridos

(b) Ficheiro csv com as informações dos percursos

Figura 4.43: Representação dos percursos e ficheiro csv exportado

4.3.4.7 Histórico de Transações

Na página "Histórico de Transações", temos o registo da atividade dos utilizadores na plataforma. Vamos utilizador como exemplo o sensortemp-ca2 para demonstrar o funcionamento desta página. Quando os dados do sensor são adquiridos por um comprador, a atividade fica registada na página "Histórico de Transações"do proprietário do sensor, com adição da quantia transferida para este. A atividade de compra fica também registada na página do próprio comprador. A atividade de venda dos dados do sensor por parte do vendedor, é registada na própria página do vendedor e de todos os utilizadores com permissão de acesso ao sensor. Quando os dados são exportados pelo comprador, a ação também fica registada tanto na página do comprador como do vendedor. Na Fig 4.44a temos a página "Histórico de Transações"de um utilizador que adquiriu dados do sensor sensortemp-ca2. Primeiramente, é mostrado a informação de que o proprietário do sensor vendeu dados do sensor. Em segundo lugar, é mostrado o registo da compra realizada pelo comprador. Quando este exporta os dados, a informação é também apresentada. No caso da Fig 4.44b temos a página do utilizador que vendeu os dados do sensor. É apresentada a informação de aquisição dos dados do sensor por parte do comprador, seguido das informações que indicam ação de venda por parte do utilizador e também a informação da quantia recebida pelos dados. A informação de exportação dos dados pelo comprador também é visível na figura. Em adição, nesta figura, temos também o exemplo de dados vendidos por outro utilizador da plataforma, que permitiu o acesso ao sensor à organização à qual o autenticado pertence.



(a) Registo de atividade de um comprador

(b) Registo de atividade de um vendedor

Figura 4.44: Histórico de Transações de utilizadores

4.3.5 Fluxo de Venda

Nesta subsecção iremos abordar as diferentes fases que constituem o fluxo de venda de dados de sensores no *data marketplace* como o acesso dos vendedores aos seus sensores (4.3.5.1). Iremos também abordar como os vendedores definem as informações que o sensor irá apresentar (4.3.5.2), as permissões de localização após compra dos dados (4.3.5.3) e também como os sensores são colocados disponíveis para compra (4.3.5.4). Para o esclarecimento dos três últimos processos mencionados iremos ter por base o diagrama 4.46. Também nesta subsecção apresentamos como os sensores são colocados na página de comercialização, ou página *Marketplace* (4.3.5.5).

4.3.5.1 Sensores na página Os Meus Sensores

Para uma melhor compreensão de como ocorre o fluxo de venda no nosso *data marketplace*, primeiramente, é preciso perceber como é que cada vendedor tem acesso aos respetivos sensores na página "Os Meus Sensores". Como referimos anteriormente, dividimos o nosso projeto nas etapas de simulação de dados de sensores IoT reais e desenvolvimento do *data marketplace*. É aqui que entra a primeira etapa, pois o simulador de dados tem como foco a simulação de sensores reais. Assim sendo, e uma vez que se trata de uma simulação, para que um vendedor adicione um novo sensor ao *data marketplace*, este processo tem de ser realizado no simulador. No caso do tipo do sensor ser de algum tipo já existente no simulador, o sensor é adicionado na sessão correspondente. Caso a sessão não exista, é necessário proceder à criação da mesma e

do respetivo tópico no *message broker*. Prosseguindo agora para o acesso do vendedor aos seus sensores, todos sensores simulados têm associado um proprietário, ou seja, os vendedores no *data marketplace*. Seguindo um pensamento lógico, os dados de cada sensor são enviados a partir da sessão para o respetivo tópico do *message broker* (1). Estes são então reencaminhados do *message broker* para o consumidor (2), que os processa e armazena na base de dados em coleções, em que cada coleção contem os dados dos sensores de um tipo (3).

Em primeiro lugar, para que o vendedor tenha acesso aos seus dados na respetiva página, este tem de estar autenticado no *data marketplace* (4). Posto isto, são realizados pedidos à API para obtenção dos dados dos sensores presentes em cada uma das coleções da base de dados (5) que constituem a base de dados. Como anteriormente mencionado em 4.3.1, utilizamos JWT *tokens* gerados no momento de autenticação do utilizador. Estes são enviados em cada pedido à API, e são verificados a partir do componente controlo de acesso (6). Caso o JWT coincida com o do utilizador autenticado é enviado uma resposta de confirmação (7) e são feitos os pedidos à base de dados (8). A base de dados envia uma resposta com os dados dos sensores (9) que por sua vez, são enviados para a página Os "Meus Sensores"(10). Para que consigamos relacionar o utilizador autenticado com os respetivos sensores, precisamos de recuperar as informações armazenadas na base de dados do componente de controlo de acesso (11)(12), nomeadamente o e-mail. Dessa forma, ao compararmos o e-mail do utilizador autenticado com o parâmetro owner presente em cada sensor obtemos todos os sensores do proprietário, como podemos observar na Fig 4.23a.

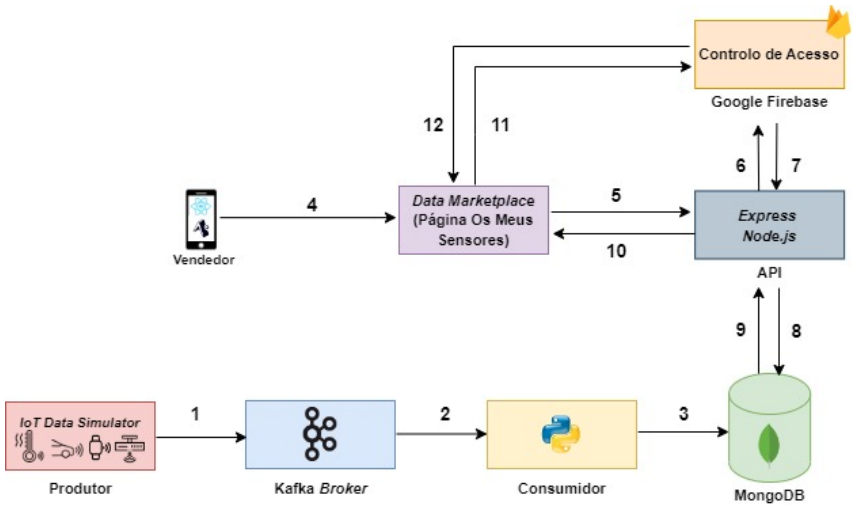


Figura 4.45: Fluxo para apresentação dos sensores na Página Os Meus Sensores

4.3.5.2 Dados que o sensor oferece consoante organização

O vendedor autentica-se na plataforma e acede à página "Os Meus Sensores"onde seleciona o sensor, do qual tem interesse em colocar os seus dados à venda (1). Para definir o tipo de dados que o sensor irá apresentar às organizações desejadas, dados estatísticos ou simplesmente as medições geradas em tempo real, é apresentada uma lista com todas as organizações com atividade na plataforma até ao momento. Por padrão, o sensor está configurado para apresentar

dados estatísticos, assim o valor das organizações na base de dados é *false*. Ao selecionar uma organização ou várias para que tenham acesso aos conjuntos de medições, é feito um pedido à API para atualização do campo AllowInfo na base de dados (2). A verificação do JWT *token* enviado no pedido, é realizada (3)(4). Na base de dados, o valor das organizações desejadas é atualizado para *true* (5). A resposta com a atualização realizada na base de dados é enviada (6) e as organizações aparecerem selecionadas para terem acesso aos conjuntos de medições (7), como pode ser observado na Fig 4.26b.

4.3.5.3 Acesso à localização do sensor após compra

Para que o vendedor selecione as organizações que terão acesso à localização exata do sensor após aquisição dos dados do mesmo pelo comprador, o processo é idêntico ao anteriormente descrito. É apresentada uma lista com todas as organizações com atividade no *data marketplace*. O utilizador seleciona aquelas que quer que tenham acesso à localização do sensor. É feito um pedido à API para atualização do campo AllowLocal na base de dados (2). O JWT *token* enviado no pedido é verificado (3)(4). O pedido segue então para a base de dados e o valor das organizações desejadas é atualizado para *true* (5). A base de dados envia a resposta com a atualização realizada (6). As organizações com acesso à localização após compra dos dados do sensor ficam selecionadas (7). O resultado deste processo é apresentado na Fig 4.24.

4.3.5.4 Organizações com permissão para aceder aos dados do sensor

Esta é a fase na qual o vendedor coloca os dados do sensor disponíveis para compra. Também aqui lhe é apresentada uma lista com as organizações disponíveis no *data marketplace*. Este seleciona as organizações para as quais pretende que o sensor fique disponível. É feito um pedido à API para atualização do campo organizations (2). O processo de verificação do JWT *token* é realizado como anteriormente descrito. A API continua com o pedido e o valor das organizações do campo organizations é atualizado para *true* (5). A base de dados envia a resposta com a atualização realizada (6) e as organizações com permissão para aceder aos dados do sensor ficam selecionadas (7). Este processo está representado na Fig 4.27. O sensor fica assim disponível na página "Marketplace"para as organizações desejadas.

