INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




SISTEMA DE NOTIFICAÇÕES EM TEMPO REAL

DANIELA VIEIRA E SILVA
Outubro de 2023

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Sistema de Notificações em Tempo Real

Daniela Vieira e Silva

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática,
Área de Especialização em Engenharia de Software

Orientador: Emanuel Silva

Porto, outubro 2023

Dedicatória

Esta tese é dedicada a todos aqueles que, de forma constritiva, contribuíram para o meu êxito académico.

Resumo

Atualmente, existe uma crescente necessidade de desenvolver aplicações informáticas que ajudam o ser humano na gestão das infraestruturas rodoviárias. A Armis possui uma forte presença nesta indústria que conta com vários produtos, sendo um deles o Drive Traffic Information. No entanto, este *software* não conta com um Sistema de Notificações em Tempo Real capaz de notificar as plataformas em que o Drive opera acerca das ocorrências nas infraestruturas.

Este documento concentra-se na implementação de uma solução para preencher esta lacuna detetada, incluindo todo o processo de implementação. Este começa pela fase de análise, que inclui um estudo e avaliação das tecnologias utilizadas e em uso pela organização. Em seguida é explicada como a arquitetura da solução foi construída para suportar a funcionalidade desejada. A implementação da solução é abordada em detalhe, descrevendo como as notificações em tempo real são incorporadas no *software*.

Por fim, o Sistema de Notificações em Tempo Real é submetido a avaliação, revelando que a solução alcança todos os objetivos propostos, contribuindo assim para uma gestão mais eficaz das infraestruturas rodoviárias.

Palavras-chave: Drive, Sistema de Notificações, MQTT, AMQP

Abstract

At the moment, there is an increasing demand for software applications that assist humans in managing road infrastructure. Armis has a strong presence in this industry with various products, one of which is Drive Traffic Information. However, this software lacks a Real Time Notification System capable of alerting the platforms where Drive operates about incidents in the infrastructure.

This document focuses on implementing a solution to address this identified gap, including all the implementation process. It starts with the analysis phase, which includes a study and evaluation of the technologies used and the ones adopted by the organization. Next, it´s explained how the architecture of the solution was design to support the desired functionality. The implementation of the solution is address in detail, describing how real-time notifications are incorporated into the software.

Finally, the Real Time Notification System undergoes evaluation, demonstrated that the solution successfully achieves all the proposed objectives, thus contributing to an effective management of road infrastructure.

Keyword: Drive, Notification System, MQTT, AMQP

Declaração de Integridade

Declaro que a presente dissertação de mestrado, é o resultado da minha própria pesquisa, realizada de forma ética e honesta.

Toda a informação apresentada neste trabalho foi obtida por meios legítimos e sem recurso à prática de plágio.

Qualquer ideia ou informação baseada em outras fontes está devidamente citada e referenciada de acordo com as normas académicas estabelecidas pela instituição de ensino.

Foram adotados os oito princípios do IEE-CS/ACM sobre a ética em Engenharia de Software e Práticas Profissionais (*Code of Ethics | IEEE Computer Society*, no date).

Índice

Lista de Figuras

Lista de Tabelas

Lista de Excertos de código

Acrónimos e Símbolos

Lista de Acrónimos

AMQP	*Advanced Message Queuing Protocol*
Drive	Drive Traffic Information
FFE	*Fuzzy Front End*
FIFO	*First-In-First-Out*
HOQ	*House of Quality*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IoT	*Internet of Things*
ISEP	Instituto Superior de Engenharia do Porto
ITS	*Intelligent Transport Systems*
M2M	*Machine-to-Machine*
MEI	Mestrado em Engenharia de Software
MQTT	*Message Queuing Telemetry Transport*
MSSQL	Microsoft SQL Server
MVC	Model-View-Controller
NCD	*New Concept Development*
NDP	*New Product Development*
OAV	Oficial de Assistência e Vigilância.
QFD	*Quality Function Deployment*
QoS	*Quality of Service*

REST	*Representational State Transfer*
SD	Diagrama de Sequência
SSD	Diagrama de Sequência de Sistema
TCP	*Transmission Control Protocol*
TLS/SSL	*Transport Layer Security/Secure Sockets Layer*
TMDEI	Tese/Dissertação/Estágio
UC	Casos de Uso
UML	*Unified Modelling Language*
URI	*Universal Resource Identifier*

1 Introdução

Neste primeiro capítulo é realizado um panorama geral do projeto. Numa primeira fase, o assunto é contextualizado a nível empresarial e aplicacional. De seguida o problema é identificado e apresentado de forma breve, revelando a abordagem para o solucionar. No final, é apresentada a estrutura deste documento com uma breve descrição de cada um dos capítulos.

1.1 Enquadramento

A presente dissertação descreve todo o trabalho realizado no âmbito da unidade curricular Tese/Dissertação/Estágio (TMDEI) do Mestrado em Engenharia de Software (MEI) do Instituto Superior de Engenharia do Porto (ISEP).

Este projeto surge no seguimento de uma proposta da Armis[1], uma organização fundada em 2005 e que se insere na área do desenvolvimento e implementação de soluções digitais complexas. A respetiva proposta diz respeito ao Drive Traffic Information (Drive), um sistema de gestão e controlo de tráfego e transportes.

1.1.1 Contexto empresarial

A Armis com sede no Porto, apresenta-se como uma entidade que se encontra na vanguarda na área de tecnologias de informação, dedicando-se ao desenvolvimento e implementação de soluções digitais.

[1] https://www.armis.pt/

De forma a acompanhar o ritmo elevado com que a tecnologia se desenvolve, a Armis, através do uso de tecnologias de informação, procura estar presente em várias áreas de intervenção, nomeadamente, transportes, desporto, serviços financeiros, etc. Posto isto, a organização atual foca-se em cinco grandes áreas (Armis | Armis, no date):

- **Information Tecnology (IT) –** focada no desenvolvimento de soluções tecnológicas como aplicações móveis e portais, com recurso a tecnologias mais recentes. Além disso, também oferece segurança, certificações e testes;
- **Intelligent Transport Systems (ITS)** – dedicada ao desenvolvimento de soluções avançadas que melhoram a gestão da mobilidade em sistemas de transporte complexos;
- **Digital Sport** – focada na conceção de soluções digitais em tecnologia desportiva, nomeadamente soluções de gestão desportiva e portais centrados nos adeptos;
- **Financial** – orientada ao desenvolvimento de soluções da Banca Digital que permitem às instituições financeiras gerir os desafios da segurança e inovação;
- **Ozono** – direcionada à criação de valor e inovação nas áreas de negócio em que a empresa opera.

A Armis conta com mais de 18 anos de experiência. Os seus mais de 200 colaboradores encontram-se distribuídos em 4 instalações (Porto, Lisboa, São Paulo e Utrecht) que visam satisfazer as necessidades dos 11 países em que opera.

O âmbito deste projeto surgiu da Armis ITS. Esta trabalha com vários clientes sendo um deles a Ascendi[2], entidade vocacionada para a gestão de ativos, serviços de cobrança de portagens e de operação e manutenção de infraestruturas rodoviárias. A Ascendi, com mais de 20 anos de experiência, atua em 6 concessões portuguesas (Norte, Costa de Prata, Beiras Litoral e Alta, Grande Porto, Grande Lisboa e Pinhal Interior) (Ascendi, no date).

1.1.2 Contexto aplicacional

Como já mencionado anteriormente, este projeto diz respeito ao Drive, produto desenvolvido pela Armis ITS, que daqui em diante será designada apenas por ITS.

[2] https://www.ascendi.pt/

Este sistema é caracterizado por um sistema inovador e seguro de Gestão da Mobilidade, em conformidade com as recomendações e objetivos dos planos de ação da ITS. Esta aplicação disponibiliza vários serviços, nomeadamente, Gestão de tráfego, Planeamento e obras, Parqueamento, Segurança e ainda Transportes de passageiros e carga (Armis | Drive, no date).

Para além do produto já implementado em clientes, a ITS está a desenvolver um novo sistema para satisfazer os serviços da mobilidade do futuro – Drive 3.0.

1.2 Descrição do Problema

O Drive dispõe de vários módulos que permitem o registo de diversos eventos, nomeadamente a criação de acidentes, o registo do envio de patrulhas para as infraestruturas, o registo de condições meteorológicas em que as infraestruturas se encontram, etc. Muitas das vezes, estes eventos implicam ações rápidas tanto por parte da central como por quem se encontra nas infraestruturas.

Com o crescente aumento do uso do telemóvel e dispositivos inteligentes em diferentes áreas, os peritos procuram aproveitar cada vez mais os benefícios que estas tecnologias trazem para otimizar o seu dia a dia de trabalho.

A área de ITS não é exceção e, em virtude das necessidades dos clientes e do crescente funcionamento do sistema, surge a necessidade que todos os dispositivos do Drive, incluindo Drive Web e Drive Mobile, estejam conectados entre si e em tempo real, permitindo que eles se comuniquem e troquem dados. Por outras palavras, sempre que ocorra uma ação por parte do Drive Mobile, todos os utilizadores conectados ao Drive Web e ao Drive Mobile devem ser imediatamente notificados, e vice-versa.

Embora exista um sistema de notificações no Drive Web, esse sistema apresenta limitações significativas, pois não se eestende ao Drive Mobile. Isto cria uma lacuna na comunicação entre os diferentes utilizadores e dispositivos do Drive. Por exemplo, quando ocorre um evento crítico, como um acidente, o sistema de notificações atual é incapaz de alertar imediatamente os utilizadores que estão em movimento e a utilizar o Drive Mobile. Isto pode resultar em atrasos na resposta a emergências e na coordenação de ações entre os utilizadores.

Posto isto, a substituição da tecnologia atual tornar-se-á não apenas desejável, mas essencial para garantir a eficiência e a eficácia do sistema Drive. Os utilizadores do Drive Web e do Drive Mobile precisam de estar interligados em tempo real, permitindo que todos recebam notificações instantâneas e atualizações importantes, independentemente do dispositivo que estão a utilizar. Esta integração entre os dispositivos proporcionará uma experiência mais completa e segura aos utilizadores, aumentado, assim, a atratividade e a comercialização do produto.

1.2.1 Objetivos

O objetivo é integrar no produto já existente, um sistema de notificações em tempo real capaz de resolver o problema em mãos na forma mais adequada.

Numa fase inicial, para a aplicação a desenvolver, foram definidos os seguintes objetivos:

1. Analisar o contexto do desenvolvimento do produto, como as tecnologias e ferramentas atualmente utilizadas;
2. Pesquisar soluções existentes no mercado atual que utilizem um sistema de notificações, em tempo real, entre aplicações;
3. Identificar e analisar as arquiteturas e tecnologias mais adequadas à resolução do problema;
4. Analisar e decidir, entre as alternativas, quais as arquiteturas e tecnologias que, potencialmente, serão mais adequadas à resolução do problema;
5. Criar um catálogo de eventos para que o Drive envie mensagens aos consumidores autorizados a subscreverem o evento;
6. Implementar um sistema de notificações no produto com a arquitetura e ferramenta escolhida.

1.3 Abordagem

A abordagem utilizada neste projeto é o desenvolvimento ágil de *software*, mais especificamente a metodologia *Scrum*.

Scrum permite que o desenvolvimento de *software* seja incremental e iterativo de forma a reduzir o tempo de entrega do produto e se adaptar a mudanças com maior facilidade durante as etapas de produção (Sachdeva, 2016).

Posto isto, para a construção do produto serão realizadas uma serie de iterações de duas semanas, chamadas de *Sprints*. Estes *Sprints*, são ciclos que têm uma duração fixa durantes os quais o produto construído é entregue para *feedback* (Sachdeva, 2016).

A primeira etapa é a criação de um *Product Backlog*. Este apresenta uma lista de requisitos bem definidos que devem ser implementados ao longo do processo de desenvolvimento, no formato de *User Stories*.

O *Sprint Backlog* é criado em seguida. As *User Stories* que deverão estar completas no fim de cada iteração são repartidas em várias *Tasks* que apresentam uma descrição do que é pretendido, uma prioridade e uma estimativa do esforço necessário para a completar.

Quando o processo de desenvolvimento de *software* é iniciado, são realizadas *Scrum Meetings* de dois em dois dias, em que o objetivo é saber o estado atual do projeto.

Por fim, cada *Sprint* deve terminar com uma breve demonstração dos resultados do trabalho produzido e execução de testes. Posteriormente, os resultados são discutidos e indicados os pontos a melhorar para prosseguir para o planeamento do próximo *Sprint*.

1.4 Estrutura do documento

Esta dissertação está organizada em sete capítulos: Introdução, Estado de Arte, Análise de Valor, Análise e Desenho, Desenvolvimento da Solução, Experimentação e Avaliação e, por último, Conclusão.

No primeiro capítulo, o de Introdução, é apresentado um enquadramento do projeto, o problema em mãos, incluindo os objetivos a alcançar, a abordagem a seguir para a sua resolução e, por fim, é apresentada a estrutura do documento.

No segundo capítulo, o do Estado da Arte, são introduzidos conceitos e modelos teóricos relacionados com o projeto, onde será feita uma análise das abordagens existentes ou similares, aplicadas a problemas idênticos ou relacionados.

No terceiro capítulo, Análise de valor, é estudado o valor deste problema, descrevendo o processo de inovação, a proposta de valor e o Quality *Function Deployment* (QFD).

No quarto capítulo, Análise e Desenho, tendo em conta a arquitetura base do Drive, são abordados os conceitos sobre o domínio do problema. A solução encontrada será apresentada com recurso a diagramas de notação UML (Linguagem de Modelação Unificada).

No quinto capítulo, o de Desenvolvimento da solução, será apresentado o processo de implementação da solução.

No sexto capítulo, o de Experimentação e Avaliação, são referidas as hipóteses de investigação, identificados os indicadores de avaliação a utilizar nas mesmas e descrita a metodologia de avaliação adotada.

Por último, o capítulo de Conclusão, são apresentadas as conclusões obtidas, os objetivos concretizados, as contribuições e as limitações encontradas. Além disso, é feita uma reflexão crítica ao projeto realizado e apresentados possíveis caminhos para melhorias no futuro.

2 Estado da Arte

O capítulo do Estado da Arte é composto por duas partes. Na primeira são analisados trabalhos relacionados com o projeto a desenvolver. Na segunda parte, são apresentadas as diferentes ferramentas e tecnologias atualmente utilizadas no produto, é também apresentado um estudo sobre as ferramentas essenciais ao desenvolvimento da solução e finaliza com uma análise crítica sobre as diferentes soluções.

2.1 Trabalhos relacionados

Atualmente, são vários os setores do mercado que utilizam aplicações informáticas para se manterem conectados e partilharem dados por todo o mundo, facilitando as tarefas do trabalhador. Nesta secção são analisadas várias soluções que, apesar de se inserirem em diferentes setores, possuem funcionalidades semelhantes às pretendias no contexto deste trabalho.

A seleção das soluções para análise teve por base um conjunto de características que de alguma forma se alinhavam com as pretendidas neste projeto. As características tidas em conta para análise e comparação das aplicações foram as seguintes:

- Registo de Eventos - espera-se que o elemento a considerar registe diversas ocorrências;
- Sistema de Notificações em Tempo Real - pretende-se um sistema que envie notificações dos eventos registados para diferentes dispositivos.

2.1.1 Sustainable Infrastructure Management System - SustIMS

Esta aplicação resultou de um projeto desenvolvido por um conjunto de entidades portuguesas no ano de 2012. O seu objetivo é gerir os principais elementos que incluem uma infraestrutura, nomeadamente obras-de-arte, pavimentos, taludes, muros e equipamentos de telemática (Projeto SustIMS, no date).

O sistema SustIMS integra uma Plataforma de Gestão, uma Plataforma Móvel e um Sistema de Monitorização Online.

A Plataforma de Gestão integra e gere toda a informação, desde o registo de inventário, inspeção das infraestruturas e monitorização online da infraestrutura.

A Plataforma Móvel, desenvolvida para tablets, apoia a atividade de monitorização da infraestrutura no terreno ('SISTEMA INTELIGENTE DE GESTÃO DE INFRAESTRUTURAS RODOVIÁRIAS', no date).

O processo de recolha de informação em campo é feito da seguinte forma (Ferreira et al., no date):

1. Plataforma de Gestão - Seleção das infraestruturas a inspecionar e agendamento de inspeção;
2. Notificação por email para o técnico de terreno;
3. Plataforma Móvel - Sincronização com a plataforma de gestão, download da inspeção a realizar;
4. Plataforma Móvel - Execução e *upload* da inspeção;
5. Plataforma de Gestão - Notificação de inspeção concluída.

A Monitorização online permite um controlo da infraestrutura em tempo real com recurso a sensores (Gestão da Manutenção | Ascendi, no date). Por exemplo, no caso de deslizamento de muros e taludes ou no embate de veículos na guarda de segurança, o sistema instalado permite enviar alertas em tempo real para as múltiplas plataformas (Backoffice[3], Telemóvel ou Painéis de mensagem) (SustIMS | Plataforma Ascendi para Gestão de Infraestruturas - YouTube, no date).

[3] Departamento de uma empresa que, geralmente, faz serviços na área operacional ou de gestão e na área administrativa.

2.1.2 Deighton Total Infrastructure Managent System - dTIMS

A aplicação dTIMS, desenvolvida pela empresa Norte Americana Deighton, permite a gestão, manutenção e controlo das infraestruturas, ativos de operação e equipamentos.

A plataforma dTIMS é composta por três componentes: (1) *Business Intelligence*, (2) *Operations Management* e (3) *Business Analytics*.

1. *Business Intelligence*, com a informação centralizada em *dashboards* personalizáveis, permite obter informações mais rápidas e acionáveis (dTIMS Business Intelligence (BI) — dTIMS® | Infrastructure Asset Management Software, no date);
2. *Operations Management*, é usado por gestores, administradores, supervisores e pessoal que se encontra em campo para acompanhar os pedidos, recursos e trabalho do dia a dia (dTIMS Operations Management (OM) — dTIMS® | Infrastructure Asset Management Software, no date).
3. *Business Analytics* constitui um apoio à decisão para ajudar a tomar decisões precisas, coerentes e informadas sobre o ciclo de vida dos ativos (dTIMS Business Analytics (BA) — dTIMS® | Infrastructure Asset Management Software, no date);

A dTIMS também possui uma app Mobile que fornece ao pessoal em campo a ligação em tempo real à gestão do trabalho, inspeções e recolha de dados (dTIMS Mobile — dTIMS® | Infrastructure Asset Management Software, no date).

2.1.3 Onfleet

A Onfleet fundada em 2015 na Califórnia apresenta como sua missão “permitir às empresas de todos os tamanhos transportar bens da forma mais eficiente” (Onfleet | Company, no date).

Esta aplicação é composta por duas principais plataformas: uma Web *Dashboard* e uma App Mobile.

A *Dashboard* permite ao gestor o controlo das tarefas de despacho e planeamento de rotas (Onfleet | Features, no date).

Os condutores que se encontram no terreno para as entregas, possuem uma aplicação móvel. Esta plataforma permite que os condutores estejam atualizados ao segundo sobre as informações do cliente, dos detalhes da entrega, da navegação que terão de fazer e, ainda,

permite comprovar a entrega com fotos anexadas, assinaturas e anotações (Onfleet | Features, no date).