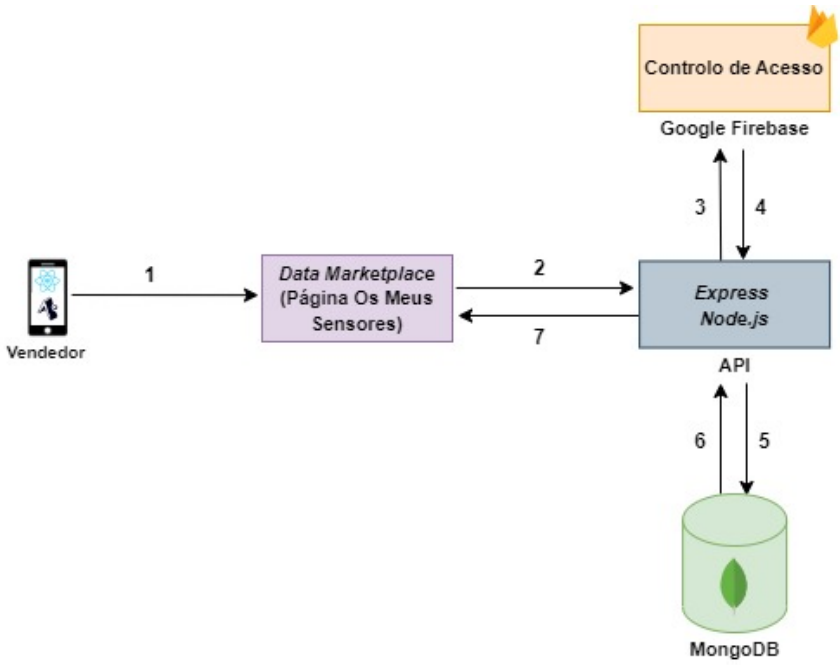


Figura 4.46: Fluxo de dados para explicação das secções 4.3.5.2, 4.3.5.3 e 4.3.5.4

4.3.5.5 Sensores na página "Marketplace"

Na página "Marketplace", aparecem os sensores colocados à venda apenas para o proprietário dos sensores e para todos os utilizadores das organizações com permissão para aceder aos dados dos sensores. Tanto o vendedor como o comprador se podem autenticar na plataforma e aceder à página "Marketplace"(1). São realizados pedidos à API para obtenção dos dados dos sensores presentes em cada uma das coleções da base de dados (2). Tal como nos processos anteriormente descritos o JWT *token* enviado nos pedidos é verificado através do componente controlo de acesso (3)(4). Os pedidos prosseguem para a base de dados (5) que envia uma resposta com os dados dos sensores das coleções (6) e estes, são enviados para a página *Marketplace* (7). Neste ponto, a atualização do campo organizations de cada sensor, consequente do processo realizado em 4.3.5.4 é visível. O valor das organizações com permissão de acesso aos dados do sensor é *true*. Para que os sensores sejam visíveis apenas para o proprietário e para os utilizadores das organizações permitidas para aceder aos dados do sensor, precisámos de recuperar as informações armazenadas na base de dados do componente de controlo de acesso (8)(9). Estas informações correspondem à organização dos utilizadores autenticados e o e-mail. Assim, se a organização do campo organizations é igual à organização do utilizador autenticado e o seu valor é *true*, os sensores estão visíveis. Para o caso do vendedor, se o e-mail do utilizador autenticado coincidir com o campo owner do sensor, este também é visível.

A representação da informação dos sensores na página *Marketplace* está diretamente relacionada com o campo AllowInfo. A descrição do sensor varia consoante o valor *true* ou false presente nesse campo. Se a organização no campo AllowInfo for igual à organização do utilizador autenticado e o seu valor for *true* podemos ver o resultado da descrição do sensor na Fig 4.33a.

No caso do valor da organização ser igual a *false*, a descrição do sensor é referente a informações estatísticas como podemos observar na Fig 4.33b.

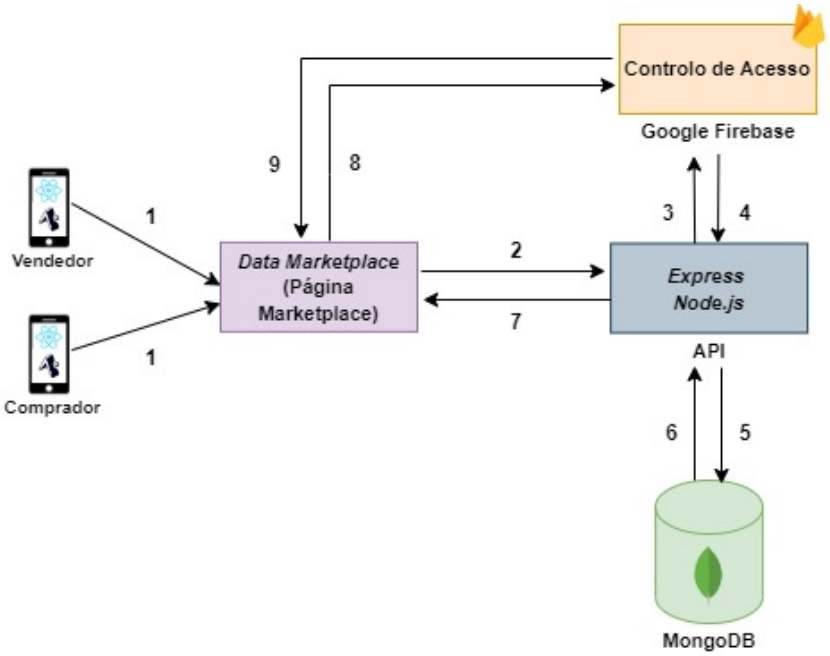


Figura 4.47: Fluxo para apresentação dos sensores na Página *Marketplace*

4.3.6 Fluxo de Compra

Na subseção de fluxo de compra iremos descrever a realização da compra de sensores, bem como é feito o registo de atividade dos utilizadores no *data marketplace* (4.3.6.1). Iremos também abordar o acesso aos dados do sensor após compra (4.3.6.2) e a perda do acesso aos dados após o prazo estipulado (4.3.6.3) através do diagrama 4.50. Por fim, as informações que o sensor disponibiliza (4.3.6.4).

4.3.6.1 Compra do sensor e registo de atividade

Seguindo o pensamento através do diagrama 4.48, o comprador autentica-se na plataforma, acede à página "Marketplace"e seleciona um sensor presente no mapa (1). Se este apresentar informações do seu interesse, escolhe o tempo pelo qual requer os dados. Assim, é feito um pedido à API para atualização do campo buyUser do respetivo sensor (4). No corpo do pedido é enviada a identificação do comprador, neste caso o e-mail do utilizador autenticado, recuperado da base de dados do componente de controlo de acesso (2)(3), juntamente com o tempo de acesso aos dados do sensor, escolhido pelo comprador, em *timestamps*. É possível visualizar um exemplo do campo buyUser após a sua atualização na Fig 4.14 presente na subsecção **??**. O JWT *token* é verificado (5)(6) e o campo buyUser é atualizado (7) e o sensor irá aparecer na página "As Minhas Compras".

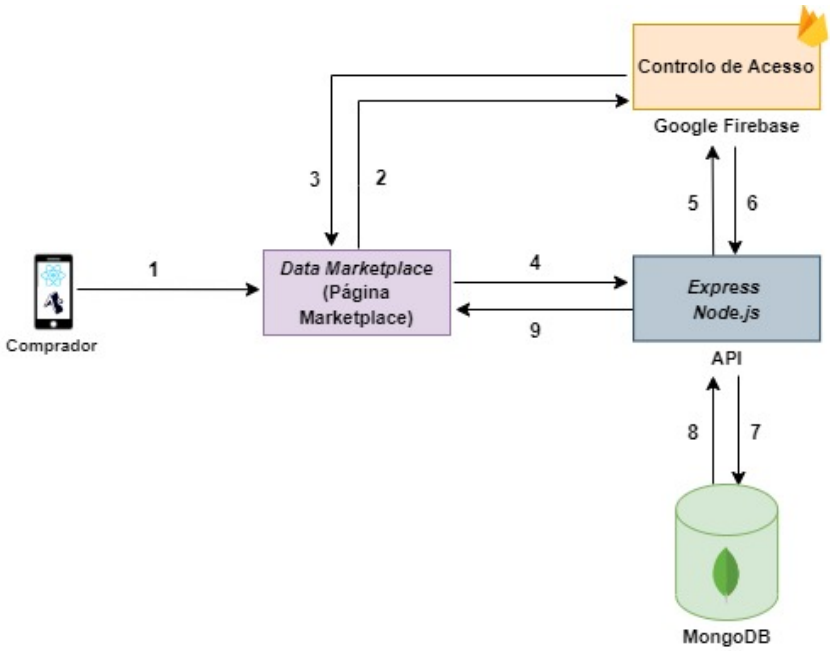


Figura 4.48: Fluxo de compra de um sensor

Em paralelo, acontece o registo de atividade dos utilizadores que pode ser compreendido através do diagrama 4.49. As ações de aquisição dos dados de um sensor por parte do comprador bem como a consequente venda do mesmo, ficam registadas na coleção correspondente na base de dados. Quando os dados de um sensor são comprados, é feito um pedido à API para inserção da atividade efetuada (4). O corpo do pedido é constituído pela identificação do sensor, identificação do comprador, o e-mail recuperado da base de dados do componente de controlo de de acesso (2)(3) e as *strings* que irão ficar registas na página do comprador, vendedor e restantes utilizadores da plataforma. O JWT *token* é verificado (5)(6), o pedido prossegue para a base de dados e os registos de atividade são inseridas na coleção correspondente (7). A resposta é então envida pela base de dados (8) e estes são apresentadas na "Página Histórico de Transações"(9). A estrutura das transações pode ser visualizada na Fig 4.13 da secção 4.2.1.