Tanto os gestores como os condutores possuem um chat na sua plataforma, de modo a assegurar que os gestores estão em contacto com a sua equipa num ambiente seguro e privado (*Onfleet | Features*, no date). Quando os utilizadores recebem uma mensagem, são notificados com alertas na sua plataforma.

O cliente também possui uma plataforma. Esta plataforma permite o envio de notificações acerca do estado da encomenda e ainda rastrear e falar com o condutor em tempo real (Onfleet | Features, no date).

2.1.4 Análise comparativa

Após o estudo das diferentes aplicações e com requisitos semelhantes aos pretendidos neste projeto é possível fazer uma análise crítica.

SustIMS aparenta ser a aplicação mais próxima à área de negócio e com ambas as características inicialmente pretendidas.

O registo de eventos tanto é feito pela aplicação móvel, que regista o estado de preservação e manutenção dos elementos constituintes das autoestradas (*upload* das inspeções), como pela plataforma de gestão que, por exemplo, faz o registo do inventário.

O sistema de notificações em tempo real é assegurado quando o *upload* da inspeção é concluído e o gestor recebe um alerta na sua *Dashboard* informando a conclusão da operação.

O sistema dTIMS também se aplica à área em estudo. No entanto, apenas podemos afirmar que esta plataforma assegura o registo de eventos, nomeadamente, no *Operations Management* (gestão de pedidos) e na aplicação móvel que possibilita a recolha de dados em inspeções realizadas.

Apesar de existir uma plataforma móvel e Web, não foi possível confirmar a existência de um sistema que permita a comunicação destas duas plataformas em tempo real.

Por último, apesar da Onfleet não estar relacionada com o domínio deste projeto também se apresenta como uma solução com potencial.

Mais uma vez o registo de eventos é efetuado tanto por Mobile como Web, por exemplo e respetivamente, na conclusão de uma encomenda e na adição de um ponto de entrega na rota do condutor.

A grande vantagem da Onfleet é que possui um chat em tempo real entre os dois principais atores (gestor e condutor). Com isto, é possível assegurar um sistema de notificações em tempo real aquando do envio de mensagens, tanto do lado do condutor como do gestor.

Na Tabela 1 é apresentada uma síntese das diferentes aplicações analisadas, com base nas características previamente definidas.

Tabela 1 - Comparação de funcionalidades entre as diferentes aplicações

Aplicação	**Registo de Eventos**	**Sistema de Notificações em Tempo Real**
SustIMS	sim	Plataforma Móvel -> Plataforma de Gestão
dTIMS	sim	n/a[4]
Onfleet	sim	Plataforma Móvel -> Plataforma de Gestão Plataforma de Gestão -> Plataforma Móvel
Drive	sim	Drive Web -> Drive Web

2.2 Tecnologias existentes

Este projeto consiste na adição de novas funcionalidades ao produto Drive. Posto isto, é importante um estudo sobre o contexto tecnológico do produto o qual, de alguma forma, condicionou o desenvolvimento deste projeto.

2.2.1 Contexto Tecnológico

Nesta secção inicia-se uma análise às tecnologias atualmente utilizadas pelo Drive Web e pelo Drive Mobile. A primeira solução é desenvolvida maioritariamente em JavaScript, mais

[4] Não se aplica.

especificamente com a utilização da plataforma ASP.NET MVC, sendo que o Drive Mobile desenvolvido com recurso à tecnologia Xamarin comunica com uma Web API desenvolvida em ASP.NET.

No que diz respeito à base de dados, que apoia o funcionamento do sistema Drive, esta encontra-se implementada em Microsoft SQL Server (MSSQL).

Na Figura 1 é apresentado um esquema do sistema Drive.

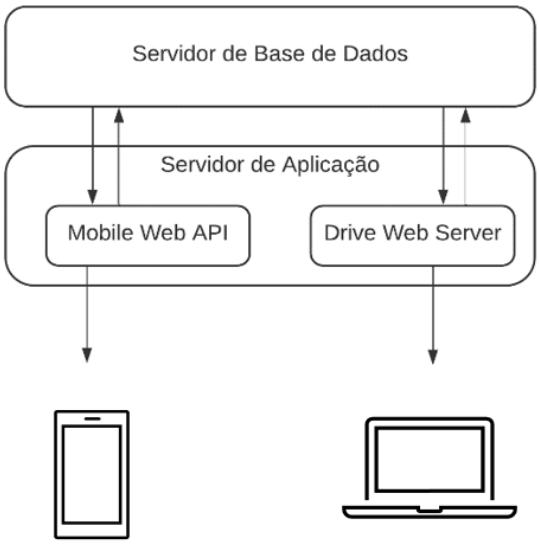


Figura 1 - Esquema arquitetural do Drive

ASP.Net Web API

A *Aplication Programming Interface* (API) construída sobre .Net Framework, ASP.Net Web API, é uma ferramenta que auxilia a construção de serviços Representational State Transfer (REST) em .Net e C#. Estes serviços fornecem informações necessárias tanto a *browsers* como dispositivos móveis (ASP.NET Web APIs | Rest APIs with .NET and C#, no date).

Xamarin

Esta plataforma, desenvolvida pela Microsoft, permite a construção de aplicações nativas em Android e em iOS, sendo que o mesmo *software* é utilizado em várias plataformas (Xamarin | Open-source mobile app platform for .NET, no date).

ASP.Net MVC

A plataforma Asp.Net MVC, apoiada no padrão arquitetural *Model-View-Controller* (MVC), é uma *framework* simples que permite para a construção de sites dinâmicos.

MVC assegura a separação clara de preocupações, de modo a diminuir o acoplamento e dividir as responsabilidades em três grandes grupos de componentes (Overview of ASP.NET Core MVC | Microsoft Learn, no date):

- Dados (Model) representa os dados e lógica de negócio;
- *User Interface* (View) é responsável por apresentar os dados do componente *Model*;
- Lógica da aplicação (Controller) são os componentes que respondem às solicitações feitas no site.

Asp.Net MVC *View* usa a *View* Engine Razor para gerar dinamicamente o conteúdo da aplicação *Web* no servidor.

A View Engine Razor é uma sintaxe que ajuda os programadores a definir as *Views*. Nesta estrutura, as páginas com a extensão. cshtml misturam *tags* HyperText Markup Language (HTML) com scripts escritos em C# (Razor syntax reference for ASP.NET Core | Microsoft Learn, no date).

Cada uma das solicitações ao site é mapeada para um *Controller*, que por sua vez escolhe a *View* ou as *Views* que quer usar. A *View* gera a página final, baseada nos dados fornecidos pelo *Model* (ver Figura 2).

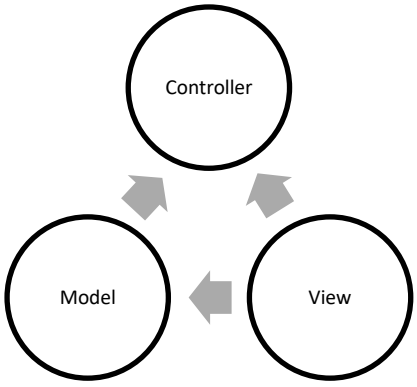


Figura 2- Padrão Arquitetural MVC (ASP.NET MVC | Microsoft Learn, no date)

Microsoft SQL Server

O Microsoft SQL Server, também desenvolvido pela Microsoft, é um sistema de gestão de bases de dados relacional. A sua principal função é persistir e fornecer os dados solicitados por outras aplicações de *software*. Os dados são modelados em tabelas (linhas e colunas).

2.2.2 Sistema de Notificações existente

Antes de iniciar este projeto, o Drive tinha um simples sistema de notificações que utilizava uma única fila para organizar as mensagens. Este sistema foi desenvolvido utilizando RabbitMQ, que é um *software* de *Message-Broker*.

A arquitetura deste sistema consiste na existência de produtores que enviam uma ou mais mensagens para um processo do servidor. Neste caso, o Drive Web atuava tanto como produtor (responsável por enviar mensagens) quanto como consumidor (aguardando a receção de mensagens).

Um *Message-Broker* é uma peça de *software* que gere a entrega de mensagens entre diferentes partes de um sistema distribuído, usando protocolos padrão para o empacotamento, roteamento e entrega de dados para os destinatários apropriados (consumidores). Estes protocolos definem como as mensagens são formatadas, transmitidas, processadas e consumidas.

Atualmente, o Drive utiliza o protocolo *Advanced Message Queuing Protocol* (AMQP). Na Figura 3 é apresentado um esquema de como o RabbitMQ se comporta com a utilização do Protocolo AMQP no Drive.

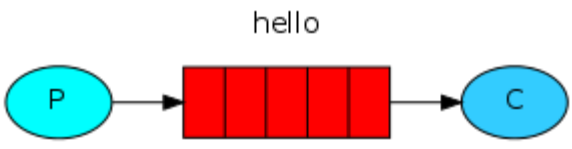


Figura 3 - Protocolo AMQP aplicado no Drive (*RabbitMQ tutorial - 'Hello world!' — RabbitMQ*, no date)

Na arquitetura, o Produtor (P) é responsável por enviar eventos e o Consumidor (C) espera para receber as mensagens enviadas pelo Produtor. A Fila serve de armazenamento de mensagens.

Vários Produtores podem enviar mensagens para a mesma Fila, e muitos Consumidores podem tentar retirar a informação da mesma fila. No entanto, apenas uma conseguirá obter a mensagem, isto porque as mensagens AMQP são *load balance* entre os Consumidores (AMQP 0-9-1 Model Explained — RabbitMQ, no date).

Para o objetivo deste projeto, em que o mesmo evento pode chegar a dois Consumidores (Drive Web e Drive Mobile) a implementação atual não é adequada, uma vez que com esta implementação de AMQP apenas um dos consumidores iria receber a mensagem.

Assim sendo, é necessária uma solução que atenda à capacidade de direcionar mensagens para um único consumidor específico, mas também a capacidade de as entregar a ambos os consumidores quando a situação exigir.

2.2.3 Protocolos de Mensagens

Existem vários protocolos de comunicação que são usados frequentemente na implementação de aplicações, tais como (1) *HyperText Transfer protocol* (HTTP), (2) Message Queuing Telemetry Transport (MQTT), (3) *Advanced Message Queuing Protocol* (AMQP), entre outros. Nesta secção é apresentado um estudo e comparação entre os diversos protocolos de mensagem.

1. *HyperText Transfer Protocol* (HTTP)

Este protocolo de mensagens, o mais usado na Web, foi introduzido em 1977. HTTP é um protocolo do tipo *request/response* em que, um cliente, através do *Universal Resource Identifier* (URI) que identifica a comunicação, solicita uma mensagem e o servidor gera a mensagem de resposta (Gemirter, Çenturca and Baydere, 2021).

O transporte padrão utilizado é *Transmission Control Protocol* (TCP) e *Transport Layer Security/Secure Sockets Layer* (TLS/SSL) para segurança (Naik, 2017). A *Quality of Service* (QoS) não é fornecida neste protocolo.

A Figura 4 representa as diferentes formas em que o protocolo HTTP pode ser utilizado.

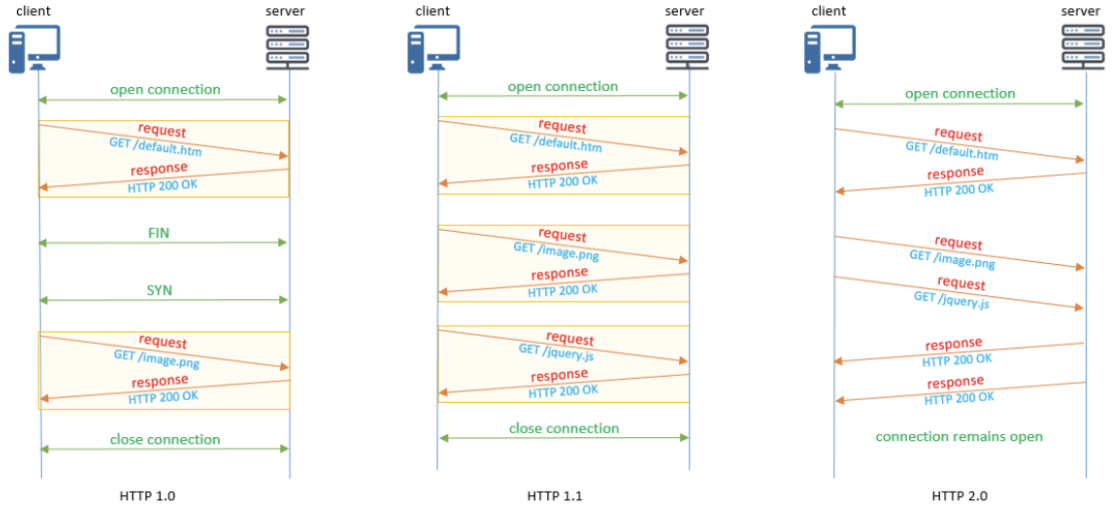


Figura 4 - Protocolo HTTP (Al-Masri et al., 2020)

2. *Message Queuing Telemetry Transport* (MQTT)

É um dos mais antigos protocolos de comunicação para *Machine-to-Machine* (M2M), introduzido em 1999. Este é um protocolo do tipo *publish/subcribe* em que à semelhança de HTTP corre sobre TCP e TLS/SSL para segurança (Naik, 2017).

MQTT é considerado um protocolo mais leve que HTTP uma vez que lida com situações que HTTP não consegue, como conectividade intermitente e troca de mensagens em tempo real usando o modelo *publish/subcribe* (Al-Masri et al., 2020).

O protocolo MQTT é usado em várias aplicações onde a troca de dados é necessária, como monitorização de tráfego, automação de casas e telemática automotiva (Al-Masri et al., 2020).

Este modelo é composto por um Broker e por clientes. A qualquer momento, os clientes podem estabelecer uma ligação com o Broker, identificando o tópico/mensagem no qual estão interessados. Quando um cliente envia uma mensagem para o Broker, este filtra as mensagens recebidas e distribui apenas para os clientes que estão interessados em receber o tópico enviado.

A Figura 5 representa a forma em que o protocolo MQTT pode ser utilizado.

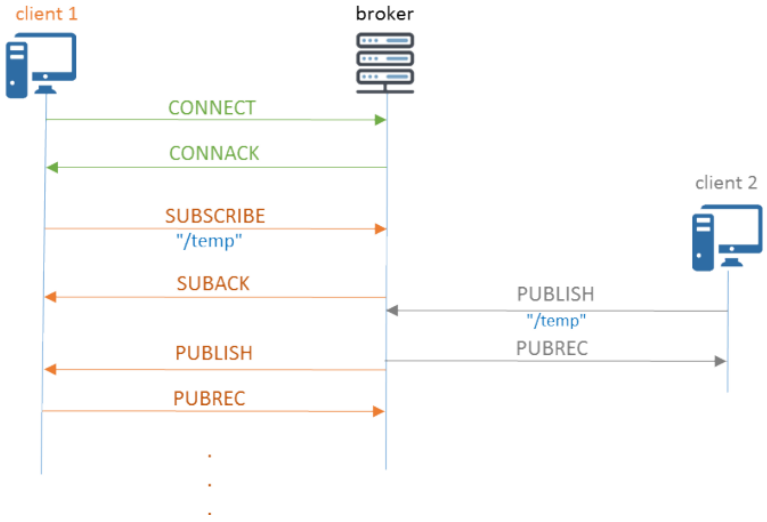


Figura 5 - Protocolo MQTT (Al-Masri et al., 2020)

Este protocolo possui ainda a vantagem de ter três níveis de QoS que podem ser configurados de acordo com as necessidades do projeto, todos estes são relacionados com a garantia da entrega das mensagens (Al-Masri et al., 2020; Shahri, Pedreiras and Almeida, 2021):

Como pode ser observado na Tabela 2, à medida que o nível de QoS a confiabilidade na entrega de mensagens também aumenta.

Tabela 2 – Níveis de QoS do protocolo MQTT

Nível de *QoS*	**Nível de Confiabilidade**	**Descrição**
QoS 0	baixo	No máximo uma vez. Não possui qualquer tipo de garantia na entrega.
QoS 1	médio	Pelo menos uma vez. Entrega da mensagem é garantida.
QoS 2	alto	Apenas uma vez. Entrega da mensagem é garantida. Elimina eventuais duplicados.

Existem várias implementações do modelo MQTT, tais como Mosquitto, eMQTT, HiveMQ, Mosquette, etc (Al-Masri et al., 2020).

3. *Advanced Message Queuing Protocol* (AMQP)

AMQP é outro protocolo de *Machine-to-Machine* (M2M), introduzido em 2003. Este protocolo suporta ambas as arquiteturas anteriormente referidas, a *request/response* e a *publish/subcribe.* Este protocolo usa TCP e TLS/SSL e SASL para segurança (Naik, 2017).

O protocolo AMQP é composto por três tipos de componentes: (1) *queues, (2) bindings e (3) exchanges.* Para além disto, possui diferentes formas na troca e entrega de mensagens.

1. Uma *queue* de mensagens é um *buffer* do tipo *First-In-First-Out* (FIFO) que armazena as mensagens temporariam*ente.* Uma fila apenas pode ser usada por um consumidor (Uy and Nam, 2019).
2. Os *bindings* representam a relação entre os *exchanges* e as *queues*. Ou seja, dependente do tipo de *Exchange*, a mensagem é reencaminhada para a *queue* apropriada.
3. Um *exchange* tem a responsabilidade de aceitar as mensagens e as reencaminhar para as filas de mensagens apropriadas. Existem vários tipos de *exchanges:*
    - *Direct*: o *exchange* utiliza a *routing key* associada a cada mensagem para enviar a mensagem as *queues*. A mensagem é entregue à *queue* que possuir um binding key exatamente igual à *routing key* enviada pelo produtor;
    - *Topic*: as mensagens são distribuídas para uma ou mais *queues* com base num *routing pattern*. A mensagem é entregue à *queue* se a *routing key* corresponder ao *routing pattern*;
    - *Fanout*: a mensagem é duplicada e entregue a todas as *queues* associadas, sem a necessidade de *routing key;*
    - *Header:* o *exchange* envia a mensagem para a *queue* com base em argumentos e propriedades especificadas no cabeçalho da mensagem.

A Figura 6 ilustra as várias formas em que o protocolo AMQP pode ser utilizado.

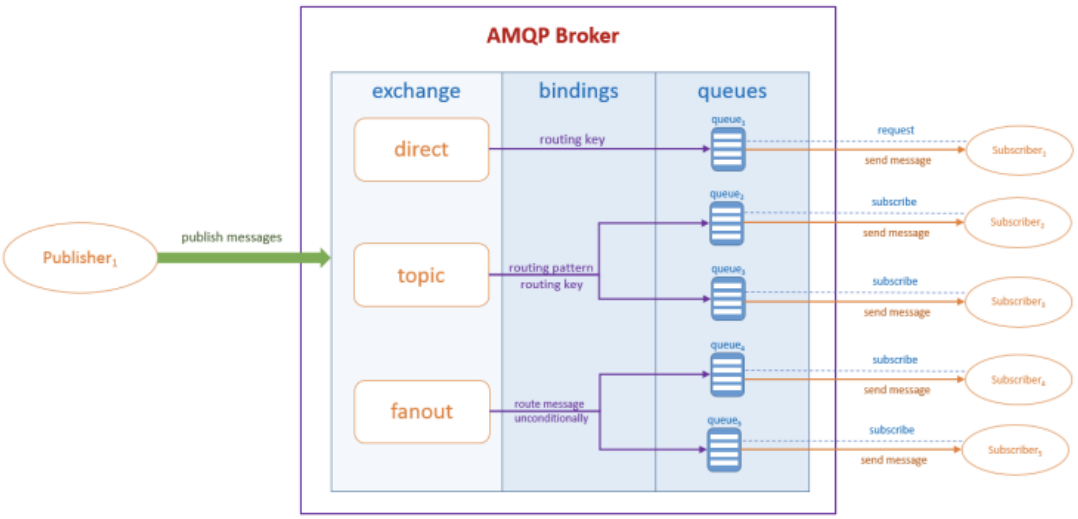


Figura 6 - Protocolo AMQP (Al-Masri et al., 2020)

AMQP também lida com o controlo da entrega de mensagens: no máximo uma vez (QoS 0), pelo menos uma vez (QoS 1) e apenas uma vez (QoS 0) (Al-Masri *et al.*, 2020). Relacionado com o QoS, o AMQP usa termos como "unsettle" e "*settle*" que lhe permitem garantir a entrega e processamento total das mensagens quando o consumidor envia o *acknowledge(ack)*[5] (Aiyagari et al., 2008; Al-Masri et al., 2020).

Como já anteriormente mencionado, o RabbitMQ é uma das tecnologias que implementa o protocolo AMQP.

2.2.4 Análise comparativa

Na Tabela 3 é apresentada uma síntese comparativa dos diferentes protocolos de mensagem estudados.

[5]Aceita e confirma a entrega da mensagem.

Tabela 3 – Análise comparativa dos protocolos de mensagem

Funcionalidade\Protocolo	**HTTP**	**MQTT**	**AMQP**
Ano	1991	1999	2003
Padrão de Mensagem	resquest/response	publish/subscribe	resquest/response; publish/subscribe
QoS	não	sim	sim
Transporte	TCP	TCP	TCP
Segurança	HTTPS	TLS	TLS/SASL
Mensagens assíncronas	não	sim	sim
Filtro de mensagens	URI	Tópico	Fila

Todos os protocolos de mensagem estudados (HTTP, MQTT e AMQP) servem para a comunicação de dados. No entanto, é necessário escolher aquele que melhor se adequa ao problema em mãos.

O protocolo HTTP não apresenta as características necessárias para a aplicação num sistema de eventos em tempo real, uma vez que apresenta uma grande latência no transporte de dados.

Várias experiências e estudos foram feitos comparando o desempenho entre MQTT e AMQP. A experiência e o estudo em análise, realizado em 2019, compara estes dois protocolos para a sua aplicação na *Internet of Things* (IoT) (Uy and Nam, 2019).

A experiência analisada utilizou Wifi para a transmissão de dados, a tecnologia Mosquitto para MQTT e RabbitMQ para AMQP. Para que os protocolos de mensagem em análise sejam comparados nas mesmas características, utilizou-se o *Exchange Fanout* que é o mais parecido ao protocolo MQTT (Uy and Nam, 2019).

Para esta experiência foi utilizado um *software* que simula um ambiente de transmissão com taxa de perda e atraso na rede, de forma a parecer um verdadeiro sistema de IoT.

Foram realizados dois tipos de testes: (1) Mensagens enviadas de forma contínua e (2) Mensagens enviadas de cinco em cinco segundos.

1. Mensagens enviadas de forma contínua (Uy and Nam, 2019):
    - O AMQP apresenta-se como melhor opção quando a taxa de perda é alta. No entanto, quando a taxa de perda é menor o MQTT- QoS 1 fornece melhores resultados;
    - Num ambiente com alta latência o protocolo que apresenta melhor comportamento é o AMQP.
2. Mensagens enviadas de cinco em cinco segundos (Uy and Nam, 2019):
    - Em ambientes sem taxa de perda, tanto o AMQP como o MQTT são bons e estáveis;
    - Em ambientes com taxa de perda, à medida que a perda aumenta, o MQTT- QoS 1 apresenta um tempo de atraso mais baixo e o MQTT- QoS 2 e o AMQP apresentam um tempo de transporte mais elevado.

Concluindo, devido à simplicidade que o MQTT traz comparativamente ao AMQP, que necessita de vários recursos em dispositivos com recursos limitados, o MQTT apresenta uma significativa melhoria em termos de eficiência. Além disso, o protocolo MQTT é ideal para o envio de mensagens que não são enviadas continuamente (Uy and Nam, 2019). Por outro lado, o AMQP é um protocolo mais fiável e com melhor segurança. É recomendado para situações em que os pacotes de mensagens precisam de ser enviados continuamente (Uy and Nam, 2019).

3 Análise de Valor

A Análise de Valor apresentada neste capítulo, tem como principal objetivo o aumento do valor do produto para o cliente ao menor custo e sem o sacrificar. O processo de Análise de Valor começa na identificação do processo de inovação, seguido da Proposta de Valor. Por fim é apresentada *Quality Function Deployment* (QFD).

3.1 Processo de inovação

Com o rápido crescimento da tecnologia e com o surgimento de cada vez mais produtos e serviços no mercado, as empresas, para assegurarem a sua sobrevivência a longo prazo, necessitam de certificar a qualidade dos produtos que oferecem aos seus clientes, não esquecendo de satisfazer as suas necessidades. Para alcançar essa qualidade, as organizações necessitam de definir um processo de inovação formal e estruturado, podendo ser dividido em três passos: (1) Fuzzy Front End (FFE), (2) New Product Development (NPD) e (3) Commercialization.

A primeira etapa é considerada uma das mais importantes no processo, isto porque é onde as oportunidades são identificadas e os conceitos são desenvolvidos antes de entrar no desenvolvimento do produto. No entanto, devido à falta de linguagens e vocabulário comum entre as empresas, o FFE pode tornar-se complicado e por isso o *New Concept Development Model* (NCD) destina-se a fornecer uma visão e terminologia comum para o FFE (*The PDMA ToolBook 1 for New Product Development - Google Livros*, no date).

O modelo NCD, ilustrado na Figura 7, é constituído por três componentes (*The PDMA ToolBook 1 for New Product Development - Google Livros*, no date):

- Motor (*Engine*): consiste na gestão do nível superior e executivo, que por sua vez influencia os cinco elementos-chave do modelo;
- Elementos-chave: atividades controláveis influenciadas pelo motor, inclui a identificação e análise de oportunidade, geração e enriquecimento de ideias, seleção de ideias e definição de conceito;
- Fatores influenciadores: fatores externos, geralmente incontroláveis pela organização, que afetam o processo de inovação, nomeadamente políticas governamentais, clima político e económico, etc.

As setas de entrada indicam que o processo deve começar na geração e enriquecimento de ideias ou então na identificação de oportunidade. A seta de saída assinala os conceitos que devem sair do modelo e entrar no *New Product Development*.

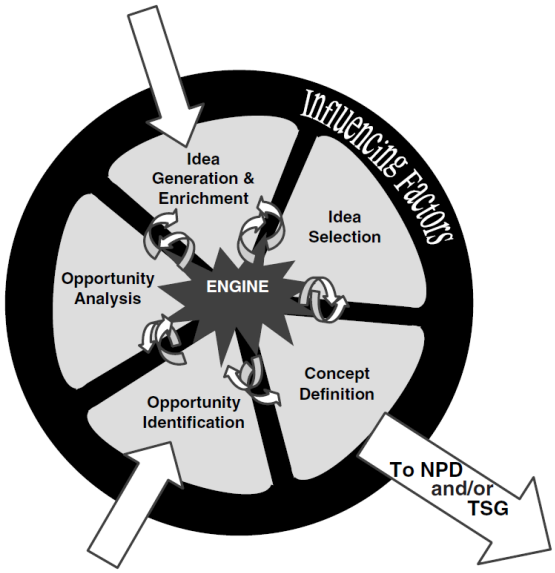


Figura 7 - Modelo NCD (*The PDMA ToolBook 1 for New Product Development - Google Livros*, no date)

Nas secções seguintes serão detalhadas as várias atividades constituintes do modelo NCD.

3.1.1 Identificação de Oportunidade

A identificação de oportunidade consiste na identificação de oportunidades empresariais e tecnológicas que uma organização quer seguir. Estas oportunidades identificadas, geralmente

levam a que os recursos sejam afetos a novas áreas de mercado, com eficácia e eficiência operacional (*The PDMA ToolBook 1 for New Product Development - Google Livros*, no date).

A oportunidade pode ser uma nova plataforma de produtos, um novo processo de serviços, uma nova abordagem de marketing ou vendas, etc (*The PDMA ToolBook 1 for New Product Development - Google Livros*, no date). Neste projeto a oportunidade identificada surge com a adição de nova funcionalidade a um serviço existente.

Como mencionado na Descrição do Problema (secção 1.2), cada vez mais, o uso de dispositivos móveis é considerado uma parte integrante do dia-a-dia de muitos trabalhadores, permitindo que as pessoas se mantenham em contacto uma com as outras. A pandemia COVID-19 veio a potenciar o uso das tecnologias para fomentar as relações sociais e de bem-estar em geral (Jonnatan *et al.*, 2022).

Com este crescimento tecnológico, neste último ano foram adicionadas várias funcionalidades à aplicação móvel do Drive. Algumas destas funcionalidades, antes comunicadas por rádio aos gestores, necessitam de ações por parte do gestor, nomeadamente a sinalização de painéis na autoestrada.

Posto isto, foi identificada a oportunidade de criar um sistema de notificações que permita os gestores e os trabalhadores em terreno estarem conectados entre si em tempo real.

Nos Trabalhos relacionados (secção 2.1), apesar de nem todas as aplicações referidas se enquadrarem na área de negócio, são mencionadas algumas plataformas que incluem um sistema de notificações em tempo real no seu sistema.

3.1.2 Análise da Oportunidade

Nesta etapa, a oportunidade anteriormente identificada é avaliada para confirmar o seu desenvolvimento.

A oportunidade identificada surge no mercado de ITS (*Intelligent Transport Systems*). Estes sistemas visam melhorar os serviços de transporte com tecnologias inovadoras que podem substituir a supervisão humana.

Em 2022 este segmento do mercado foi avaliado em vinte e oito mil milhões de dólares. Com a crescente procura de sistemas de controlo de tráfego, melhoria da segurança e vigilância e com

o rápido desenvolvimento de cidades inteligentes, prevê-se um aumento da taxa de crescimento anual entre 2023 e 2030 (*Intelligent Transportation System Market Size Report, 2030*, no date).

Estes sistemas reforçam a fiabilidade e o desempenho operacional da rede rodoviária. É do interesse das organizações governamentais a procura de novas tecnologias e sistemas de transportes inteligentes que reduzem encargos financeiros para os governos.

Na Figura 8 são identificadas as várias quotas de mercado dos diferentes segmentos na ITS. O segmento que apresenta maior quota de mercado, com mais de 32%, é o de gestão de tráfego, o mesmo que se enquadra este projeto.

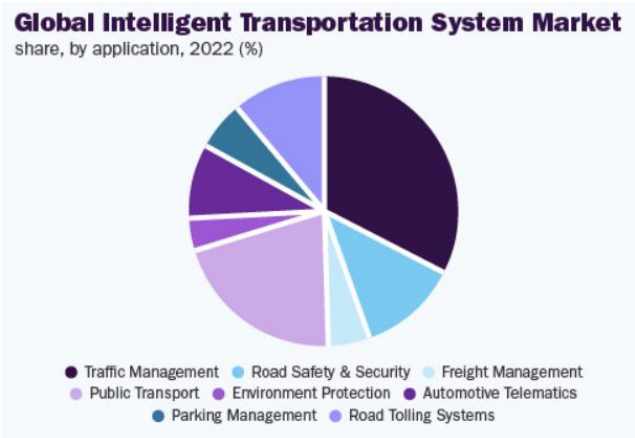


Figura 8 - Quota do mercado nos segmentos de ITS (*Intelligent Transportation System Market Size Report, 2030*, no date)

3.1.3 Geração e enriquecimento de ideias

Este elemento-chave refere-se ao processo pelo qual a ideia é concebida, desenvolvida e aprimorada resultando em uma ideia concreta. O processo da geração e enriquecimento de ideias é um processo contínuo e iterativo que muitas vezes envolve a colaboração com outras equipas fora da empresa, bem como a interação direta com os clientes e utilizadores finais (*The PDMA ToolBook 1 for New Product Development - Google Livros*, no date).

Este processo pode ajudar na identificação de oportunidades, que comprova que o modelo NCD não segue uma trajetória linear e previsível. À medida que as ideias são geradas e enriquecidas novas oportunidades podem surgir, levando a mudanças no modelo original (*The PDMA ToolBook 1 for New Product Development - Google Livros*, no date).

Como tal, durante esta fase foram analisados vários trabalhos e soluções que implementam um sistema de notificações em tempo real entre diferentes dispositivos. O objetivo desta análise foi recolher informações valiosas sobre como diferentes aplicações criam sistemas de notificações e como essas notificações são entregues.

Com esta informação, foram realizadas sessões de *brainstorming* com a equipa de desenvolvimento no qual este projeto se insere, surgindo algumas ideias como:

- Adaptar o sistema de notificações existente para atender as necessidades e preferências especificas do cliente, usando as mesmas tecnologias;
- Criar um sistema de notificações que atende as necessidades e preferências especificas do cliente, usando tecnologias que melhor se adequam a estas necessidades.

3.1.4 Seleção de ideais

Segundo Koen, "o problema da maioria dos negócios é a seleção de quais as ideias a seguir para alcançar o maior valor de negócio". Por isso, podemos dizer que a seleção de opções corretas é fundamental para a saúde e sucesso futuro de qualquer organização (*The PDMA ToolBook 1 for New Product Development - Google Livros*, no date).

Depois de analisar com a equipa de desenvolvimento as ideias anteriormente apresentadas, a escolha decorreu com a criação de um sistema de notificações. Esta decisão deve-se ao fato de que a criação de um novo sistema do zero permite mais flexibilidade e controlo no desenvolvimento atual e futuro das notificações pretendidas.

3.2 Proposta de Valor

A Proposta de Valor é definida como "uma visão geral de produtos e serviços de uma empresa que são de valor para o cliente" (*THE BUSINESS MODEL ONTOLOGY A PROPOSITION IN A DESIGN SCIENCE APPROACH*, 2004) .

O produto descrito na presente dissertação pretende beneficiar a indústria responsável pela gestão de infraestruturas e tráfego. O sistema de notificações desenhado oferece uma maneira eficiente e eficaz de gestão, fornecendo informações relevantes, precisas e oportunas em tempo real.

Com a implementação deste produto, existem alguns benefícios e sacrifícios que o cliente pode esperar, resultando no valor percebido.

Os benefícios incluem a melhoria da eficácia e eficiência na gestão de infraestruturas e tráfego, permitindo uma melhor tomada de decisões em tempo real, que pode levar a uma redução no tempo de inatividade e a uma melhor comunicação entre as equipas. Além disto, o cliente que solicita este desenvolvimento tem confiança na equipa de desenvolvedores encarregue de projetá-lo, este é um elemento muito importante pois permite que o cliente tenha a segurança de que o produto será entregue de acordo com as suas espectativas e requisitos.

Por outro lado, alguns sacrifícios incluem o custo, as mudanças nos processos de trabalho para se adaptarem ao novo sistema e treinar as equipas para o uso adequado do sistema. O tempo o esforço e a energia também necessitam de ser considerados.

Na Figura 9 é apresentado o Modelo Canvas da Proposta de Valor deste produto. Este é dividido em dois blocos principais: o Mapa de Valor (à esquerda) e o Perfil do Cliente (à direita).

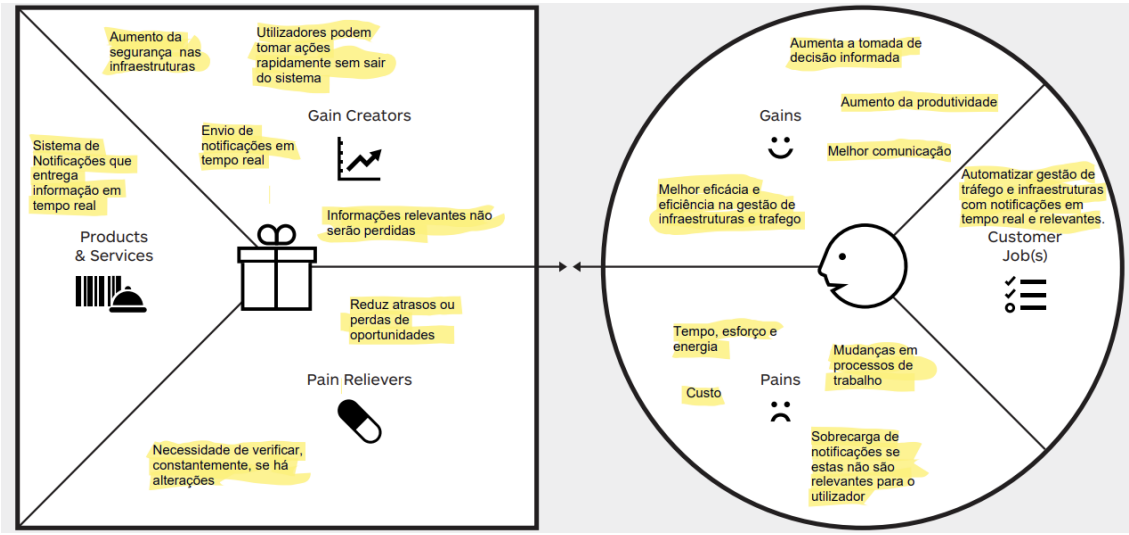


Figura 9 - Modelo CANVAS Proposta de Valor

3.3 Quality Function Deployment

A *Quality Function Deployment* (QFD) é uma técnica de desenho de um produto ou serviço que garante as necessidades dos clientes, envolvendo todas as partes interessadas, desde os produtores até aos fornecedores. QFD, do português Implementação da Função de Qualidade,

significa que as responsabilidades de produção de um produto com qualidade devem ser da responsabilidade de todos os elementos da organização. No contexto de QFD, cada um dos termos pode ser definido da seguinte forma (Warwick, no date):

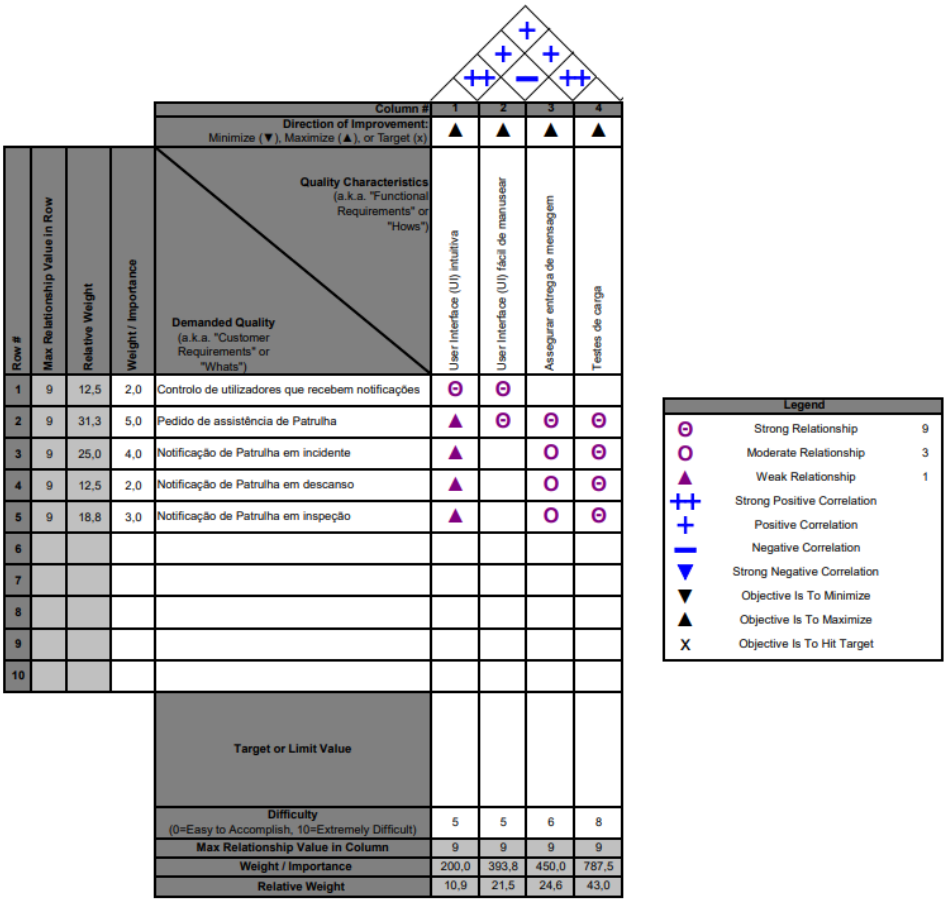
- Implementação (Deployment) – quem e quando o fará;
- Função (Function) – focar a atenção no que deve ser feito;
- Qualidade (*Quality*) – satisfazer as necessidades do cliente.