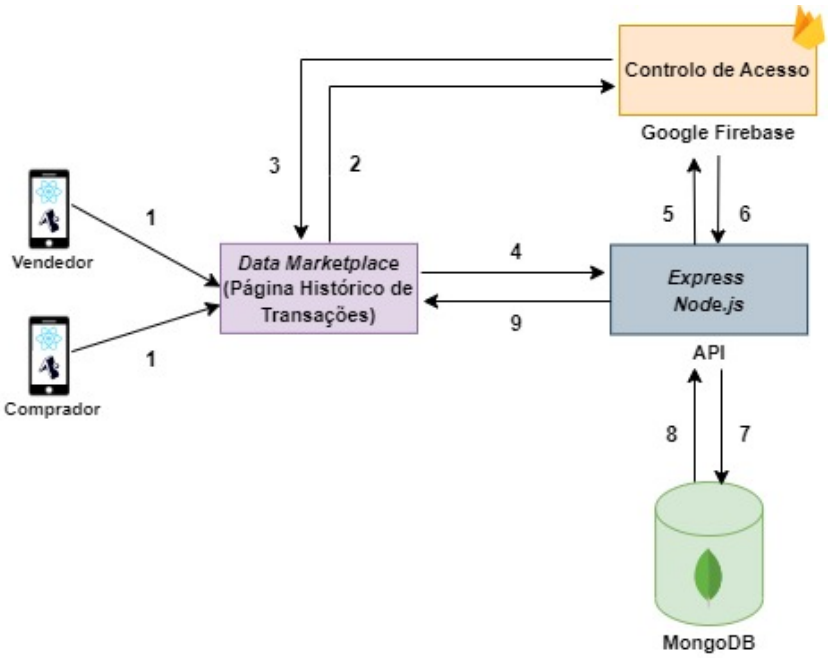


Figura 4.49: Fluxo do registo de atividade dos utilizadores

4.3.6.2 Sensores na página "As Minhas Compras"

Após a compra dos dados do sensor e para que fique registada a sua compra, este é fixado no mapa da página "As Minhas Compras"do comprador. Neste caso, também é realizado um pedido à API para obtenção dos dados dos sensores de todas as coleções da base de dados (2), os JWT *tokens* são verificados (3)(4) e os dados dos sensores das coleções são recuperados com todas as atualizações realizadas até ao momento (5)(6)(7). Há dois aspetos importantes a ter em consideração. Primeiro, a identificação do utilizador no campo buyUser do sensor, ou seja o seu e-mail, tem de coincidir com o e-mail do utilizador autenticado (8)(9). Segundo é preciso ter em atenção se na fase do vendedor colocar o sensor à venda, se este deu permissão de acesso à localização após compra dos dados do sensor à organização à qual o utilizador pertence, como descrito em 4.3.5.3. Assim, também o campo AllowLocal é importante. Se a organização do campo AllowLocal for igual à organização do utilizador autenticado e o seu valor for *true*, o comprador tem acesso à localização, caso contrário permanece anónima. Podemos ver o resultado em 4.39a e 4.41a.

4.3.6.3 Expiração do tempo de acesso aos dados do sensor

Como dito em 4.3.6.1, o utilizador adquire os dados por períodos de tempo. Para tal, criámos um *trigger* na base de dados que corre uma função em períodos de um minuto, assim que o tempo de aquisição dos dados é expirado, é retirado o comprador do campo buyUser do sensor. A resposta com a atualização é enviada (6)(7) para a página. Aqui, como o e-mail do utilizador autenticado (8)(9) não coincide com o e-mail no campo buyUSer, o sensor é retirado da página As Minhas

Compras e o utilizador perde o acesso aos dados. Durante esse período de tempo é permitido exportar os dados do sensor para CSV. A informação de exportação dos dados é registada na página "Histórico de Transações", através de um processo idêntico ao descrito em 4.3.6.1.

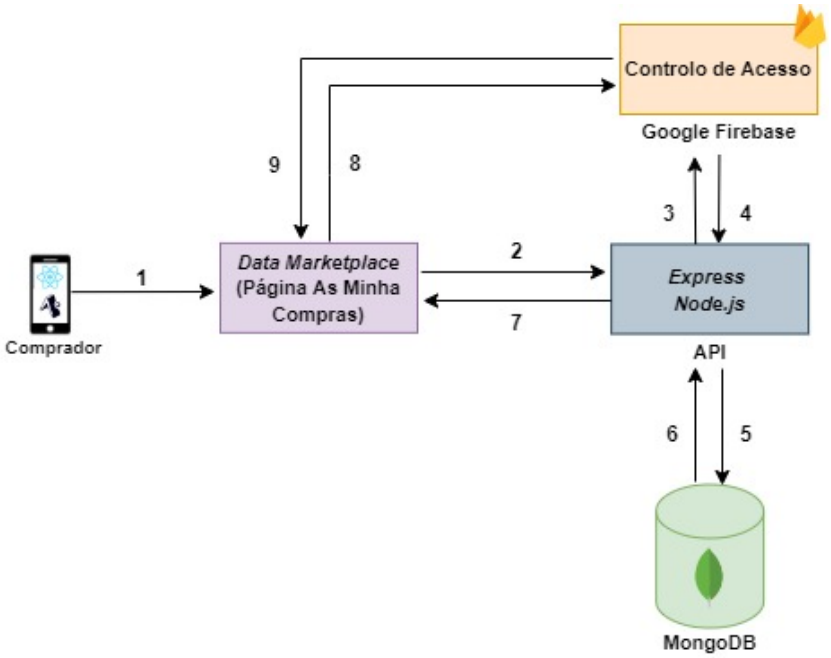


Figura 4.50: Fluxo para explicação das secções 4.3.6.2 e 4.3.6.3

4.3.6.4 Informações do sensor

Se o comprador adquiriu dados estatísticos, são realizados pedidos à API para obtenção de média, valor máximo e valor mínimo (2). O JWT *token* é verificado (3)(4) e a API faz o pedido dos valores estatísticos ao componente de agregação e controlo dos dados (5). Este, que também funciona como uma API, faz o pedido à base de dados dos sensores das coleções (6), que lhe envia os dados (7) para realização dos respetivos cálculos. O componente de agregação e controlo dos dados envia uma resposta com os cálculos estatísticos (8), que são posteriormente enviados (11) e apresentados ao utilizador como visível na Fig 4.41. No caso de dados estatísticos com acesso à localização exata do sensor, a primeira parte do processo é igual à anteriormente descrita. No entanto, são também recuperadas as organizações com valor *true* no campo AllowLocal (9)(10) para que se possam comparar com a organização do utilizador autenticado (12)(13) e permitir-lhe o acesso à localização do sensor. Um exemplo do resultado deste processo pode ser visualizado nas Fig 4.39b e Fig 4.41b.

No caso de o utilizador ter adquirido os conjuntos de medições (medições geradas em tempo real), podemos ignorar os fluxos 5, 6, 7, 8 e ter em atenção os fluxos 9 e 10, uma vez que o pedidos feitos à API apenas implicam a obtenção dos conjuntos de medições. No caso de conjunto de medições com acesso à localização exata do sensor, o processo utiliza os mesmo passos com adição da recuperação das organizações com valor *true* no campo AllowLocal (9)(10) para que se

possam comparar com a organização do utilizador autenticado (12)(13) e permitir-lhe o acesso à localização do sensor como demonstrado na Fig 4.40a.