O objetivo do QFD é fabricar produtos de alta qualidade para serem comercializados mais rápido e a um custo menor. Para além disto, este sistema defende que ao conseguir desenhar um produto orientado às necessidades do cliente também será possível rastrear as futuras melhorias do produto ou do processo de desenvolvimento (Warwick, no date).

Estes resultados são obtidos, dividindo os requisitos do cliente em segmentos. O processo QFD é feito nas seguintes 5 fases (Warwick, no date):

- Requisitos do cliente;
- Desenho dos Requisitos;
- Características dos componentes;
- Requisitos de operações;
- Procedimentos de trabalho.

Na Figura 10 encontra-se apresentado a House of *Quality* (HOQ), que apresenta os requisitos do cliente e os traduzem em características de engenharia.



Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	1 User Interface (UI) intuitiva	2 User Interface (UI) fácil de manusear	3 Assegurar entrega de mensagem	4 Testes de carga
				Column #	1	2	3	4
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	▲	▲	▲	▲
1	9	12,5	2,0	Controlo de utilizadores que recebem notificações	⊙	⊙		
2	9	31,3	5,0	Pedido de assistência de Patrulha	▲	⊙	⊙	⊙
3	9	25,0	4,0	Notificação de Patrulha em incidente	▲		O	⊙
4	9	12,5	2,0	Notificação de Patrulha em descanso	▲		O	⊙
5	9	18,8	3,0	Notificação de Patrulha em inspeção	▲		O	⊙
6								
7								
8								
9								
10								
				Target or Limit Value				
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	5	5	6	8
				Max Relationship Value in Column	9	9	9	9
				Weight / Importance	200,0	393,8	450,0	787,5
				Relative Weight	10,9	21,5	24,6	43,0

Legend		
⊙	Strong Relationship	9
O	Moderate Relationship	3
▲	Weak Relationship	1
++	Strong Positive Correlation	
+	Positive Correlation	
−	Negative Correlation	
▼	Strong Negative Correlation	
▼	Objective Is To Minimize	
▲	Objective Is To Maximize	
X	Objective Is To Hit Target	

Figura 10 - *House of Quality*

No campo *Customer Requirements* encontram-se listados todos os requisitos que o cliente acha importante para o produto, sendo que cada um destes requisitos tem um grau de importância (*Importance*), 5 será o requisito que apresenta maior importância e 1 o que apresenta menor.

No campo *Quality Characteristics* são apresentados os atributos técnicos do produto, ou seja, como o produto pode atender as necessidades do cliente.

Entre os *Customer Requirements* e os *Quality Characteristics* são apresentados os símbolos que mostram a força entre esta relação. O campo *Importance* soma o valor atribuído à coluna dos *Function Requirements* para concluir acerca da sua importância.

Analisado a HOQ podemos evidenciar que o requisito funcional que apresenta maior grau de importância, com um peso relativo de 43%, é “Testes de carga”, seguido de o requisito “Assegurar entrega de mensagem” com um peso relativo de 24.6%.

4 Análise e Desenho da solução

Neste capítulo é exposto o processo de compreensão do negócio, iniciando na análise onde são apresentados e descritos o domínio do problema. De seguida é apresentada a Engenharia de Requisitos, e por último, numa vertente mais técnica, é apresentado o *Design* da solução proposta que compreende a arquitetura da solução.

4.1 Domínio do Problema

O Drive é um sistema que integra diversos módulos que permitem o seu normal funcionamento. Os diversos módulos formam uma divisão lógica de serviços e regras de negócio. Posto isto, e de forma a simplificar a compreensão do domínio da solução, optou-se por não realizar uma análise e desenho global do sistema como um todo. Neste capítulo, apenas são analisados os módulos relevantes para o âmbito deste projeto.

Na Tabela 4 são descritos os conceitos do domínio do problema, isto é, apresentada a definição de cada entidade.

Tabela 4 - Conceitos do domínio do problema

Entidade	Descrição
AssistanceMeans	Veículo utilizado no patrulhamento.
Concession	Conjunto de estradas.
Drivers	Condutor da Patrulha.
Element	Elementos constituintes de uma autoestrada.
ElementType	Tipo do Elemento. (Exemplo: Área de serviço)
GeographicArea	Área geográfica.
GroupOfLocation	Conjunto de características que compõem uma localização.
Kilometerpoint	Ponto do Quilómetro da autoestrada.
Orientation	Sentido do movimento.
Path	Caminho que a Patrulha deve percorrer.
Patrol	Patrulhamento de uma autoestrada.
PatrolType	Tipo da Patrulha.
Road	Estrada.
RoadSection	Secção pertencente a uma Estrada.
Situation	Acontecimento/Ocorrência numa autoestrada.
User	Utilizador do sistema.

Na Figura 11 é apresentado o Modelo de Domínio. De forma a melhorar a compreensão deste, posteriormente será feita uma breve explicação da forma como as entidades se relacionam entre si.

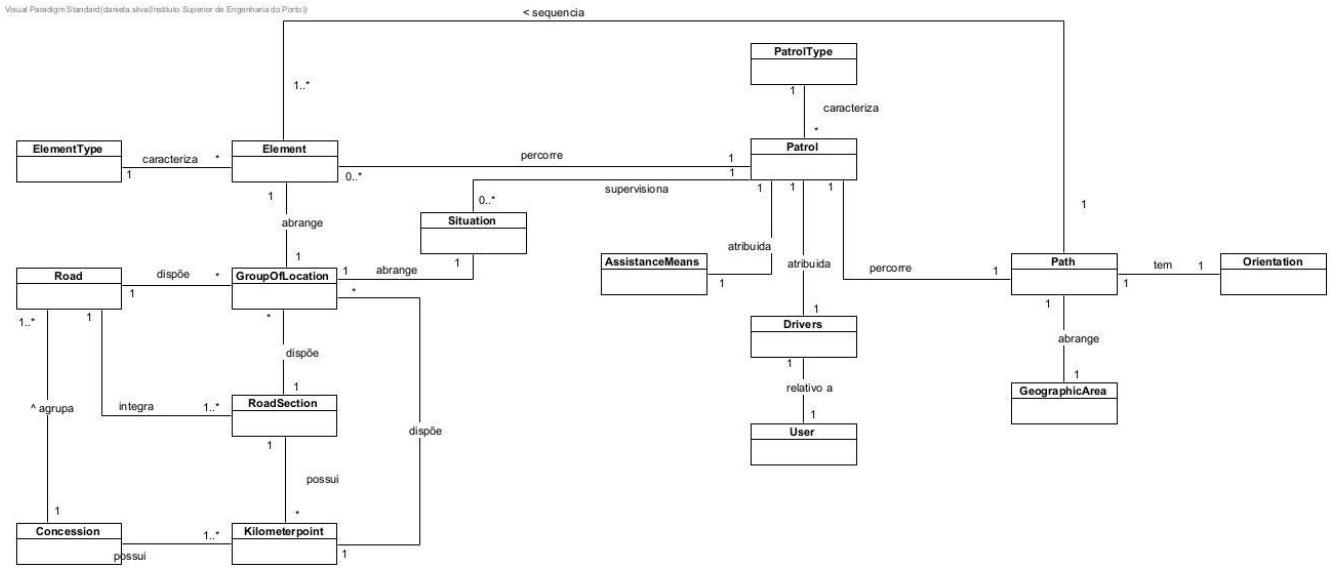


Figura 11 - Modelo de Domínio

O conceito de Patrulha (*Patrol*) diz respeito aos veículos que realizam o patrulhamento das infraestruturas e cada uma destas está associada a um tipo que a caracteriza (*PatrolType*).

Na criação de uma Patrulha é obrigatório indicar o seu condutor (*Driver*), o veículo utilizado (*AssistanceMeans*) e o trajeto pretendido percorrer (*Path*).

Os condutores também representam um utilizador do Drive (*User*), que por sua vez estão responsáveis por uma determinada área geográfica (*GeographicArea*).

Depois de se iniciar uma Patrulha, podem ocorrer os seguintes acontecimentos durante o seu percurso:

- Paragem em incidente – Patrulha fica associada a essa ocorrência (*Situation*) na autoestrada;
- Paragem para inspeção ou descanso – patrulha fica associada a um Elemento (*Element*) que constitui a autoestrada.

4.2 Engenharia de Requisitos

A presente secção tem como objetivo detalhar o levantamento dos requisitos, fruto das necessidades identificadas pelo cliente. Inicialmente são explicados os requisitos funcionais e, posteriormente, as especificações suplementares.

4.2.1 Casos de Uso

Como já mencionado na Abordagem (secção 1.3), os requisitos foram descritos numa visão mais geral, designados de *User Stories*, cada uma delas é atribuída a um ator, que representa o utilizador que interage com o sistema.

Na fase de levantamento foram identificados 6 Casos de Uso (UC) e 2 atores. A Figura 12 apresenta o diagrama de Casos de Uso resultante do levantamento dos requisitos.

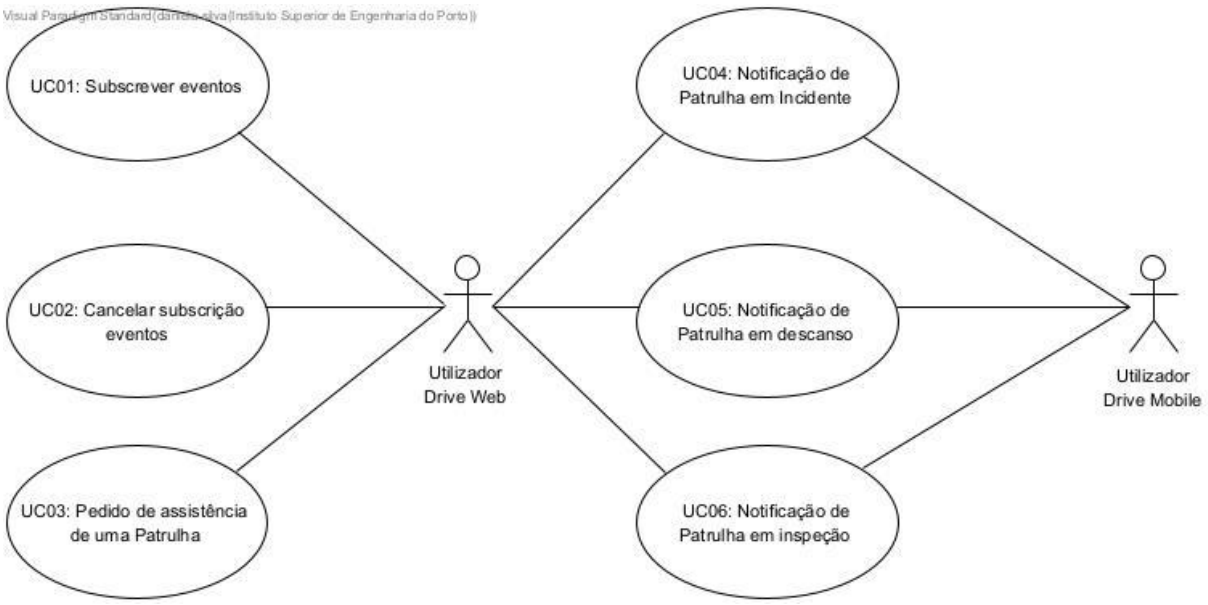


Figura 12 - Diagrama de Casos de Uso

UC01: Subscrever eventos

Formato Breve:

Utilizador do Drive inicia sessão no sistema e acede à administração. Sistema disponibiliza árvore de eventos disponíveis. O utilizador especifica qual ou quais os eventos de que deseja ser notificado e grava alterações. Sistema informa sucesso da operação.

Esta troca de informação entre o Sistema e o Utilizador é apresentado no Diagrama de Sequência do Sistema do Sistema (SSD) ilustrado na Figura 13.

Figura 13 - SSD do UC01: Subscrever eventos

Ator principal:

- Utilizador do Drive Web.

Partes Interessadas e seus interesses:

- Utilizador: pretende subscrever um ou mais eventos que ocorram no sistema.

Pré-condições:

- Utilizador autenticado no sistema.

Pós-condições:

- Utilizador subscrito a eventos selecionados.

Cenário de sucesso principal:

1. Utilizador inicia sessão no sistema;
2. Sistema autoriza sessão;
3. Utilizador acede à administração;

4. Sistema disponibiliza árvore com eventos disponíveis à sua subscrição;
5. Utilizador seleciona eventos e guarda alterações;
6. Sistema informa sucesso da operação.

UC02: Cancelar subscrição de eventos

Formato Breve:

Utilizador do Drive inicia sessão no sistema e acede à administração. Sistema disponibiliza árvore de eventos disponíveis. O utilizador desseleciona qual ou quais os eventos que deseja não ser notificado e grava alterações. Sistema informa sucesso da operação.

Esta troca de informação entre o Sistema e o Utilizador é apresentado no Sequência do Sistema do Sistema (SSD) ilustrado na Figura 14.

Figura 14 - SSD do UC02: Cancelar subscrição de eventos

Ator principal:

- Utilizador do Drive Web.

Partes Interessadas e seus interesses:

- Utilizador: pretende cancelar a subscrição de um ou mais eventos que ocorram no sistema.

Pré-condições:

- Utilizador autenticado no sistema;
- Utilizador subscrito a eventos (UC01).

Pós-condições:

- Utilizador não subscrito a eventos.

Cenário de sucesso principal:

1. Utilizador inicia sessão no sistema;
2. Sistema autoriza sessão;
3. Utilizador acede à administração;
4. Sistema disponibiliza árvore com eventos disponíveis;
5. Utilizador seleciona eventos que deseja cancelar a subscrição e guarda alterações;
6. Sistema informa sucesso da operação.

UC03: Pedido de assistência de uma Patrulha

Formato Breve:

Utilizador do Drive inicia sessão no sistema e realiza um pedido de assistência de Patrulha em Incidente. Sistema disponibiliza lista com incidentes. Utilizar seleciona incidente que pretende dar assistência. O Sistema informa sucesso do envio do pedido e envia notificação à Patrulha. Patrulha aceita ou rejeita pedido. Sistema informa operação efetuada.

Esta troca de informação entre o Sistema e o utilizador é apresentado no Diagrama de Sequência do Sistema (SSD) ilustrado na Figura 15.

Figura 15 - SSD do UC03: Pedido de assistência de uma Patrulha

Ator principal:

- Utilizador do Drive Web.

Partes Interessadas e seus interesses:

- Utilizador: pretende enviar uma Patrulha a um Incidente.

Pré-condições:

- Utilizador autenticado no sistema;

Pós-condições:

- Patrulha notificada.

Cenário de sucesso principal:

1. Utilizador inicia sessão no sistema;
2. Sistema autoriza sessão;
3. Utilizador solicita assistência de Patrulha em Incidente;
4. Sistema disponibiliza incidentes;
5. Utilizador escolhe incidente que deseja prestar assistência;
6. O Sistema informa sucesso da operação e notifica Patrulha.

UC04: Notificação de Patrulha em Incidente

Formato Breve:

Utilizador do Drive inicia sessão no sistema e associa uma Patrulha a um Incidente. Sistema informa sucesso da operação e notifica restantes utilizadores acerca da ocorrência.

Esta troca de informação entre o Sistema e o Utilizador é apresentado no Diagrama de Sequência do Sistema (SSD) ilustrado na Figura 16.

Figura 16 - SSD do UC04: Notificação de Patrulha em Incidente

Ator principal:

- Utilizador do Drive Web;
- Utilizador do Drive Mobile.

Partes Interessadas e seus interesses:

- Utilizador: pretende saber em tempo real quando foi feita a associação de uma Patrulha a um Incidente.

Pré-condições:

- Utilizador autenticado no sistema;

- Utilizador subscrito ao evento específico (UC01).

Pós-condições:

- Patrulha associada a Incidente;
- Utilizadores notificados.

Cenário de sucesso principal:

1. Utilizador inicia sessão no sistema;
2. Sistema autoriza sessão;
3. Utilizador associa Patrulha a Incidente;
4. O Sistema informa sucesso da operação e notifica utilizadores interessados.

UC05: Notificação de Patrulha em Descanso

Formato Breve:

Utilizador do Drive inicia sessão no sistema e regista Patrulha em Descanso. Sistema informa sucesso da operação e notifica restantes utilizadores acerca da ocorrência.

Esta troca de informação entre o Sistema e o Utilizador é apresentado no Diagrama de Sequência do Sistema (SSD) ilustrado na Figura 17.

Figura 17 - SSD do UC05: Notificação de Patrulha em descanso

Ator principal:

- Utilizador do Drive Web;
- Utilizador do Drive Mobile.

Partes Interessadas e seus interesses:

- Utilizador: pretende saber em tempo real quando uma Patrulha se encontra em Descanso.

Pré-condições:

- Utilizador autenticado no sistema;
- Utilizador subscrito ao evento específico (UC01).

Pós-condições:

- Registo de Patrulha em descanso;
- Utilizadores notificados.

Cenário de sucesso principal:

1. Utilizador inicia sessão no sistema;
2. Sistema autoriza sessão;

3. Utilizador regista Patrulha em Descanso;
4. O Sistema informa sucesso da operação e notifica utilizadores interessados.

UC06: Notificação de Patrulha em Inspeção

Formato Breve:

Utilizador do Drive inicia sessão no sistema e regista Patrulha em Inspeção. Sistema informa sucesso da operação e notifica restantes utilizadores acerca da ocorrência.

Esta troca de informação entre o Sistema e o Utilizador é apresentado no Diagrama de Sequência do Sistema (SSD) ilustrado na Figura 18.

Figura 18 - SSD do UC06: Notificação de Patrulha em Inspeção

Ator principal:

- Utilizador do Drive Web;
- Utilizador do Drive Mobile.

Partes Interessadas e seus interesses:

- Utilizador: pretende saber em tempo real quando uma Patrulha se encontra em Inspeção.

Pré-condições:

- Utilizador autenticado no sistema;
- Utilizador subscrito ao evento específico (UC01).

Pós-condições:

- Registo de Patrulha em Inspeção;
- Utilizadores notificados.

Cenário de sucesso principal:

1. Utilizador inicia sessão no sistema;
2. Sistema autoriza sessão;
3. Utilizador regista Patrulha em Inspeção;
4. O Sistema informa sucesso da operação e notifica utilizadores interessados.

4.2.2 Especificação Suplementar

Para categorizar os requisitos não funcionais, procedeu-se à sua enumeração segundo o modelo FURPS+ (*Functionality, Usability, Reliability, Performance, Supportability and more*).

Funcionalidade

Capacidade de o produto responder a necessidades relacionadas com a adequação, precisão, interoperabilidade e segurança (Jung, Kim and Chung, 2004).

Foram identificados os seguintes requisitos quanto à funcionalidade do produto:

1. Todas as funcionalidades devem ser protegidas por um método de autenticação e autorização, conforme o papel de cada utilizador no sistema.

Usabilidade

Capacidade de o produto responder a necessidades relacionadas com a compreensão, aprendizagem, atratividade e operabilidade (Jung, Kim and Chung, 2004).

Foram identificados os seguintes requisitos quanto à usabilidade do produto:

1. Eventuais erros que possam ocorrer no sistema devem ser precocemente detetados de forma transparente para o utilizador;

2. A interface deve ser adaptada à interface já utilizada;
3. A interface deve ser fácil e intuitiva para o uso do utilizador.

Confiabilidade

Capacidade de o produto responder a necessidades relacionadas com a tolerância a falhas, capacidade de recuperação e tempo médio entre falhas (Jung, Kim and Chung, 2004).

1. Conforme as necessidades do sistema a aplicação deve garantir a entrega de mensagens.

Desempenho

Capacidade de o produto responder a necessidades relacionadas com a utilização de recursos e desempenho do *software* (Jung, Kim and Chung, 2004).

Não foram identificados requisitos quanto ao desempenho.

Suportabilidade

Capacidade de o produto responder a necessidades relacionadas com a adaptabilidade, instabilidade, estabilidade, compatibilidade, testabilidade, etc (Eeles, 2001; Jung, Kim and Chung, 2004).

Foram identificados os seguintes requisitos quanto à suportabilidade do produto:

1. O sistema deve ser aberto para expansão, permitindo a adição de novas funcionalidades quando necessário;
2. Funcionalidades deverão ser testadas adequadamente.

Restrições de Desenho

Restrições ou especificações a ter no desenho da solução (Eeles, 2001).

Foram identificadas as seguintes especificações quanto as restrições de desenho:

- Desenvolver a solução segundo as melhores práticas de *design* orientado a objetos;
- Suporte MSSQL.

Restrições de Implementação

Restrições ou especificações a ter na implementação e construção da solução (Eeles, 2001).

Foram identificadas as seguintes especificações quanto as restrições de implementação:

- Solução Web desenvolvida em ASP.NET MVC;
- Mobile API desenvolvida em ASP.NET;
- Utilização de linguagem de programação como C#, JavaScript e HTML;
- Adoção de controlo de versões.

Restrições de Interface

Especificações a ter na interação com um sistema externo (Eeles, 2001).

Não foram identificadas restrições quanto à interface.

Restrições Físicas

Especificações que restringem o hardware usado para acolher o sistema (Eeles, 2001).

Não foram identificadas restrições físicas.

4.3 Desenho

Nesta secção é descrito o desenho da solução com recursos a diagramas em notação UML para ilustrar a arquitetura atual do sistema bem como a arquitetura proposta.

4.3.1 Arquitetura da solução

O primeiro passo desta fase é a representação arquitetural atual da solução numa perspetiva de maior abstração. Assim, na Figura 19 encontra-se apresentado o diagrama de vista lógica, nível 2.

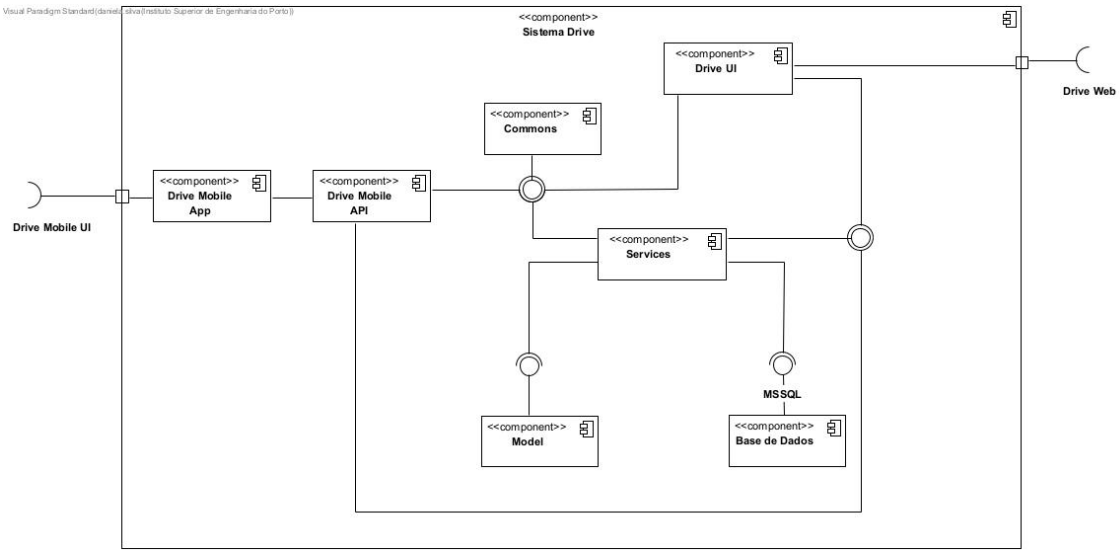


Figura 19 - Vista lógica (nível 2) do diagrama de componentes inicial

Após a identificação de seis componentes estruturais é importante proceder à descrição dos mesmos e das relações entre eles:

- Drive.UI - compreende todos os módulos que constituem o sistema Drive. Este componente fornece uma interface (Drive Web) aos gestores para controlarem as operações que decorrem nas autoestradas;
- Drive Mobile APP - contruída em Xamarin é a aplicação móvel que fornece uma interface (Drive Mobile UI) aos Oficias de assistência e Vigilância (OAVs) que patrulham e inspecionam as vias;
- Drive Mobile API - contruída em REST integra todos os dados disponibilizados e fornecidos pela aplicação móvel (Drive Mobile App);
- Serviços (*Services*) - comum aos componentes Drive.UI e Drive Mobile API, compreende as regras de negócio, sendo responsável pelo acesso à base de dados;
- Modelo (*Model*) - integra todas as entidades do domínio;
- *Commons* - fornece dados às camadas *Services*, Drive.UI e Drive Mobile API. Reúne tudo o que é comum no sistema, nomeadamente as interfaces, as enumerações, as exceções, etc.

A necessidade deste projeto surge com a criação de um *Messager-Broker* entre o Drive.UI e a Drive Mobile APP. O Broker implementado servirá de intermediário entre o Drive Mobile App e o Drive UI de forma a estes poderem comunicar e trocar informação entre si de forma direta e em tempo real.

Messager-Brokers permitem a validação, o armazenamento e o encaminhamento de mensagens entre um Produtor e um Consumidor de destino.

A Figura 20 apresenta um diagrama de vista lógica nível 2 que o sistema Drive terá no final da implementação.

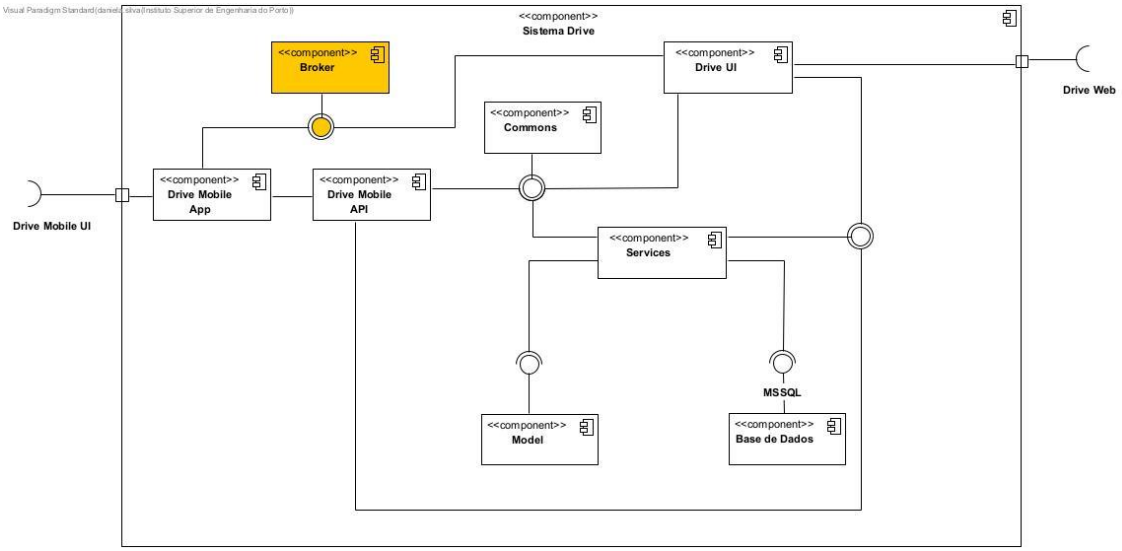


Figura 20 - Vista lógica (nível 2) do diagrama de componentes final

O segundo passo para uma análise completa da arquitetura da solução, passa por uma abordagem mais detalhada dos principais componentes identificados neste produto (Drive Mobile API, Drive.UI e *Services*). Nas Figura 21 e Figura 22 são apresentados diagramas de vista logica, nível 3, do componente Drive.UI e Drive Mobile API respetivamente.

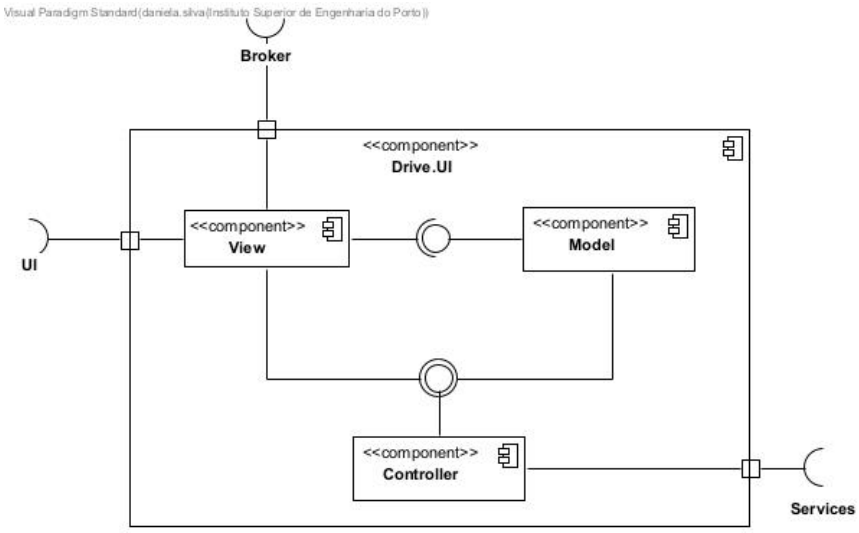


Figura 21 - Vista lógica (nível 3) do Drive.UI

Como anteriormente mencionado na secção de Contexto Tecnológico, o componente Drive.UI segue uma arquitetura MVC, sendo que o conceito de *Model* é substituído pelo *ViewModel*. Nesta arquitetura o utilizador interage com a *View* e por sua vez a *View* requisita ao *Controller* o processamento do pedido. O *Controller* com auxílio dos Serviços envia os dados requisitados (*ViewModel)*, à View. Por fim a *View* é recarregada e mostra os dados ao utilizador.

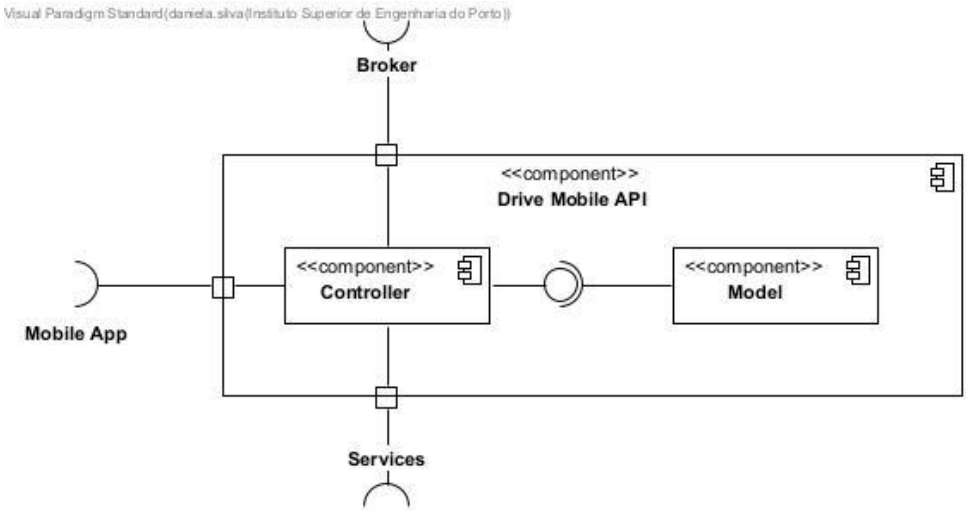


Figura 22 - Vista lógica (nível 3) da Drive Mobile API

A Drive Mobile API, sendo uma REST API apenas é constituída pelos componentes *Controller* e *Model*. A Mobile APP realiza vários pedidos REST ao *Controller*. O *Controller* com auxílio dos Serviços processa o pedido e constrói o *Model* que será enviado na resposta do pedido REST.

Na Figura 23 encontra-se representado num diagrama de classes o componente de serviços. As classes envolvidas na definição de um serviço são:

- *AbstractService* determina a implementação de um serviço genérico. Desta forma, é garantido que todos os serviços serão executados da mesma forma;
- *Service* possui um método *execute* herdado pela classe *AbstractService*. É neste método que se encontra toda a lógica de negócio;
- *IService* implementado no *Service* permite aos *Controllers* obter uma instância do serviço. O uso desta interface permite isolar completamente a camada de Serviços, uma vez que os *Controlleres* não conhecem a implementação dos serviços, apenas a sua abstração.
- *ServiceOutput/ServiceInput* definem, respetivamente, os parâmetros de entrada e de saída de um serviço.

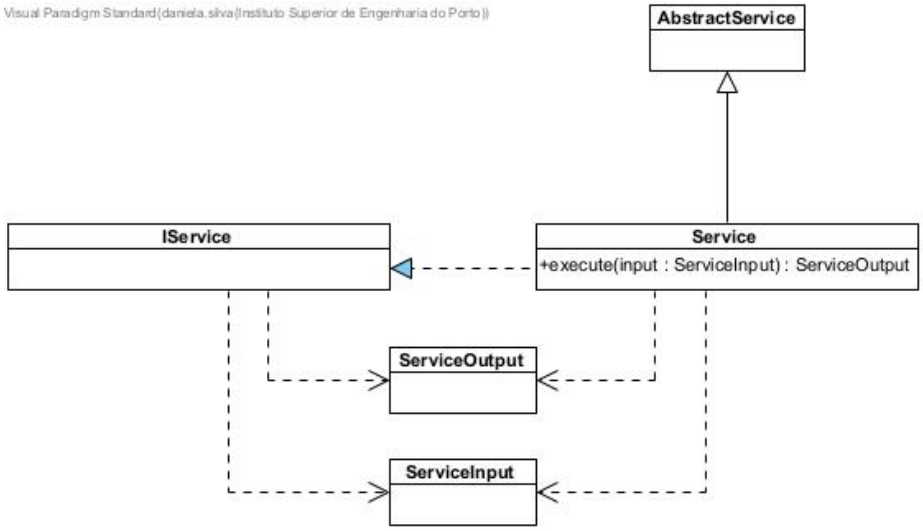


Figura 23 - Diagrama de Classes do componente *Services*

A comunicação entre um *Controller*, tanto do Drive.UI como da Drive Mobile API, com um serviço é apresentada na Figura 24.

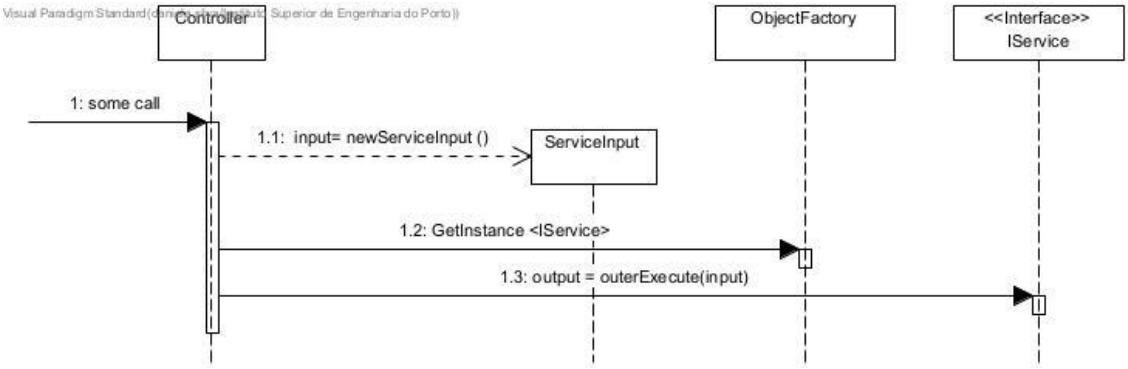


Figura 24 - Diagrama de sequência que representa a comunicação entre um *Controller* e um Serviço

Uma vez que o *Controller* apenas tem conhecimento das interfaces que os Serviços implementam, com recurso ao *StructureMap* que permite o Inversion of Control (IoC) e a Dependency Injection (DI) (StructureMap, no date), a classe *ObjectFactory* permite localizar o serviço pretendido. Esta arquitetura fornece uma vertente mais flexível e escalável a alterações ou implementações futuras.