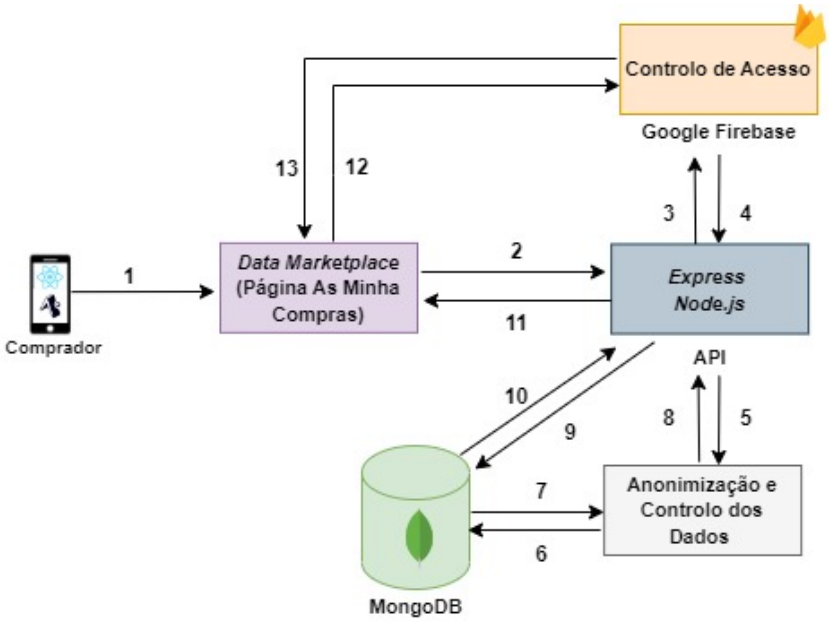


Figura 4.51: Fluxo para apresentação da informação dos dados do sensor comprado

Capítulo 5

Resultados e análise

Neste capítulo vamos abordar o questionário utilizado para obter a opinião das pessoas relativamente à aplicação desenvolvida na dissertação. Com a realização deste questionário, pretendemos compreender se as diversas funcionalidades da aplicação foram desenvolvidas de uma forma percetível e de fácil manuseamento. Pretendemos também perceber qual a opinião dos inquiridos, enquanto vendedores no *data marketplace*, no que diz respeito ao controlo que estes têm sobre os seus dados, como por exemplo, sobre a granularidade dos dados. No que concerne ao inquirido enquanto comprador, pretendemos compreender se a forma como o *data marketplace* oferece e disponibiliza os dados corresponde ao expectável.

Este questionário foi concebido através do Google Forms [20]. Esta é uma plataforma desenvolvida pela Google muito intuitiva que permite criar questionários de uma forma simples. Estes, podem ser partilhados via e-mail ou por meio de um *link*. O nosso questionário é dividido em nove secções. Sete destas secções correspondem às páginas que constituem a nossa aplicação, enquanto que as restantes dizem respeito aos sensores em si. As perguntas apresentadas no questionário são as seguintes:

- Página de Registo:
  - P1: Registou-se facilmente na plataforma?
  - P2: Percebeu a função do parâmetro "Objetivo"?
- Página de *Login*:
  - P3: Percebeu imediatamente que a autenticação era e-mail e palavra-passe?
  - P4: Realizou a autenticação facilmente?
- *Menu* da aplicação:
  - P5: Conseguiu facilmente aceder à página de comercialização dos dados?
  - P6: Como vendedor - Identificou facilmente a página dos dados dos sensores que possui?

  - P7: Como comprador - Identificou facilmente a página dos sensores comprados?

- Página "Os Meus Sensores":
  - P8: Percebeu que sensores possuía?

- Sensor (Esta secção corresponde aos sensores quando clicados da Página "Os Meus Sensores"):
  - P9: Definiu facilmente a informação que o sensor apresenta para as organizações pretendidas?
  - P10: Percebeu a função do parâmetro “Acesso à Localização Após Compra"?
  - P11: Conseguiu escolher para que organizações o sensor ficaria disponível para compra?
  - P12: Conseguiu definir diferentes níveis de acesso aos dados do sensor para as organizações?
  - P13: Conseguiu colocar os dados do sensor à venda?

- Página "*Marketplace*":
  - P14: Conseguiu diferenciar os tipos de sensores através dos ícones apresentados?
  - P15: Como vendedor/comprador - Conseguiu distinguir os seus sensores dos sensores dos restantes utilizadores?

- Sensor (Esta secção corresponde aos sensores quando clicados da Página "*Marketplace*"):
  - P16: Como comprador - Conseguiu facilmente perceber os dados que o sensor disponibiliza?
  - P17: Percebeu que foram aplicadas técnicas de agregação aos dados?
  - P18: Como comprador - Percebeu a função do parâmetro “Acesso aos Dados”?
  - P19: Como comprador - Conseguiu adquirir os dados do sensor?

- Página "Histórico de Transações":
  - P20: Percebeu facilmente a função da Página “Histórico de Transações”?
  - P21: Conseguiu ver as entidades que compraram dados dos sensores?
  - P22: Conseguiu ver as entidades que venderam dados dos sensores?

- Página "As Minhas Compras":
  - P23: Diferenciou facilmente o tipo dos sensores comprados?
  - P24: As informações apresentadas pelo sensor comprado correspondem as expetativas?

Tendo em conta que o nosso sistema poderá ser utilizado por qualquer tipo de utilizador, até mesmo por uma pessoa com pouquíssimo conhecimento de sistemas semelhantes, decidimos então recolher algum *feedback* acerca da usabilidade da nossa aplicação. Para isso selecionamos um

conjunto diverso de pessoas, desde estudantes do Departamento de Ciência de Computadores da *Faculdade de Ciências da Universidade do Porto* (FCUP), algumas pessoas com conhecimento na área da *Internet of Things* (IoT), e também algumas pessoas sem qualquer tipo de experiência em aplicações no género.

Após a recolha das respostas dos inquiridos, procedemos a uma análise das mesmas, de forma a poder detetar algumas lacunas ou pontos de maior importância para trabalho futuro, por serem funcionalidades de difícil notoriedade ou compreensão. Para isso, iremos em seguida apresentar as questões mais críticas deste questionário e tentar perceber pelas mesmas, as dificuldades sentidas pelos utilizadores quando confrontados com estas funcionalidades. É preciso notar que todas as perguntas são de escolha múltipla e as respostas a cada uma, são dadas com base no nível de dificuldade sentido pelos inquiridos após a utilização da aplicação. Assim, as opções **baixa**, **média** e **alta** correspondem, respetivamente, a dificuldade baixa, média e alta. Iremos de seguida, apresentar os resultados obtidos da realização do questionário.

As respostas obtidas da pergunta **P9:"Definiu facilmente a informação que o sensor apresenta para as organizações pretendidas?"**, mostram que esta funcionalidade foi claramente uma das menos compreendidas. Após uma revisão à mesma, podemos entender que provavelmente o parâmetro "Conjunto de Medições", assim como o ícone que sucede estes parâmetros, não são percetíveis na função que desempenham. Sendo esta, mostrar a lista das organizações registadas na aplicação até ao momento. Uma das sugestões propostas por um dos inquiridos foi a eliminação do parâmetro "Conjunto de Medições". Assim, indicaríamos que por padrão o sensor está programado para apresentar valores gerados em tempo real. Dessa forma, o utilizador apenas teria de escolher para que organizações o sensor iria disponibilizar dados estatísticos.

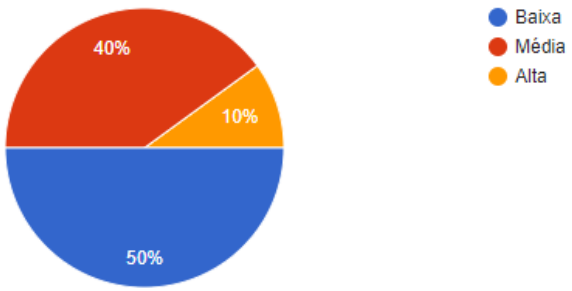


Figura 5.1: P9:"Definiu facilmente a informação que o sensor apresenta para as organizações pretendidas?"

Outro ponto importante de analisar são as respostas à **P11:"Conseguiu escolher para que organizações o sensor ficaria disponível para compra?"**. Através delas, reparamos que embora a maior parte dos inquiridos considera esta uma funcionalidade bem intuitiva, 40% deles (cerca de 4 pessoas) consideraram que esta implementação deveria também ser revista. Com o *feedback* obtido, podemos concluir que esta funcionalidade pode não estar

implementada da maneira mais compreensível para todos os utilizadores da aplicação. Assim, tivemos em consideração as sugestões de melhoramento propostas pelos inquiridos. Como tal, esta funcionalidade poderia ser melhorada, de forma a que as organizações selecionadas nos parâmetro "Estatísticas"e"Conjunto de Medições", ficassem automaticamente selecionadas na opção de tornar o sensor disponível para compra, enquanto que as não selecionadas aparecessem bloqueadas.

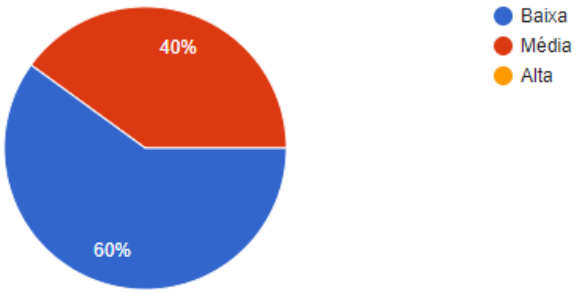


Figura 5.2: P11:"Conseguiu escolher para que organizações o sensor ficaria disponível para compra?"

Um dos pontos mais críticos desta análise recai nas respostas à pergunta **P15:"Como vendedor/comprador - Conseguiu distinguir os seus sensores dos sensores dos restantes utilizadores?"**. Nesta, apenas 30% dos inquiridos consideraram que esta funcionalidade tinha uma dificuldade baixa, ou seja, mais de dois terços dos inquiridos sentiram alguma dificuldade em conseguir distinguir os seus sensores dos sensores dos restantes utilizadores. Depois de revermos esta funcionalidade, podemos notar que existem alguns pontos onde esta possa ser melhorada para melhor distinção dos sensores. Neste momento, para que um vendedor/comprador possa distinguir os seus sensores dos restantes, tem de identificar aqueles em que aparece o botão "Adquirir dados do sensor". Nos sensores que pertencerem ao utilizador autenticado não irá aparecer esse botão, pois não faria sentido o utilizador readquirir estes dados. Como sugestão para trabalho futuro, poderíamos pensar em identificar de uma forma mais efetiva estes sensores através de, por exemplo, diferentes ícones, diferentes cores ou até diferentes filtros. Outra sugestão para o caso de um vendedor, seria retirar a visibilidade dos sensores de outros utilizadores, pois este perfil não teria permissões de compra.

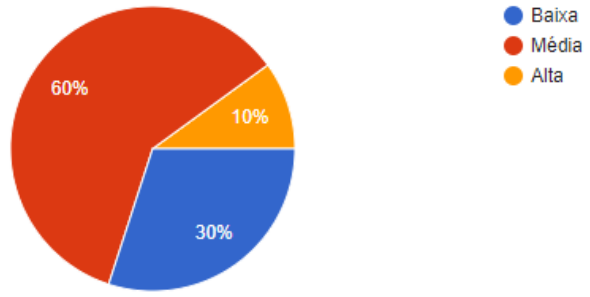


Figura 5.3: P15:"Como vendedor/comprador - Conseguiu distinguir os seus sensores dos sensores dos restantes utilizadores?"

Por fim, o ponto crucial e que requereu mais atenção e análise, foi a **P17:"Percebeu que foram aplicadas técnicas de agregação aos dados?"**. Esta é uma funcionalidade que necessita de alguma experiência por parte do utilizador em sistemas no género e de conhecimento no tema em geral. Assim, os inquiridos com alguma experiência na área, após abrirem o marcador que representa dados estatísticos em toda a região do Porto, perceberam de imediato que foram utilizadas técnicas de agregação de dados. O mesmo não aconteceu com os utilizadores com menos experiência no tema pois, ao não terem conhecimento sobre técnicas de agregação de dados, a funcionalidade não foi bem compreendida.

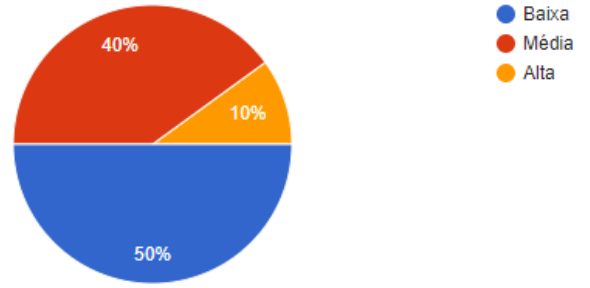


Figura 5.4: P17:"Percebeu que foram aplicadas técnicas de agregação aos dados?"

Ao realizarmos uma análise ao gráfico 5.5, podemos concluir que funcionalidades relacionadas com o registo (P1) e autenticação (P3, P4) do utilizador na plataforma foram utilizadas sem qualquer dificuldade. Da mesma forma, o objetivo da participação do utilizador na plataforma (P2) foi facilmente compreendido por todos os inquiridos, assim como a identificação das diferentes páginas que são apresentadas ao utilizador conforme este seja comprador, vendedor ou ambos (P6, P7). Também as ações de colocação de dados do sensor à venda (P13) e a compra de dados de sensores (P19) foram realizadas com facilidade. É também possível concluir que, através da descrição apresentada pelo sensor, os utilizadores não sentiram dificuldade em perceber qual a informação disponibilizada pelo mesmo (P16). Quanto à página correspondente ao registo de atividade, não houve qualquer dificuldade sentida por parte dos utilizadores na compreensão do

conteúdo desta (P20, P21, P22).

Contudo, algumas funcionalidades não foram tão fáceis de identificar ou de compreender o seu funcionamento, como é o caso da identificação da página de comercialização dos dados (P5), a identificação dos sensores pertencentes ao utilizador (P8), a função de determinados parâmetros como "Acesso à localização após compra"(P10) e "Acesso aos Dados"(P18). Também a informação apresentada pelo sensor após compra, pode não ter sido bem compreendida por todos os inquiridos (P24). Às questões relacionadas com as funcionalidades mencionadas, entre um a dois inquiridos responderam com dificuldade média e alta. Por outro lado, existe também outro ponto que pensamos que pudesse ser de mais difícil compreensão para o utilizador mas, que na verdade, se relevou ser bastante intuitivo. Este é o caso da pergunta **P12:"Conseguiu definir diferentes níveis de acesso aos dados do sensor para as organizações?"**, que demonstrou ser uma funcionalidade que o utilizador teve bastante facilidade em compreender. Em suma, podemos concluir que num panorama geral, os inquiridos consideram que a aplicação é bastante intuitiva e as suas funcionalidades são de fácil compreensão.

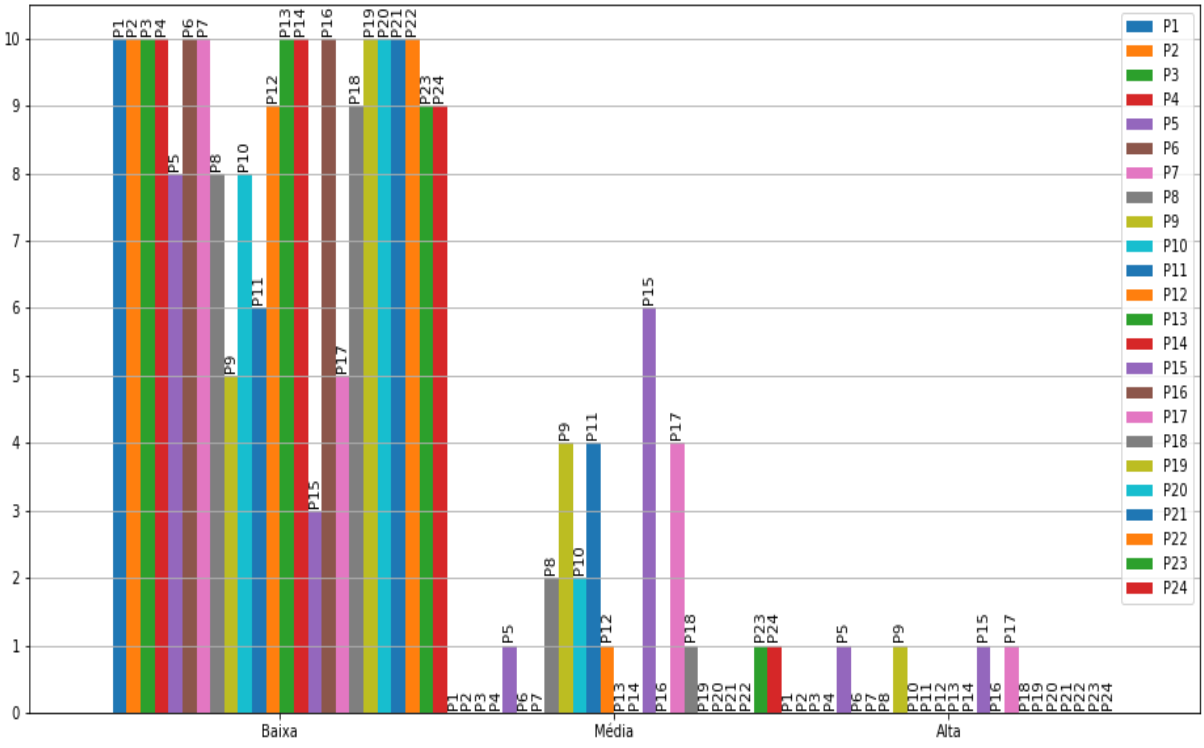


Figura 5.5: Apreciação geral da aplicação.

Capítulo 6

Conclusões

Por fim, após a descrição do trabalho realizado, podemos concluir que conseguimos desenvolver um *data marketplace* para *smart cities*, no qual dados de sensores *Internet of Things* (IoT) são colocados à venda e onde compradores como indivíduos, empresas ou até entidades governamentais que necessitam desses dados, os podem adquirir. Assim, esta plataforma é integrada por um mecanismo de autenticação que permite que os utilizadores sejam autenticados na plataforma e que nos dá garantias da identidade do utilizador. Desenvolvemos funcionalidades que permitem ao proprietário de sensores controlar os seus dados. Como tal, implementámos um mecanismo que permite ao utilizador escolher com quais organizações, registadas na plataforma até ao momento, pretendem partilhar os seus dados e também definir diferentes níveis de acesso aos dados para as organizações pretendidas. Também com o intuito de conceder controlo sobre os dados aos proprietários, implementámos algumas técnicas que permitem ocultar informações relevantes como a localização dos sensores. Numa tentativa de expandir as informações oferecidas nesta plataforma, aplicámos técnicas de agregação de dados, tornando assim possível a oferta de dados não processados ou dados estatísticos. Procedemos também à gestão da atividade dos utilizadores na plataforma, através da qual as ações de compra e venda de dados de sensores ficam registadas.