4.3.2 Desenho dos Casos de Uso

Após o desenho da solução que permitiu a identificação e a relação entre os diferentes componentes, nesta secção é feito um enquadramento à arquitetura da solução com os Casos de Uso levantados. Assim sendo, é apresentada uma breve descrição com recursos a diagramas de sequência (SD) dos UC01: Subscrever Eventos, UC03: Pedido de assistência de uma Patrulha e UC04: Notificação de Patrulha em Incidente. A escolha destes casos de uso surge com o facto de serem os que apresentam maior complexidade e os restantes UCS de alguma forma se assemelham com os apresentados.

UC01: Subscrever Eventos

Na Figura 25 apresenta-se o diagrama de sequência do UC01: Subscrever Eventos. Neste diagrama é detalhado todo processo necessário que o sistema deve executar para este requisito ser cumprido.

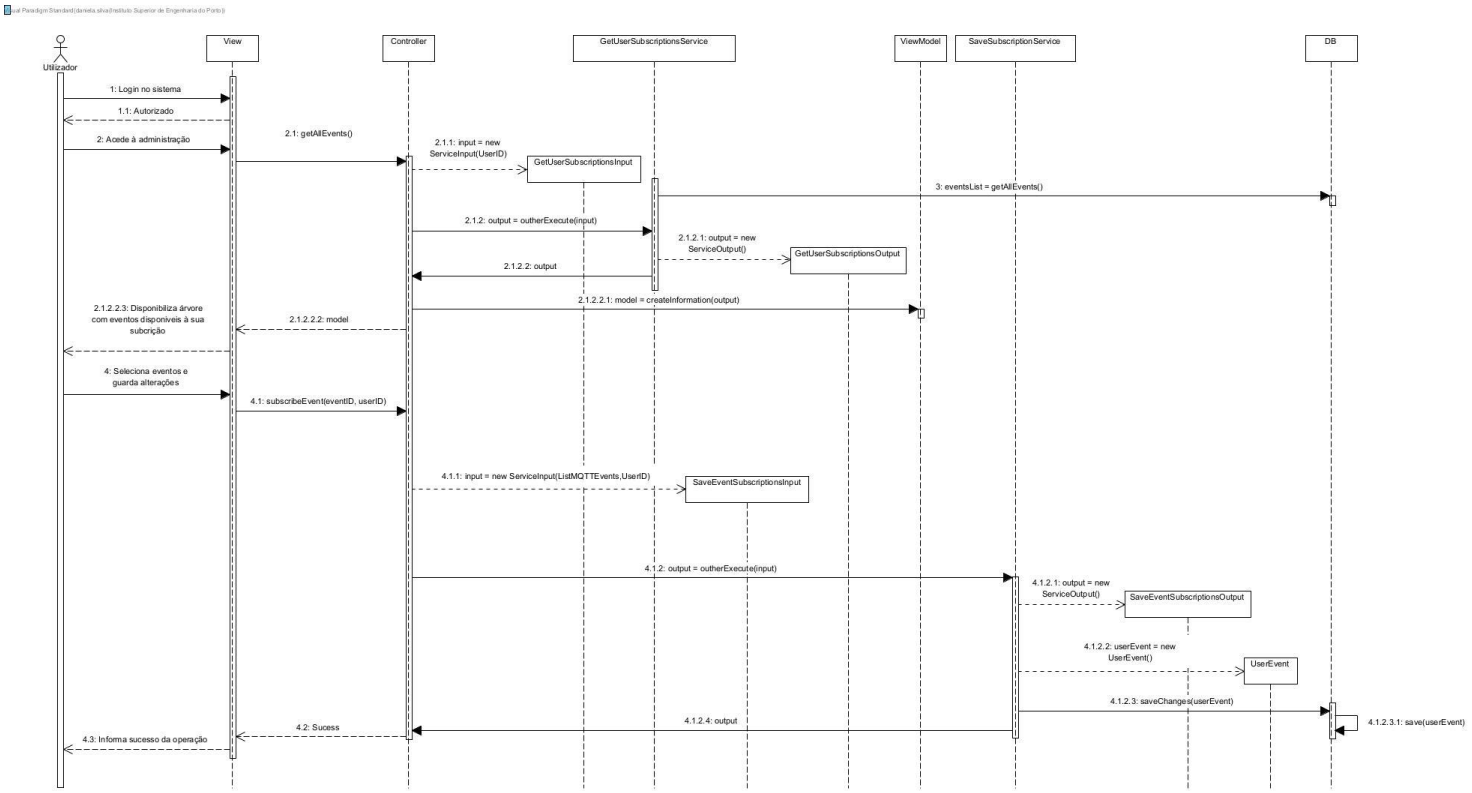


Figura 25 – Diagrama de Sequência do UC01: Subscrever Eventos

Para a implementação da primeira funcionalidade, o utilizador já autenticado no sistema, tem a possibilidade de escolher os eventos que deseja ser notificado.

Após aceder ao módulo da administração do Drive é apresentada uma lista de eventos disponíveis. Para a apresentação desta lista será necessária a criação de um serviço (GetUserSubscriptionsService), que comunica diretamente com a base de dados. Para este serviço é enviado o input GetUserSubscriptionsInput com o identificador do utilizador (UserID). O output deste serviço resulta numa lista de todos os eventos, indicando se o utilizador se encontra previamente subscrito a cada um deles (GetUserSubscriptionsOuput). Esta lista de eventos será disponibilizada ao utilizador através da View.

Após a apresentação desta lista, o utilizador escolhe o evento que quer ser notificado e guarda as alterações efetuadas. Para guardar estas alterações o sistema, mais uma vez, recorre à utilização de um serviço (SaveSubscriptionService). Neste caso, para a execução do serviço será necessário o envio de um input (SaveEventSubscriptionsService) com o identificador do

utilizador e da lista de eventos escolhidos (ListMQTTEvents,userID). O retorno do sucesso da operação é enviado pelo SaveEventSubscriptionsOutput.

UC03: Pedido de assistência de uma Patrulha

Para uma melhor compreensão deste caso de uso, na Figura 26, é apresentado um diagrama de sequência de nível 2. Esta funcionalidade permite a comunicação bidirecional e em tempo real entre os componentes Drive.UI e Drive.Mobile.

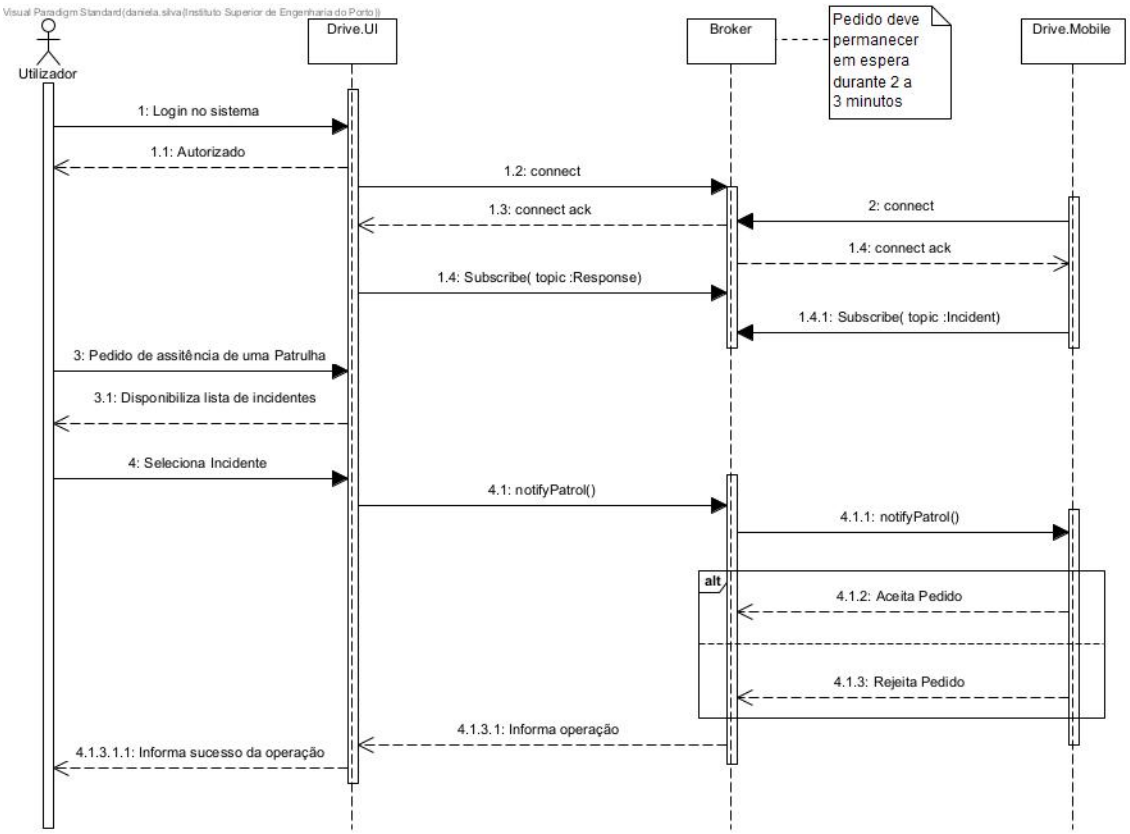


Figura 26 – Diagrama de Sequência do UC03: Pedido de assistência de uma patrulha

Inicialmente, um gestor do Drive que utiliza o Drive.Web, solicita a assistência de uma Patrulha em um Incidente. Este pedido será enviado a um Broker que irá gerir a mensagem. Desta forma o Broker será responsável por guardar a mensagem até um máximo de três minutos.

Findo este período, a mensagem deve expirar caso a Patrulha não receba a mensagem na sua aplicação ou não responder, podendo por exemplo estar offline, e o Gestor deverá ser notificado disso. Essa capacidade é fundamental para garantir que a mensagem seja entregue, mesmo que o destinatário não esteja imediatamente disponível.

No caso da Patrulha receber a mensagem, a mesma será retirada da fila de espera, e o utilizador do Drive.Mobile poderá enviar a resposta em tempo real ao gestor (Aceita ou Rejeita).

UC04: Notificação de Patrulha em Incidente

O caso de uso UC04: Notificação de Patrulha em Incidente, é apresentado no diagrama de sequência da Figura 27, e é muito semelhante ao caso de uso UC05: Notificação de Patrulha em Descanso e ao UC06: Notificação de Patrulha em Inspeção.

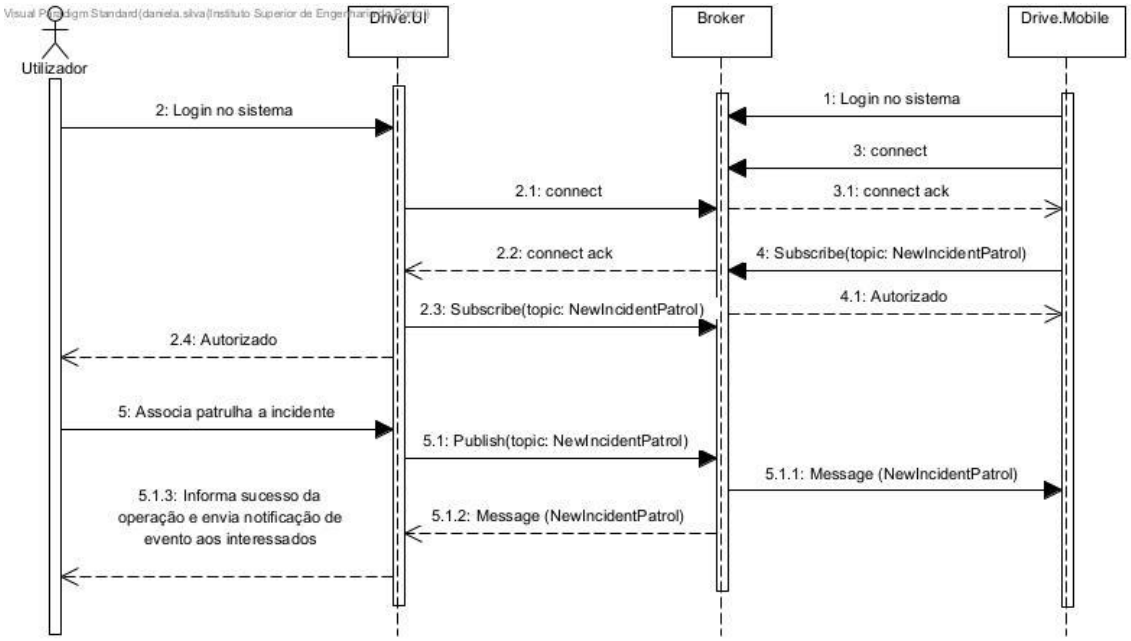


Figura 27 - Diagrama de Sequência do UC04: Notificação de Patrulha em Incidente

Os utilizadores, previamente subscritos ao evento “NewIncidentPatrol” (UC01: Subscrever Eventos), aquando do login no sistema devem conectar-se ao Broker de forma a receber notificação em tempo real quando uma Patrulha é registada em Incidente.

Quando um destes clientes (Drive.Mobile ou Drive.UI) conectados ao Broker publica uma mensagem sobre o tópico “NewIncidentPatrol”, imediatamente a mensagem é distribuída aos clientes que subscreveram este tópico/evento.

5 Implementação da solução

Neste capítulo é apresentada uma descrição e discussão detalhada dos aspetos mais relevantes na implementação do projeto, a qual foi proposta no Design.

São apresentadas as tecnologias escolhidas para a implementação da solução bem como os motivos para a escolha destas tecnologias.

5.1 Gestão de trabalho

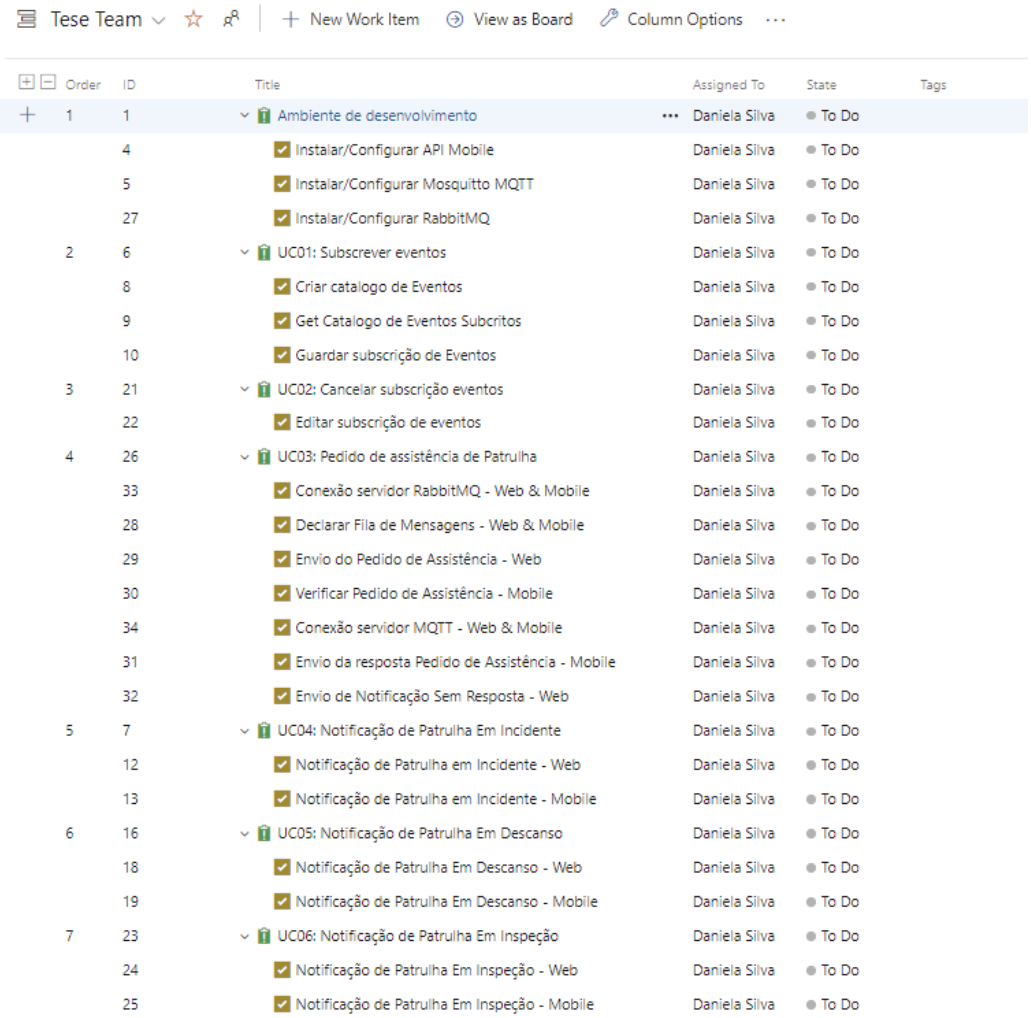
No que diz respeito ao controlo de versões, optou-se por manter a consistência com as ferramentas e processos já estabelecidos na empresa. Assim, para o Desenvolvimento Web, a equipa optou por utilizar o TortoiseSVN como a ferramenta de controlo de versões e para o desenvolvimento móvel, a opção recaiu sobre o Azure DevOps.

Em ambos os projetos, foi criado um *branch* dedicado ao projeto onde o código pôde ser desenvolvido de forma independente em relação à *branch* principal de desenvolvimento mantida pela empresa. Com esta abordagem a equipa de desenvolvimento continua a trabalhar em novos recursos, correções e melhorias sem interferir diretamente no código em produção, mantendo a estabilidade do sistema.

Esta escolha estratégica de manter a consistência com as ferramentas já em uso pela empresa é segura, pois minimiza a necessidade de formação adicional da equipa e garante uma transição suave para o desenvolvimento e gestão do projeto.

No que diz respeito ao planeamento do trabalho, mais uma vez, optou-se por utilizar a abordagem atualmente adotada pela empresa, que é o Azure DevOps. Esta escolha é motivada pelo facto de o Azure DevOps ser uma plataforma completa de gestão do ciclo de vida de um projeto de *software*. Essa plataforma abrange todo o espectro de desenvolvimento de *software*, desde o planeamento até a entrega e operação, incluindo funcionalidades como integração contínua, entrega contínua, entre outras (Azure DevOps Services | Microsoft Azure, no date).

Na Figura 28 encontra-se apresentado o *Product Backlog* inicialmente criado para o projeto a desenvolver.