No entanto, para além destes aspetos positivos, houve certos objetivos que não foram cumpridos. Estes, passam pelo estudo e a implementação de técnicas de anonimização dos dados, assim como técnicas que permitissem aos proprietários dos dados controlar para que fins os seus dados são utilizados e também a autenticação dos dispositivos, neste caso o simulador de dados IoT na plataforma.

Posto isto, como trabalho futuro seria importante cumprir com os objetivos inicialmente propostos e, para tal, aproveitar algumas sugestões propostas pelos inquiridos e conclusões retiradas no capítulo 5, de forma a melhorar a usabilidade da aplicação e tornar este sistema mais completo. Como anteriormente referido, um dos pontos mais importantes para trabalho futuro, seria a pesquisa e implementação de técnicas de anonimização dos dados para que estes estivessem devidamente anonimizados na base de dados, antes destes serem comercializados, de forma a preservar a segurança e privacidade dos utilizadores. A implementação de mecanismos que

permitam ao utilizador transparência na usabilidade dos seus dados é outro ponto fundamental a ser desenvolvido. Algumas melhorias poderiam ser feitas nomeadamente no mecanismo de atribuição de permissões de acesso às organizações, de forma o proprietário de um determinado sensor, não tenha de estar constantemente a verificar se novas organizações começaram a sua atividade na plataforma para lhes conceder ou não o acesso aos dados do sensor. Uma outra melhoria recai sobre o desenvolvimento de estatísticas dos dados dos sensores em determinados intervalos de tempo, como por exemplo, a temperatura média, máxima e mínima do sensor A desde o dia 1 de outubro de 2022 até 5 de novembro de 2022 ou até de mês em mês desde a data de criação até ao presente. Um último ponto bastante importante a considerar, seria a integração de todo este sistema com sensores e dispositivos reais de forma a que esta nossa aplicação consiga também interagir com dados reais.

Apêndice A

Blocos de Código

```
import os
from pymongo import MongoClient
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred)

firestore_db = firestore.client()

users_ref = firestore_db.collection('users')
docs = users_ref.stream()
list = []
for doc in docs:
    organization = u'{}'.format(doc.to_dict()['organization'])
    list.append(organization)
list2 = set(list)

organizations_dictionary = dict.fromkeys(list2, False)
```

Bloco de Código A.1: Comunicação e recuperação das organizações contidas na Firestore.

```
for msg in consumer:
   if msg.topic == "temptopic":
      record = json.loads(msg.value)
      id = record['id']
      lat = record['lat']
      long = record['long']
      temperature = record['temperature']
      owner = record['owner']
      creationDate = record['creationDate']

      tempbucket_rec = {'temperature':temperature }
      deviceId = id
      latitude = lat
      longitude = long
      db.temp_bucket.update_many(
         {'deviceId': deviceId, 'creationDate': creationDate, 'latitude'
             :latitude, 'longitude' :longitude, 'owner' :owner,
             'organizations': organizations_dictionary, 'allowLocal':
             organizations_dictionary , 'allowInfo': organizations_dictionary,
             'buyUser' :[],
         {
            '$push': {'samples': tempbucket_rec},
         },
         upsert=True
      )
```

Bloco de Código A.2: Inserção em *buckets* dos conjuntos de dados de sensores de temperatura na base de dados.

```
elif msg.topic == "routestopic":
    record = json.loads(msg.value)
    id = record['id']
    owner = record['owner']
    date = record['date']
    coordinates = record['coordinates']

    route = {'id':id, 'date': date, 'owner': owner, 'coordinates':coordinates,
             'buyUser':[], 'organizations': organizations_dictionary}
    db.routes.insert_one(route)
```

Bloco de Código A.3: Inserção dos dados de percursos de trotinetas elétricas.

```
app.get('/api/temp_bucket', verifyToken, async (req, resp) => {
    try {
        await database.collection('temp_bucket').find({}).toArray((err, result)
            => {
            if (err) throw err
            resp.status(200).send(result)
        })
    }
    catch (e) {
        resp.status(400).send(e)
    }
})
```

Bloco de Código A.4: Método para recuperação dos dados de sensores de temperatura.

```
app.post('/api/transactions', verifyToken, async (req, resp) => {
    try {
        let transaction = await database.collection('transactions').insertOne({
            transactions: req.body })
        resp.status(200).send(transaction)
    }
    catch (e) {
        resp.status(400).send(e)
    }
})
```

Bloco de Código A.5: Método para inserção de dados das transações na base de dados.

```
app.patch('/api/updatebuyuser/:bucket/:sensor', verifyToken, async (req, resp)
    => {
    try {
        let update = await database.collection(req.params.bucket)
            .updateOne({ deviceId: req.params.sensor },
                { $push: { buyUser: req.body } })
        resp.status(200).send(update)
    }
    catch (e) {
        resp.status(400).send(e)
    }
})
```

Bloco de Código A.6: Método para atualização do campo buyUser na base de dados.

```
const verifyToken = (req, res, next) => {
    const token = req.headers.authorization.split(' ')[1]
    try {
        admin.auth().verifyIdToken(token)
            .then(result => {
                if (result) {
                    return next()
                }
            }).catch(err => {
                return res.sendStatus(401)
            })
    } catch (e) {
        return res.sendStatus(500)
    }
}
```

Bloco de Código A.7: Função para verificação dos *tokens* de acesso.

```
app.get('/api/temp_bucketsensoravg', async (req, resp) => {
    try {
        request('http://192.168.1.101:9090/datacontrol/temp_bucketsensoravg', {
            json: true }, (err, res, body) => {
            if (err) throw err
            resp.status(200).send(body)
        });
    } catch (e) {
        resp.status(500).send(e)
    }
})
```

Bloco de Código A.8: Pedido da média de cada sensor de temperatura ao componente de agregação e controlo de acesso.

```
app.get('/datacontrol/hum_sensortotalavg', async (req, resp) => {
    try {
        var queryTotalAvgHumidity = [{ $unwind: "$samples" }, { $group: { _id:
            "$id", avg_val: { $avg: "$samples.humidity" } } }];
        await database.collection('allSensors').aggregate(queryTotalAvgHumidity)
        .toArray(function (err, result) {
            if (err) throw err;
            resp.status(200).send(result)
        })
    }
    catch (e) {
        resp.status(400).send(e)
    }
})
```

Bloco de Código A.9: Obtenção do valor médio de humidade relativa registado na região do Porto.

```
app.get('/datacontrol/temp_sensortotalmax', async (req, resp) => {
    try {
        var queryTotalMaxTemperature = [{ $unwind: "$samples" }, { $group: {
            _id: "$id", max_val: { $max: "$samples.temperature" } } }];
        await
            database.collection('allSensors').aggregate(queryTotalMaxTemperature)
        .toArray(function (err, result) {
            if (err) throw err;
            resp.status(200).send(result)
        })
    }
    catch (e) {
        resp.status(400).send(e)
    }
})
```

Bloco de Código A.10: Obtenção do valor máximo de temperatura registado na região do Porto.

```
app.get('/datacontrol/atmpressure_sensortotalmin', async (req, resp) => {
    try {
        var queryTotalMinAtmPressure = [{ $unwind: "$samples" }, { $group: {
            _id: "$id", min_val: { $min: "$samples.atmpressure" } } }];
        await
            database.collection('allSensors').aggregate(queryTotalMinAtmPressure)
        .toArray(function (err, result) {
            if (err) throw err;
            resp.status(200).send(result)
        })
    }
    catch (e) {
        resp.status(400).send(e)
    }
})
```

Bloco de Código A.11: Obtenção do valor mínimo de pressão atmosférica registado na região do Porto.

```
app.get('/datacontrol/temp_bucketsensoravg', async (req, resp) => {
    try {
        var queryAvgTemperature = [{ $unwind: "$samples" }, { $group: { _id:
            "$deviceId", avg_val: { $avg: "$samples.temperature" } } }];
        await database.collection('temp_bucket').aggregate(queryAvgTemperature)
        .toArray((err, result) => {
            if (err) throw err
            resp.status(200).send(result)
        })
    }
    catch (e) {
        resp.status(400).send(e)
    }
})
```

Bloco de Código A.12: Obtenção do valor médio de temperatura para cada sensor na região do Porto.

```
app.get('/datacontrol/hum_bucketsensormax', async (req, resp) => {
    try {
        var queryMaxHumidity = [{ $unwind: "$samples" }, { $group: { _id:
            "$deviceId", max_val: { $max: "$samples.humidity" } } }];
        await database.collection('hum_bucket').aggregate(queryMaxHumidity)
        .toArray(function (err, result) {
            if (err) throw err;
            resp.status(200).send(result)
        })
    }
    catch (e) {
        resp.status(400).send(e)
    }
});
```

Bloco de Código A.13: Obtenção do valor máximo de humidade relativa para cada sensor na região do Porto.

```
app.get('/datacontrol/atmpressure_bucketsensormin', async (req, resp) => {
    try {
        var queryMinAtmpPressure = [{ $unwind: "$samples" }, { $group: { _id:
            "$deviceId", min_val: { $min: "$samples.atmpressure" } } }];
        await database.collection('atmpressure_bucket')
        .aggregate(queryMinAtmpPressure)
        .toArray(function (err, result) {
            if (err) throw err;
            resp.status(200).send(result)
        })
    }
    catch (e) {
        resp.status(400).send(e)
    }
});
```

Bloco de Código A.14: Obtenção do valor mínimo de pressão atmosférica para cada sensor na região do Porto.

Bibliografia

[1] Secaas implementation guidance, category 1: Identity and access management. Technical report, Cloud Security Alliance, September 2012.

[2] adrianmuino. Iot-data-simulator. https://github.com/adrianmuino/IoT-Data-Simulator, 2020. Acedido em: 2021-12-20.

[3] Amir H Alavi, Pengcheng Jiao, William G Buttlar, and Nizar Lajnef. Internet of things-enabled smart cities: State-of-the-art and future trends. *Measurement*, 129:589–606, 2018.

[4] Pelin Angin, Bharat Bhargava, Rohit Ranchal, Noopur Singh, Mark Linderman, Lotfi Ben Othmane, and Leszek Lilien. An entity-centric approach for privacy and identity management in cloud computing. In *2010 29th IEEE symposium on reliable distributed systems*, pages 177–183. IEEE, 2010.

[5] Mike Bainbridge and James Moar. Smart cities technologies shaping the urban future. Technical report, Juniper Research Ltd, January 2022.

[6] John Barkley. Comparing simple role based access control models and access control lists. In *Proceedings of the second ACM workshop on Role-based access control*, pages 127–132, 1997.

[7] Rômy Bergman, Antragama Ewa Abbas, Sven Jung, Claudia Werker, and Mark de Reuver. Business model archetypes for data marketplaces in the automotive industry. *Electronic Markets*, pages 1–19, 2022.

[8] Anup W Burange and Harshal D Misalkar. Review of internet of things in development of smart cities with data management & privacy. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 189–195. IEEE, 2015.

[9] Simin Cai, Barbara Gallina, Dag Nyström, and Cristina Seceleanu. Data aggregation processes: a survey, a taxonomy, and design guidelines. *Computing*, 101(10):1397–1429, 2019.

[10] Alan Colman, M Jabed Morshed Chowdhury, and Mohan Baruwal Chhetri. Towards a trusted marketplace for wearable data. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pages 314–321. IEEE, 2019.

[11] OpenID Connect. Welcome to openid connect. https://openid.net/connect/, 2022. Acedido em: 2022-10-10.

[12] Jin Cui, Khaled Boussetta, and Fabrice Valois. Classification of data aggregation functions in wireless sensor networks. *Computer Networks*, 178:107342, 2020.

[13] Edward Curry. The big data value chain: definitions, concepts, and theoretical approaches. In *New horizons for a data-driven economy*, pages 29–37. Springer, Cham, 2016.

[14] Dipankar Dasgupta, Arunava Roy, Abhijit Nag, et al. *Advances in user authentication*. Springer, 2017.

[15] Dawex. Security and privacy. https://www.dawex.com/en/security-privacy/, 2022. Acedido em: 2022-01-15.

[16] Dawex. Data marketplace. https://www.dawex.com/en/data-exchange-platform/data-marketplace/, 2022. Acedido em: 2022-01-16.

[17] Dawex. Global data marketplace. https://www.dawex.com/en/why-data-exchange/success-stories/global-data-marketplace/, 2022. Acedido em: 2022-01-15.

[18] Dawex. Industries. https://www.dawex.com/en/industries/, 2022. Acedido em: 2022-01-15.

[19] Johannes Deichmann, Kersten Heineke, Thomas Reinbacher, and Dominik Wee. Creating a successful internet of things data marketplace. *McKinsey & Company*, 2016.

[20] A Demarest. What are google forms? everything you need to know about google workspace's online form builder. https://www.businessinsider.com/what-is-google-forms, 2021. Acedido em: 2022-09-01.

[21] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003.

[22] Express. Using middleware. https://expressjs.com/en/guide/using-middleware.html, 2017. 2022-04-04.

[23] Express. Routing. https://expressjs.com/en/guide/routing.html, 2017. 2022-04-04.

[24] Abou Zakaria Faroukhi, Imane El Alaoui, Youssef Gahi, and Aouatif Amine. Big data monetization throughout big data value chain: a comprehensive review. *Journal of Big Data*, 7(1):1–22, 2020.

[25] Daniel Fernandez, Ariel Futoransky, Gustavo Ajzenman, Matias Travizano, and Carlos Sarraute. Wibson protocol for secure data exchange and batch payments. *arXiv preprint arXiv:2001.08832*, 2020.

[26] Daniel Fett, Ralf Küsters, and Guido Schmitz. A comprehensive formal security analysis of oauth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1204–1215, 2016.

[27] Firebase. Choose a database: Cloud firestore or realtime database. https://firebase.google.com/docs/database/rtdb-vs-firestore, 2022. Acedido em: 2022-04-10.

[28] Firebase. Firebase authentication. https://firebase.google.com/docs/auth, 2022. Acedido em: 2022-04-10.

[29] Firebase. Verify id tokens. https://firebase.google.com/docs/auth/admin/verify-id-tokens, 2022. Acedido em: 2022-04-10.

[30] IOTA Foundation. Introducing masked authenticated messaging. https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e/, 2017. Acedido em: 2022-01-07.

[31] IOTA Foundation. The tangle: an illustrated introduction. https://blog.iota.org/the-tangle-an-illustrated-introduction-4d5eae6fe8d4/, 2018. Acedido em: 2022-01-07.

[32] IOTA Foundation. Part 1: Iota data marketplace — update. https://blog.iota.org/part-1-iota-data-marketplace-update-5f6a8ce96d05/, 2019. Acedido em: 2022-01-05.

[33] IOTA Foundation. Part 4: Cloud backend configuration. https://blog.iota.org/the-iota-data-marketplace-a-tech-intro-part4-47b608c527c9/, 2019. Acedido em: 2022-01-07.

[34] IOTA Foundation. Working with trust, integrity and transparency. https://blog.iota.org/working-with-trust-integrity-and-transparency/, 2022. Acedido em: 2022-05-07.

[35] Michael Fruhwirth, Michael Rachinger, and Emina Prlja. Discovering business models of data marketplaces. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.

[36] Jon Glasco. Smartcitiesworld city profile – barcelona. Technical report, SmartCitiesWorld, November 2019.

[37] IBA Group. Iot-data-simulator. https://github.com/IBA-Group-IT/IoT-data-simulator, 2018. Acedido em: 2021-12-20.

[38] Pooja Gupta, S Kanhere, and Raja Jurdak. A decentralized iot data marketplace. *arXiv preprint arXiv:1906.01799*, 2019.

[39] Mohammad Hasan. State of iot 2022. https://iot-analytics.com/number-connected-iot-devices//, 2022. Acedido em: 2022-08-12.

[40] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162):1–54, 2013.

[41] Lihua Huang, Yifan Dou, Yezheng Liu, Jinzhao Wang, Gang Chen, Xiaoyang Zhang, and Runyin Wang. Toward a research framework to conceptualize data as a factor of production: the data marketplace perspective. *Fundamental Research*, 1(5):586–594, 2021.

[42] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, pages 791–798. IEEE, 2008.

[43] i3. Account creation. https://i3sdk.readthedocs.io/en/latest/account.html, 2019. Acedido em: 2022-02-01.

[44] Iberdrola. Smart cities: the technological revolution reaches the cities. https://www.iberdrola.com/innovation/smart-cities, 2022. Acedido em: 2022-09-25.

[45] RP Janani, K Renuka, A Aruna, et al. Iot in smart cities: A contemporary survey. *Global Transitions Proceedings*, 2(2):187–193, 2021.

[46] Audun Jøsang and Simon Pope. User centric identity management. In *AusCERT Asia Pacific information technology security conference*, volume 22, page 2005. Citeseer, 2005.

[47] Apache Kafka. Everything you need to know about kafka in 10 minutes. https://kafka.apache.org/intro, 2022. Acedido em: 2022-02-10.

[48] William Kennedy and Aspen Olmsted. Three factor authentication. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 212–213. IEEE, 2017.

[49] Pantelis Koutroumpis, Aija Leiponen, and Llewellyn DW Thomas. The (unfulfilled) potential of data marketplaces. Technical report, ETLA working papers, 2017.

[50] Pantelis Koutroumpis, Aija Leiponen, and Llewellyn DW Thomas. Markets for data. *Industrial and Corporate Change*, 29(3):645–660, 2020.

[51] Bhaskar Krishnamachari, Jerry Power, Cyrus Shahabi, and Seon Ho Kim. Iot marketplace: A data and api market for iot devices. *University of Southern California: Los Angeles, CA, USA*, 2017.