Order	ID	Title	Assigned To	State	Tags
1	1	Ambiente de desenvolvimento	Daniela Silva	To Do	
	4	Instalar/Configurar API Mobile	Daniela Silva	To Do	
	5	Instalar/Configurar Mosquitto MQTT	Daniela Silva	To Do	
	27	Instalar/Configurar RabbitMQ	Daniela Silva	To Do	
2	6	UC01: Subscrever eventos	Daniela Silva	To Do	
	8	Criar catalogo de Eventos	Daniela Silva	To Do	
	9	Get Catalogo de Eventos Subcritos	Daniela Silva	To Do	
	10	Guardar subscrição de Eventos	Daniela Silva	To Do	
3	21	UC02: Cancelar subscrição eventos	Daniela Silva	To Do	
	22	Editar subscrição de eventos	Daniela Silva	To Do	
4	26	UC03: Pedido de assistência de Patrulha	Daniela Silva	To Do	
	33	Conexão servidor RabbitMQ - Web & Mobile	Daniela Silva	To Do	
	28	Declarar Fila de Mensagens - Web & Mobile	Daniela Silva	To Do	
	29	Envio do Pedido de Assistência - Web	Daniela Silva	To Do	
	30	Verificar Pedido de Assistência - Mobile	Daniela Silva	To Do	
	34	Conexão servidor MQTT - Web & Mobile	Daniela Silva	To Do	
	31	Envio da resposta Pedido de Assistência - Mobile	Daniela Silva	To Do	
	32	Envio de Notificação Sem Resposta - Web	Daniela Silva	To Do	
5	7	UC04: Notificação de Patrulha Em Incidente	Daniela Silva	To Do	
	12	Notificação de Patrulha em Incidente - Web	Daniela Silva	To Do	
	13	Notificação de Patrulha em Incidente - Mobile	Daniela Silva	To Do	
6	16	UC05: Notificação de Patrulha Em Descanso	Daniela Silva	To Do	
	18	Notificação de Patrulha Em Descanso - Web	Daniela Silva	To Do	
	19	Notificação de Patrulha Em Descanso - Mobile	Daniela Silva	To Do	
7	23	UC06: Notificação de Patrulha Em Inspeção	Daniela Silva	To Do	
	24	Notificação de Patrulha Em Inspeção - Web	Daniela Silva	To Do	
	25	Notificação de Patrulha Em Inspeção - Mobile	Daniela Silva	To Do	

Figura 28 - Azure DevOps Backlog

5.2 Implementação

Nesta seção, serão detalhadas as funcionalidades implementadas no projeto, juntamente com uma análise das tecnologias escolhidas para sua concretização.

As funcionalidades são: (i) Cancelar e Subscrever Eventos, (ii) Notificação de Patrulha em Elementos e (iii) Pedido de Assistência de patrulha.

I. Cancelar e Subscrever Eventos

As funcionalidades UC01 e UC02 permitem o registo da subscrição de eventos por parte de um utilizador. Para alcançar esse propósito a tabela MQTTEvent foi criada (Figura 29), na base de dados Microsoft SQL Server, destinada a armazenar o catálogo de eventos e tópicos correspondentes. A estrutura escolhida para a organização dos eventos é hierárquica, onde o evento “Patrulha” atua como o evento pai dos eventos ”Descanso”, ”Inspeção”, “Incidente” e “Pedido de Assistência”. Quando um utilizador opta por ser notificado sobre o evento ”Patrulha”, receberá notificações de todos os eventos filhos.

MQTTEventsID	DESCRIPTION	EventFatherID	MQTTTopic
6	Drive	NULL	drive/+
7	Patrulha	6	drive/patrol/+
8	Inspeção	7	drive/patrol/inspection
9	Descanso	7	drive/patrol/rest
10	Incidente	7	drive/patrol/incident
10002	Pedido de Assistência Resposta	7	drive/patrol/replay
10003	Pedido de Assistência Sem Resposta	7	drive/patrol/notReplay

Figura 29 - Tabela MQTTEvent

O símbolo "+" utilizado nos elementos pai possibilita a substituição de um único nível de tópico. Isso significa que, ao subscrever um tópico com esse símbolo, qualquer tópico que contenha uma sequência arbitrária no lugar desse símbolo será automaticamente correspondido.

Para complementar esta funcionalidade, foi criada a tabela UserMQTTEvents (Figura 30), que possibilita que um utilizador esteja subscrito a um ou mais eventos, permitindo uma gestão flexível e eficaz das subscrições.

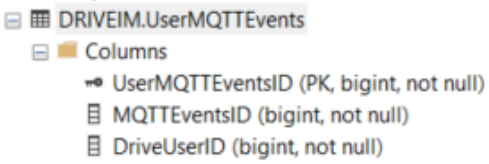


Figura 30 – Tabela UserMQTTEvents

Posto isto, e para que o utilizador tenha acesso ao catálogo de eventos foi criado um serviço capaz de comunicar com a base de dados e devolver em *Data Transfer Object* (DTO) todos os eventos disponíveis e aqueles que se encontra subscritos. O Excerto de código 1 mostra como é processado este serviço.

```
protected override GetUserMQTTEventSubscriptionsOutput execute(GetUserMQTTEventSubscriptionsInput input)
{
    GetUserMQTTEventSubscriptionsOutput output = new GetUserMQTTEventSubscriptionsOutput();

    if (input.UserID.Equals(default(long)))
    {
        throw new InvalidArgumentException("Identificador do utilizador inválido");
    }

    using (var session = Factories.NHibernateSessionFactory.OpenSession())
    {

        List<MQTTEvent> rootEvents = session.Query<MQTTEvent>()
        .Where(e => e.EventFather == null)
        .ToList();

        List<MQTTEventsTreeNode> rootNodes = new List<MQTTEventsTreeNode>();
        foreach (var root in rootEvents)
        {
            MQTTEventsTreeNode elements = MapToMQTTEventsTreeNode(session, input.UserID, root);
            rootNodes.Add(elements);
        }

        output.rootNodes = rootNodes;

    }

    return output;
}
```

Excerto de código 1 – Serviço Get subscrições de um utilizador

Após o utilizador ter acesso aos eventos disponíveis para subscrição, foi desenvolvido um serviço para que o utilizador registe as suas escolhas. Este serviço é responsável por eliminar as escolhas anteriores do utilizador e armazenar as novas, garantindo assim a atualização adequada das suas preferências no sistema (ver Excerto de código 2).

```
protected override SaveUserMQTTEventSubscriptionsOutput execute(SaveUserMQTTEventSubscriptionsInput input)
{
    SaveUserMQTTEventSubscriptionsOutput output = new SaveUserMQTTEventSubscriptionsOutput();
    if (input.UserID.Equals(default(long)))
    {
        throw new InvalidArgumentException("Identificador do utilizador inválido");
    }
    using (var session = Factories.NHibernateSessionFactory.OpenSession())
    {
        using (var trans = session.BeginTransaction())
        {
            var previousSubscriptions = session.Query<UserMQTTEvents>()
                .Where(e => e.DriveUser.DRIVEUSERID == input.UserID).ToList();
            foreach (var pre in previousSubscriptions)
                session.Delete(pre);

            foreach (var MQTTEventID in input.ListMQTTEvents)
            {
                UserMQTTEvents userMQTTEvents = new UserMQTTEvents();
                userMQTTEvents.DriveUser = session.Load<Driveuser>(input.UserID);
                userMQTTEvents.MQTTEvent = session.Load<MQTTEvent>(MQTTEventID);
                session.Save(userMQTTEvents);
            }
            trans.Commit();
        }
    }
    return output;
}
```

Excerto de código 2 – Serviço Save subscrições de um utilizador

Para complementar esta funcionalidade foi desenhada na página de edição dos utilizadores uma árvore de eventos que resulta na Figura 31.

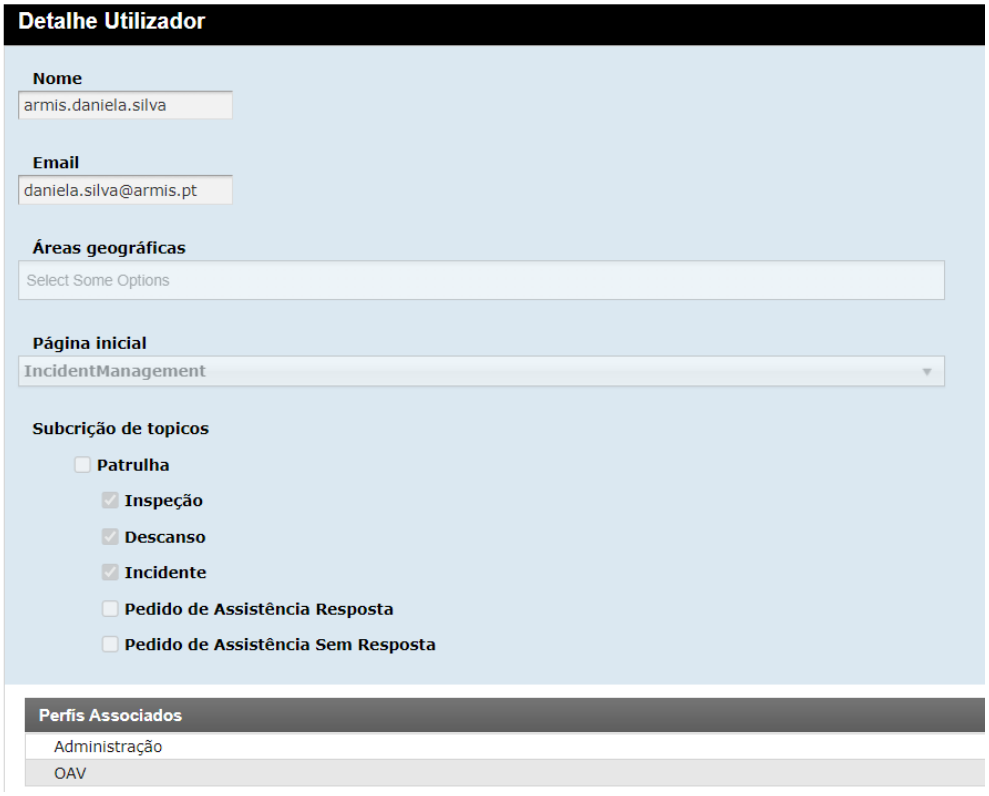


Figura 31 – Árvore de Eventos

II. Notificação de Patrulha em Elementos

Para os casos de uso UC04, UC05 e UC06, que requerem o envio de notificações de "Patrulha" em situações de "Incidente," "Descanso" ou "Inspeção" é necessária a utilização de um protocolo de mensagens, analisados na secção 2.2.3, capaz de enviar estas notificações para todas as plataformas e utilizadores previamente subscritos ao evento desejado.

Nestas notificações, não se espera uma resposta à mensagem enviada, e mais de um utilizador podem ser notificados em simultâneo. Portanto, optou-se pela escolha da tecnologia MQTT em conjunto com o Broker Mosquitto devido à sua simplicidade e adequação às necessidades destes casos de uso.

Para implementar esta solução, iniciou-se com a instalação e configuração do Broker Mosquitto em um servidor fornecido pela Armis. Isto permite estabelecer uma conexão MQTT e, tanto a aplicação Web quanto a aplicação Mobile, podem aceder ao Broker para trocarem notificações de forma eficaz e coordenada.

Após a criação da conexão MQTT no Mosquitto MQTT Broker, prosseguiu-se com a configuração de ambas as plataformas, tanto Mobile quanto Web para que pudessem criar um canal de comunicação com o Broker criado.

Esta configuração envolveu a definição dos parâmetros da conexão MQTT, como o endereço do Broker, o número da porta e credenciais de autenticação. Além disso, as plataformas foram ajustadas para subscreverem os tópicos escolhidos na UC01: Subscrição de Eventos.

No Excerto de código 3 é apresentado como o Drive Web se conecta ao Broker e se subscreve aos eventos.

```
const client = mqtt.connect(host, options)
client.on('error', (err) => {
    console.log('Connection error: ', err)
    client.end()
})
client.on('reconnect', () => {
    console.log('Reconnecting...')
})

$.ajax({
    url: baseUrlLegacy + 'Administration/Users/GetMQTTSubscription',
    cache: false,
    type: "GET",
    datatype: "text/html; charset=utf-8",
    contentType: "application/json",
    success: function (data) {
        $.each(data, function (i, topic) {
            client.subscribe(topic + "/+", function (err) {
                if (err) {
                    console.error('Error subscribing to topic:', err);
                } else {
                    console.log('Subscribed to topic:', topic);
                }
            });
        });
    },
    error: function (data) {
        console.log("GetMQTTSubscription Error : " + data);
    }
});
```

Excerto de código 3 – Drive Web - Conexão com o Broker e subscrição de eventos

Para que os gestores, utilizadores do Drive Web, tenham uma visão completa de todas as atividades no terreno, a subscrição de eventos é configurada de forma ampla, acrescentando o símbolo “+” ao tópico.

Por outro lado, no Drive Mobile, para que os OAVs apenas recebam notificações consigo relacionadas, é acrescentado o identificador da Patrulha ao tópico, ver Excerto de código 4.

```
ClientMqtt _mqttClient = new ClientMqtt();
_mqttClient.Connect();
foreach (var topic in App.MqttTopicsList)
{
    await _mqttClient.Subscribe(topic + "/" + App.PatrolId);
}
```

Excerto de código 4 – Drive Mobile - Conexão com o Broker e subscrição de eventos

Esta diferenciação na subscrição de tópicos garante que os gestores obtenham uma visão global enquanto os OAVs recebem apenas informações diretamente relevantes, evitando uma sobrecarga desnecessária de dados.

Com esta configuração, as plataformas estão prontas a usar o protocolo MQTT para enviar notificações. No Excerto de código 5 encontra-se implantado o método Publish_Application_Message que permite a plataforma Mobile publicar mensagens no Broker MQTT.

```
public void Publish_Application_Message(string topic, MQTTMessage message, int QOS)
{
    var myContent = JsonConvert.SerializeObject(message);
    var buffer = Encoding.UTF8.GetBytes(myContent);

    var applicationMessage = new MqttApplicationMessageBuilder()
        .WithTopic(topic)
        .WithRetainFlag(false)
        .WithPayload(buffer)
        .WithQualityOfServiceLevel((MqttQualityOfServiceLevel)QOS)
        .Build();
    mqttClient.PublishAsync(applicationMessage, CancellationToken.None);
}
```

Excerto de código 5 – Drive Mobile - Envio de notificações para o Broker

É importante destacar que, para os caos de uso em questão, a especificação do tópico de destino e o nível do QoS varia. A garantia da entrega da mensagem (QoS01) foi considerada necessária apenas para a UC04: Notificação de Patrulha em Incidente. As restantes, UC05: Notificação de Patrulha em Descanso e UC06: Notificação de Patrulha em Inspeção, são enviadas com QoS0, sem garantia da entrega da mensagem.

Sempre que uma mensagem relacionada com o tópico ao qual o utilizador se inscreveu chega ao servidor MQTT, a exibição das notificações é facilitada pelo uso do plugin Gritter em Web (conforme mostrado no Excerto de código 6).

```
// Listen for incoming messages
client.on('message', function (Atopic, message) {
    console.log('Received message:', message);
    const decodedString = new TextDecoder().decode(message);
    const parsedObject = JSON.parse(decodedString);
    showGritter(parsedObject.CustomMessageHeader, parsedObject.CustomMessage);
});


function showGritter(PopupHeader, PopupDescrition) {
    PopupHeader += '     ';
    GritterID = $.gritter.add({
        title: PopupHeader,
        text: PopupDescrition,
        sticky: true,
        class_name: 'gritter-success gritter-light'
    });
}
```

Excerto de código 6 - Drive Web – Notificações

Em Mobile a exibição das notificações é facilitada com o uso de noticações (conforme mostrado no Excerto de código 7).

```
public void SendLocalNotification(string title, string message)
{
    try
    {
        var context = Android.App.Application.Context;

        // Create notification
        var intent = new Intent(context, typeof(MainActivity));
        intent.AddFlags(ActivityFlags.ClearTop);
        var pendingIntent = PendingIntent.GetActivity(context, 0, intent, PendingIntentFlags.OneShot);
        var notificationBuilder = new Notification.Builder(context, Constants.CHANNEL_ID)
            .SetSmallIcon(Resource.Drawable.icon)
            .SetContentTitle(title)
            .SetContentText(message)
                .SetOngoing(true);

        // Get the NotificationManager service
        var notificationManager = (NotificationManager)context.GetSystemService(Context.NotificationService);
        notificationManager.Notify(0, notificationBuilder.Build());
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Excerto de código 7 – Drive Mobile - Notificações

III. Pedido de Assistência de patrulha

No caso do UC03: Pedido de assistência de uma Patrulha, optou-se por usar o protocolo AMQP, mais especificamente o RabbitMQ, em conjunto com o Mosquitto MQTT.

Como mencionado anteriormente, na seção 2.2.2, o RabbitMQ, é um *software* de Message-Broker que tem uma capacidade importante: armazenar mensagens em filas, algo que o MQTT não faz automaticamente.

Para esta funcionalidade, as mensagens devem ficar temporariamente em espera até o que o OAV esteja pronto a recebê-la. O RabbitMQ foi a escolha ideal para atender a este requisito.

O Mosquitto MQTT, nesta funcionalidade, entregará as respostas dos pedidos de assistência aos gestores que inicialmente requisitaram este pedido.

Posto isto, para implementar esta funcionalidade foi necessária a instalação e configuração do RabbitMQ e Mosquitto MQTT (previamente instalado nas funcionalidades UC04, UC05 e UC06) no servidor fornecido pela Armis.

Após a instalação de ambos os Brokers, foi estabelecida uma conexão com o Drive (Web e Mobile) com o servidor RabbitMQ para lidar com os pedidos de assistência. Isto inclui a configuração do servidor e as credencias de acesso, para o envio e receção de mensagens (ver Excerto de código 8). As variáveis essenciais à configuração do Broker foram armazenadas em ficheiros de configuração oferecendo uma maior segurança e facilidade na manutenção do sistema.

```
public IConnection GetConnection()
{
    try
    {
        _factory = new ConnectionFactory()
        {
            HostName = ConfigurationManager.AppSettings["HostNameRabbitMQ"],
            UserName = ConfigurationManager.AppSettings["UsernameRabbitMQ"],
            Password = ConfigurationManager.AppSettings["PasswordRabbitMQ"],
            AutomaticRecoveryEnabled = true
        };
        return _factory.CreateConnection();
    }
    catch (Exception e)
    {
        log.Error($"Erro durante criação de conexão ao RabbitMQ - {e.Message}", e);
        throw new Exception(e.Message);
    }
}
```

Excerto de código 8 - Conceção RabbitMQ

Depois do sistema estar conectado ao RabbitMQ, para cada Patrulha, é criada uma fila de mensagens dedicada a armazenar os seus pedidos de assistência por um período máximo de três minutos. Para isto foi utilizado o parâmetro "x-message-ttl" que permite definir o *Time-To-Live* (Tempo de Vida) de uma mensagem, garantindo que a notificação é retida pelo tempo necessário, mas não mais do que o estipulado (ver Excerto de código 9).

```
public void CreateQueue(string queue, string dlQueueName, IModel channel)
{
    channel.QueueDeclare(queue: queue,
     durable: false,
     exclusive: false,
     autoDelete: true,
     arguments: new Dictionary<string, object>
     {
            { "x-dead-letter-exchange", "" },
            { "x-dead-letter-routing-key", dlQueueName},
            { "x-message-ttl", 60000 }
     });
}
```

Excerto de código 9 - Conexão ao servidor RabbitMQ e Fila de Mensagens

Se o RabbitMQ não conseguir entregar a mensagem ao OAV dentro de três minutos, cada fila de mensagens foi configurada para redirecionar essa mensagem para uma *Dead Letter Queue* (fila de mensagens mortas) em vez de descartá-la. Esta abordagem acrescenta uma camada adicional de segurança e rastreabilidade ao processo. Se a Patrulha não receber a mensagem

dentro do prazo estabelecido, as mensagens “dead-lettered” são posteriormente revistas e tratadas adequadamente pelos gestores.

Para que o gestor responsável possa lidar com estas mensagens “mortas”, foi estabelecida uma conexão à fila de mensagens mortas. Quando uma mensagem é adicionada a essa fila o servidor Mosquitto MQTT é notificado. Estas mensagens são envidas com um nível de Qualidade do Serviço de 1 (QoS01), o que implica um processo de entrega de mensagem aos responsáveis com garantia de pelo menos uma vez. Isto assegura que nenhum Pedido de Assistência seja perdido ou ignorado (ver Excerto de código 10).