[52] Bhaskar Krishnamachari, Jerry Power, Seon Ho Kim, and Cyrus Shahabi. I3: An iot marketplace for smart communities. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 498–499, 2018.

[53] L Krishnamachari, Deborah Estrin, and Stephen Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings 22nd international conference on distributed computing systems workshops*, pages 575–578. IEEE, 2002.

[54] Information Technology Laboratory. security assertion markup language (saml). https://csrc.nist.gov/glossary/term/security_assertion_markup_language. Acedido em: 2022-08-15.

[55] Information Technology Laboratory. Access control policy and implementation guides. https://csrc.nist.gov/Projects/Access-Control-Policy-and-Implementation-Guides, 2016. Acedido em: 2022-07-13.

[56] Chun Sing Lai, Youwei Jia, Zhekang Dong, Dongxiao Wang, Yingshan Tao, Qi Hong Lai, Richard TK Wong, Ahmed F Zobaa, Ruiheng Wu, and Loi Lei Lai. A review of technical standards for smart cities. *Clean Technologies*, 2(3):290–310, 2020.

[57] Sebastian Lawrenz, Priyanka Sharma, and Andreas Rausch. The significant role of metadata for data marketplaces. In *International Conference on Dublin Core and Metadata Applications*, pages 95–101, 2020.

[58] Qinghua Li and Guohong Cao. Efficient and privacy-preserving data aggregation in mobile sensing. In *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, 2012.

[59] Jean-Luc Marichal. Aggregation functions for decision making. *arXiv preprint arXiv:0901.4232*, 2009.

[60] Mark Masse. *REST API design rulebook: designing consistent RESTful web service interfaces*. "O'Reilly Media, Inc.", 2011.

[61] Guillaume Meister. Iot-generator-mongodb. https://github.com/Wilpapa/IoT-generator-mongodb, 2020. Acedido em: 2021-12-20.

[62] Chui Michael, Löffler Markus, and Roberts Roger. The internet of things. *McKinsey Quarterly*, 2(2010):1–9, 2010.

[63] Krešimir Mišura and Mario Žagar. Data marketplace for internet of things. In *2016 International Conference on Smart Systems and Technologies (SST)*, pages 255–260. IEEE, 2016.

[64] Mohamed A Mohamed, Obay G Altrafi, and Mohammed O Ismail. Relational vs. nosql databases: A survey. *International Journal of Computer and Information Technology*, 3 (03):598–601, 2014.

[65] Ishaq Azhar Mohammed. Intelligent authentication for identity and access management: a review paper. *International Journal of Managment, IT and Engineering (IJMIE)*, 3(1): 696–705, 2013.

[66] MongoDB. Get started with atlas. https://www.mongodb.com/docs/atlas/getting-started/, 2022. Acedido em: 2022-03-23.

[67] MongoDB. What is a document database? https://www.mongodb.com/document-databases, 2022. Acedido em: 2022-02-22.

[68] MongoDB. What are database triggers? https://www.mongodb.com/features/database-triggers, 2022. Acedido em: 2022-07-10.

[69] MongoDB. Relational vs. non-relational databases. https://www.mongodb.com/compare/relational-vs-non-relational-databases, 2022. Acedido em: 2022-02-22.

[70] Yoonjong Na, Yejin Joo, Heejo Lee, Xiangchen Zhao, Kurian Karyakulam Sajan, Gowri Ramachandran, and Bhaskar Krishnamachari. Enhancing the reliability of iot data marketplaces through security validation of iot devices. In *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 265–272. IEEE, 2020.

[71] Andy Neumann, Nuno Laranjeiro, and Jorge Bernardino. An analysis of public rest web service apis. *IEEE Transactions on Services Computing*, 14(4):957–970, 2018.

[72] SmartCitiesWorld news team. Seoul deploys smart city services to promote safety and wellbeing. https://www.smartcitiesworld.net/health-and-social-care/health-and-social-care/seoul-deploys-smart-city-services-to-promote-safety-and-wellbeing, 2022. Acedido em: 2022-09-25.

[73] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. *Business & Information Systems Engineering*, 59(3):183–187, 2017.

[74] United Nations Department of Economic and Social Affairs. 2014 revision of the world urbanization prospects. https://www.un.org/en/development/desa/publications/2014-revision-world-urbanization-prospects.html, 2014. Acedido em: 2022-08-12.

[75] Okta. Introduction to json web tokens. https://jwt.io/introduction. Acedido em: 2022-07-10.

[76] Maria Papathanasaki, Leandros Maglaras, and Nick Ayres. Modern authentication methods: A comprehensive survey. 2022.

[77] NYC Environmental Protection. Resources for homeowners - water meter faqs. https://www.nyc.gov/site/dep/pay-my-bills/water-meter-faqs.page, 2022. Acedido em: 2022-09-25.

[78] Gowri Sankar Ramachandran, Rahul Radhakrishnan, and Bhaskar Krishnamachari. Towards a decentralized data marketplace for smart cities. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–8. IEEE, 2018.

[79] Mauricio A Ramírez-Moreno, Sajjad Keshtkar, Diego A Padilla-Reyes, Edrick Ramos-López, Moisés García-Martínez, Mónica C Hernández-Luna, Antonio E Mogro, Jurgen Mahlknecht, José Ignacio Huertas, Rodrigo E Peimbert-García, et al. Sensors for sustainable smart cities: A review. *Applied Sciences*, 11(17):8198, 2021.

[80] M Mazhar Rathore, Awais Ahmad, Anand Paul, and Seungmin Rho. Urban planning and building smart cities based on the internet of things using big data analytics. *Computer networks*, 101:63–80, 2016.

[81] Juniper Research. World's no 1 smart city for 2022: Shanghai - thanks to world-leading citizen data platform. https://www.juniperresearch.com/press/worlds-no-1-smart-city-for-2022-shanghai, 2022. Acedido em: 2022-09-25.

[82] Leonard Richardson and Sam Ruby. *RESTful web services*. "O'Reilly Media, Inc.", 2008.

[83] Karen Rose, Scott Eldridge, and Lyman Chapin. The internet of things: An overview. *The internet society (ISOC)*, 80:1–50, 2015.

[84] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer, 2000.

[85] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[86] Fabian Schomm, Florian Stahl, and Gottfried Vossen. Marketplaces for data: an initial survey. *ACM SIGMOD Record*, 42(1):15–26, 2013.

[87] Mohamed Seliem, Khalid Elgazzar, and Kasem Khalil. Towards privacy preserving iot environments: a survey. *Wireless Communications and Mobile Computing*, 2018, 2018.

[88] Anuja Sharma, Sarita Sharma, and Meenu Dave. Identity and access management-a comprehensive study. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1481–1485. IEEE, 2015.

[89] Hai-bo Shen and Fan Hong. An attribute-based access control model for web services. In *2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)*, pages 74–79. IEEE, 2006.

[90] Bc Jakub Smolár and Cílem práce bylo rozšířit rámec Patriot. Data generator for iot testing. 2020.

[91] Patrícia Raquel Vieira Sousa. Privacy preserving middleware platform for iot. 2021.

[92] Markus Spiekermann. Data marketplaces: Trends and monetisation of data goods. *Intereconomics*, 54(4):208–216, 2019.

[93] Florian Stahl, Fabian Schomm, and Gottfried Vossen. The data marketplace survey revisited. Technical report, ERCIS Working Paper, 2014.

[94] Ali Sunyaev. Distributed ledger technology. In *Internet Computing*, pages 265–299. Springer, 2020.

[95] Saber Talari, Miadreza Shafie-Khah, Pierluigi Siano, Vincenzo Loia, Aurelio Tommasetti, and João PS Catalão. A review of smart cities based on the internet of things concept. *Energies*, 10(4):421, 2017.

[96] Temmermans. iot-device-simulator. https://github.com/Temmermans/iot-device-simulator, 2017. Acedido em: 2021-12-20.

[97] Llewellyn DW Thomas and Aija Leiponen. Big data commercialization. *IEEE Engineering Management Review*, 44(2):74–90, 2016.

[98] Juris Tihomirovs and Jānis Grabis. Comparison of soap and rest based web services using software evaluation metrics. *Information technology and management science*, 19(1):92–97, 2016.

[99] Matias Travizano, Carlos Sarraute, Mateusz Dolata, Aaron M French, and Horst Treiblmaier. Wibson: A case study of a decentralized, privacy-preserving data marketplace. In *Blockchain and Distributed Ledger Technology Use Cases*, pages 149–170. Springer, 2020.

[100] Jeannette M. Wing. The Data Life Cycle. *Harvard Data Science Review*, 1(1), jul 1 2019. https://hdsr.mitpress.mit.edu/pub/577rq08d.

[101] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[102] Xiangchen Zhao, Kurian Karyakulam Sajan, Gowri Sankar Ramachandran, and Bhaskar Krishnamachari. Demo abstract: the intelligent iot integrator data marketplace-version 1. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 270–271. IEEE, 2020.

[103] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105:475–491, 2020.