```
public void RabbitMQReceiver()
{
    if (channel != null)
    {
        CreateQueue(dlQueueName, channel);
        var consumer = new EventingBasicConsumer(channel);
        consumer.Received += (model, ea) =>
        {
            try
            {
                var body = ea.Body.ToArray();
                var message = Encoding.UTF8.GetString(body);
                var json = JsonConvert.DeserializeObject<string>(message);

                ClientMqtt _mqttClient = new ClientMqtt();
                _mqttClient.Connect();
                MQTTResult mqttMessage = new MQTTResult()
                {
                    CustomMessageHeader = "Patrulha " + json.Split('/')[1],
                    CustomMessage = "Patrulha não recebeu Pedido de Assistência",
                    IsSuccessful = true,

                };
                var topic = MQTTTopics.DrivePatrolNotReply + "/" + json.Split('/')[1];
                _mqttClient.Publish_Application_Message(topic,  mqttMessage, 1);
                _mqttClient.Disconnect();
                channel.BasicAck(ea.DeliveryTag, false);
            }
            catch (Exception ex)
            {
                log.Error(ex.ToString());
                channel.BasicNack(ea.DeliveryTag, false, true);
            }
        };
        channel.BasicConsume(queue: dlQueueName, autoAck: false, consumer: consumer);
    }
}
```

Excerto de código 10 – DriveWeb - Mensagens “*dead-lettered”*

No caso do OAV receber o Pedido dentro dos três minutos, ele pode responder à notificação com aceitação ou rejeição do Pedido. Esta notificação também é enviada com o auxílio do Mosquitto MQTT QoS01 (ver Excerto de código 11).

```
consumer.Received += (model, ea) =>
{
    try
    {
        var body = ea.Body.ToArray();
        var message = Encoding.UTF8.GetString(body);
        var json = JsonConvert.DeserializeObject<string>(message);
        bool attend = false;
        Device.BeginInvokeOnMainThread(async () =>
        {
            attend = await DisplayAlert(Resx.AppResources.Warning, json.Split('/')[0], Resx.AppResources.Yes, Resx.AppResources.No);

            ClientMqtt _mqttClient = new ClientMqtt();
            _mqttClient.Connect();
            MQTTResult mqttMessage = new MQTTResult();
            if (attend)
            {
                mqttMessage = new MQTTResult()
                {
                    CustomMessageHeader = "Patrulha " + App.PatrolId,
                    CustomMessage = "Patrulha confirma assistência em Incidente",
                    IsSuccessful = true,
                };
            }
            else
            {
                mqttMessage = new MQTTResult()
                {
                    CustomMessageHeader = "Patrulha " + App.PatrolId,
                    CustomMessage = "Patrulha nega assistência em Incidente",
                    IsSuccessful = true,
                };
            }
            _mqttClient.Publish_Application_Message(MQTTTopics.DrivePatrolReply + "/" + json.Split('/')[1], mqttMessage, 1);
            _mqttClient.Disconnect();
        });
    }
    catch (Exception ex)
    {
        channel.BasicNack(ea.DeliveryTag, false, true);
    }
};
```

Excerto de código 11 – Drive Mobile - Resposta ao Pedido de Assistência

6 Experiências e Avaliação

Nesta secção será apresentado o processo de experimentação e avaliação. Numa primeira fase, será apresentada a hipótese de investigação a ser testada, os indicadores e as fontes de informação. Por fim, será descrita a metodologia de avaliação utilizada.

6.1 Especificação da hipótese de investigação

Tendo por base a Descrição do Problema e dos Objetivos (secções 1.2 e 1.2.1, respetivamente) deste projeto é possível formular duas hipóteses de investigação.

A Hipótese Nula (H0), representa uma teoria que foi apresentada (ou porque acredita-se ser verdade ou porque é usada como base de argumento), mas não aprovada (Shalini Prasad, Ajith Rao and Eeshoo Rehani, 2001).

Hipótese Alternativa (H1), deve ser a oposta de H0, apenas atingida se H0 for rejeitada, é frequentemente a conclusão real desejada (Shalini Prasad, Ajith Rao and Eeshoo Rehani, 2001).

H0: O sistema de notificação não envia as mensagens aos utilizadores finais.

H1: O sistema de notificação envia as mensagens aos utilizadores finais.

6.2 Identificação dos indicadores e fontes de informação

Para avaliar as hipóteses formuladas é necessário identificar os indicadores de avaliação e fontes de informação a utilizar neste projeto.

6.2.1 Indicadores de avaliação

Os indicadores de avaliação têm como objetivo apresentar os resultados de forma quantitativa ou qualitativa auxiliando na avaliação do projeto realizado. Com isto, foram definidos os seguintes indicadores de avaliação:

- Usabilidade: esta métrica avalia a compreensão e a facilidade do uso do *software.* Uma alta usabilidade que atende aos requisitos do cliente permite uma maior facilidade, rapidez e eficiente utilização do *software*;

- Fluxo de mensagens: o objetivo deste indicador é avaliar a quantidade de mensagens publicadas/subscritas por unidade de tempo. Quanto maior o fluxo de mensagens enviadas maior é a capacidade do *software* para lidar com um grande volume de dados;

- Latência: com este indicador é possível medir o tempo de chegada de uma mensagem deste o remetente até ao recetor. Uma pequena latência na entrega da mensagem indica que o *software* é capaz de transmitir mensagens com um atraso mínimo;

- Confiabilidade: este indicador averigua a entrega de mensagens sem perdas ou duplicação. Quanto maior a confiabilidade do *software* mais fiável é a entrega de mensagens sem erros;

- Compatibilidade: este indicador mede a capacidade do sistema de mensagens contruído funcionar em diversos dispositivos. Quanto maior a compatibilidade do *software* mais versátil o sistema se tornará.

6.2.2 Fontes de informação

Para avaliar os indicadores de avaliação previamente definidos foram definidas as seguintes fontes de informação:

- Teste de Carga;
- Questionários.

6.3 Descrição da metodologia de avaliação

Tendo em consideração as hipóteses enunciadas, a identificação dos indicadores e fontes de informação, nesta secção será descrita a metodologia de avaliação.

6.3.1 Testes de Carga

Estes testes têm como objetivo avaliar a latência e a confiabilidade do sistema de notificações construído.

O primeiro passo para a execução destes testes é a simulação de um ambiente real. Para isto serão conectados diferentes utilizadores ao sistema de notificações de mensagens, será definida a frequência de envio de mensagens, o tempo de espera entre cada mensagem enviada e qualidade de entrega da mensagem. Após o primeiro teste ser realizado, as métricas serão registadas e, novamente, serão executados testes com configurações diferentes às anteriores. Por fim, os resultados dos testes de carga serão avaliados de modo a garantir que o sistema atende as métricas estabelecidas.

6.3.2 Questionários

O questionário criado permite avaliar todas métricas anteriormente definidas. Para isto, serão disponibilizados questionários aos *stakeholders* com perguntas relativas à usabilidade, fluxo de mensagens, latência, confiabilidade e compatibilidade.

6.4 Resultados da metodologia de avaliação

6.4.1 Testes de carga

Para a realização dos testes de carga, optou-se pela tecnologia JMeter devido à sua utilização gratuita, experiência prévia do autor e a capacidade de adicionar *plugins* que suportam os protocolos de mensagens utilizados neste projeto: MQTT e AMQP.

Em ambos os protocolos foram conduzidos testes de carga para avaliar o desempenho destes protocolos neste projeto e em vários cenários:

- Cenário 1: 50 utilizadores um simultâneo
    - Tempo entre utilizadores: 0 milissegundos (ms), nenhum tempo de espera entre os utilizadores
- Cenário 2: 50 utilizadores com intervalo entre mensagens
    - Tempo entre utilizadores: 60 segundos de intervalo entre os utilizadores
- Cenário 3: 200 utilizadores um simultâneo
    - Tempo entre utilizadores: 0 milissegundos, nenhum tempo de espera entre os utilizadores
- Cenário 4: 200 utilizadores com intervalo entre mensagens
    - Tempo entre utilizadores: 60 segundos de intervalo entre os utilizadores

MQTT

No âmbito da avaliação do protocolo MQTT, foram executados diversos testes (12 no total) que variam no número de mensagens enviadas, no intervalo entre elas e nos níveis de QoS.

Para a realização destes testes, foram adicionados e configurados dois tipos de utilizadores, representados por Threads no contexto do JMeter: Subscritores e Publicadores. Para cada tipo de utilizador, foi adicionada e configurada a conexão ao MQTT (MQTT *Connect*), especificando parâmetros como o nome e o número da porta do servidor.

No caso da subscrição de mensagens (MQTT Sub), apenas é necessário configurar o tópico ao qual o pedido deve responder. Para a publicação de mensagens foi indicado o QoS, o tópico e o *payload* a enviar. Após o envio das mensagens foi realizada a desconexão ao servidor a cada operação de publicação (MQTT DisConnect).

Este conjunto de operações permitiu uma avaliação detalhada do desempenho que o protocolo MQTT terá neste projeto em diferentes cenários e condições. Na Figura 32 é apresentado o fluxo e conjunto de operações que foram adicionados para atender ao propósito destes testes.

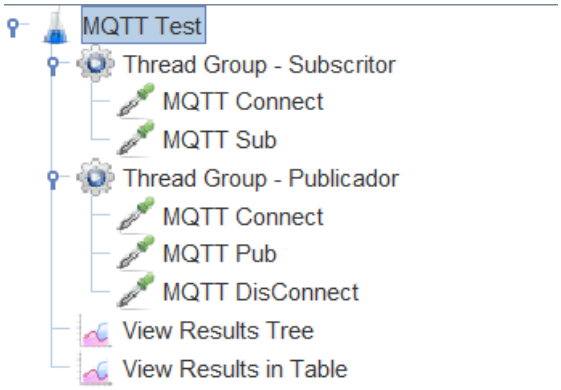


Figura 32 – JMeter Operações MQTT

Na Tabela 5 é apresentada uma breve descrição dos resultados obtidos.

Tabela 5 – Resultados dos testes de carga ao protocolo MQTT

Quantidade de mensagens a enviar	**Tempo entre mensagens(s)**	**Qualidade de entrega**	**Latência (ms)**	**Confiabilidade**
50 Utilizadores	**0**	**QOS0**	0 a 1	50/50 mensagens entregues
		QOS01	0 a 55	50/50 mensagens entregues
		QOS02	0 a 77	50/50 mensagens entregues
	60	**QOS0**	0 a 20	50/50 mensagens entregues
		QOS01	0 a 71	50/50 mensagens entregues

200 Utilizadores	**0**	**QOS02**	0 a 137	50/50 mensagens entregues
		QOS0	0 a 4	165/200 mensagens entregues
		QOS01	0 a 55	200/200 mensagens entregues
		QOS02	0 a 111	200/200 mensagens entregues
	60	**QOS0**	0 a 1	200/200 mensagens entregues
		QOS01	0 a 69	200/200 mensagens entregues
		QOS02	0 a 102	200/200 mensagens entregues

Em resumo, o MQTT demostrou ser capaz de manter alta confiabilidade, independentemente do nível de QoS ou da frequência de envio.

Para uma latência mais baixa, o QoS0 é preferível, especialmente quando as mensagens são enviadas em simultâneo. Quando as mensagens são enviadas com um espaço entre elas a latência aumentou, mas a confiabilidade permaneceu alta.

AMQP

No que diz respeito à avaliação do protocolo AMQP, também foram realizados testes semelhantes aos realizados com o MQTT. Foram executados um total de 4 testes, que variam no número de mensagens enviadas e no intervalo entre elas.

Para a execução destes testes foi criada uma Thread que representa o Produtor. O Produtor foi configurador com diversos parâmetros, o nome do Servidor, o número da porta, as credenciais de autenticação, o nome da fila e o *exchange* ao qual as notificações deveriam ser enviadas, ver Figura 33.

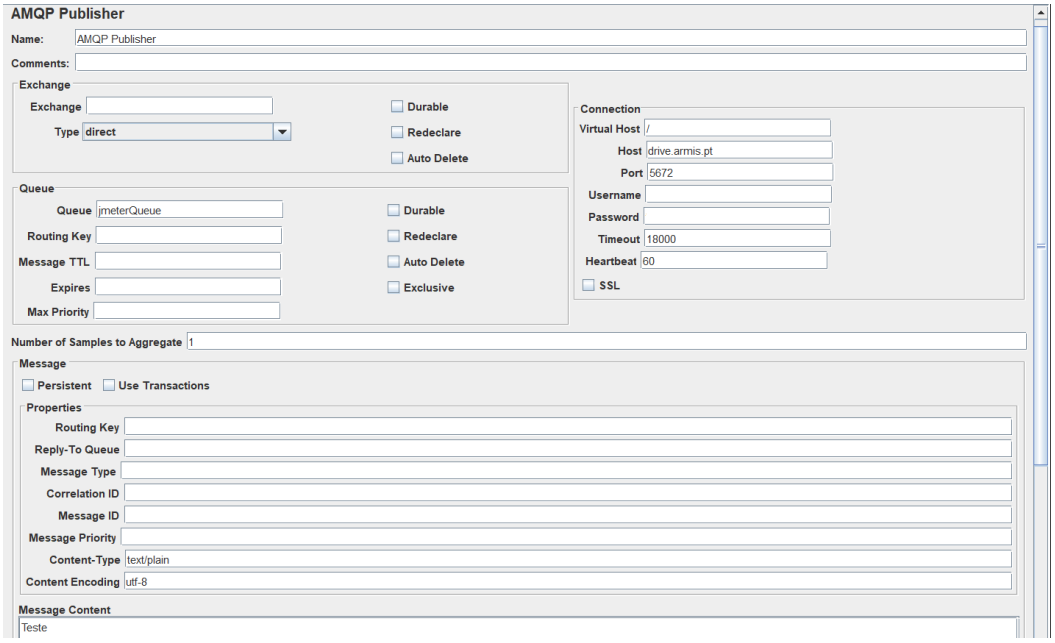


Figura 33 – Jmeter AMQP Produtor

Na Tabela 6 apresentada uma descrição breve dos resultados obtidos.

Tabela 6 - Resultados dos testes de carga ao protocolo AMQP

Quantidade de mensagens a enviar	**Tempo entre mensagens(s)**	**Latência (ms)**	**Confiabilidade**
50 Utilizadores	**0**	0	50/50 mensagens entregues
	60	0	50/50 mensagens entregues
200 Utilizadores	**0**	0	200/200 mensagens entregues
	60	0	200/200 mensagens entregues

Com os resultados obtidos, podemos concluir que o protocolo AMQP mantém a confiabilidade alta, independentemente da quantidade de mensagens a enviar ou do tempo entre elas, sem impactar a latência.

6.4.2 Questionários

Com o objetivo de garantir que o sistema de notificações contruído satisfaz as necessidades e requisitos dos *stakeholders*, optou-se por desenvolver um questionário específico para recolher o *feedback* dessas partes interessadas. A escolha de utilizar a plataforma Google Forms para a criação deste questionário foi motivada pela facilidade de construção que a ferramenta oferece, bem como pela familiaridade tanto do autor quanto dos *stakeholders.* É relevante ressaltar que os próprios *stakeholders* são os únicos responsáveis por responder a este questionário, dada a sua *expertise* e envolvimento com o sistema em questão.

Este questionário, apresentado no Anexo A - Questionário Sistema de Notificações Drive, foi cuidadosamente dividido em cinco secções distintas, cada uma com um foco específico. Esta abordagem tem como objetivo facilitar a recolha e análise do feedback. A seguir, é fornecida uma breve descrição de cada uma das secções:

- Secção da Usabilidade: as perguntas permitem avaliar a facilidade com que o utilizador consegue encontrar e configurar a subscrição de eventos de um utilizador específico e se as notificações recebidas são intuitivas;
- Secção do Fluxo de mensagens: perguntas que abordam a quantidade de eventos enviados para o sistema de notificações;
- Secção de Latência: perguntas relacionadas com o tempo médio que uma mensagem leva até ser entregue ao seu destinatário;
- Secção de Confiabilidade: perguntas acerca da perda ou duplicação de mensagens durante o uso do sistema;
- Secção de Compatibilidade: perguntas acerca da receção e envio de mensagens em ambas as plataformas, Mobile e Web.

Em análise, ver Anexo B - Questionário Sistema de Notificações Drive - Respostas , podemos concluir que, no que diz respeito à secção da usabilidade, ainda há espaço para melhorias na aparência das notificações para tornar a experiência do utilizador ainda mais agradável. A latência não se apresentou nenhum problema em relação as mensagens enviadas, com tempos de entrega rápidos.

Por fim, é importante destacar que tanto a confiabilidade quanto a compatibilidade não apresentaram problemas.

6.5 Avaliação

Ao analisar os resultados dos testes, é possível fazer uma avaliação à hipótese de investigação inicialmente definida na seção 6.1. Com isto, podemos concluir que a H1: O sistema de notificação envia as mensagens aos utilizadores finais foi comprovada. Durante os testes, o sistema de notificações demostrou ser eficaz no envio de mensagens aos utilizadores finais.

7 Conclusão

Neste capítulo são apresentados os objetivos atingidos, identificando quais dos objetivos iniciais foram alcançados. Em seguida, são identificadas as contribuições deste projeto e discutidas as dificuldades encontradas e algumas possíveis melhorias. Por fim, o autor apresenta a sua apreciação final.

7.1 Objetivos concretizados

Tendo em conta os objetivos inicialmente propostos, na secção 1.2.1, nesta secção serão detalhados os desenvolvimentos alcançados para cada um dos objetivos:

1. Analisar o contexto do desenvolvimento do produto, como as tecnologias e ferramentas atualmente utilizadas;

Tendo em consideração que o Drive existe há 15 anos, o estudo do contexto de desenvolvimento do produto foi realizado. Isto inclui a análise arquitetural do sistema Drive e de todos os seus componentes relevantes à realização deste projeto, bem como as tecnologias e ferramentas atualmente em uso. Compreender o contexto existente foi fundamental para tomar decisões informadas durante o desenvolvimento do sistema de notificações.

2. Pesquisar soluções existentes no mercado atual que utilizem um sistema de notificações, em tempo real, entre aplicações;

Foi realizado um estudo detalhado para identificar outras soluções que possuem um sistema de notificações em tempo real, incluindo as aplicações SustIMS, dTIMS, Onfleet e aquele já implementado no Drive.

3. Identificar e analisar as arquiteturas e tecnologias mais adequadas à resolução do problema;

Diversas arquiteturas e tecnologias, incluindo os protocolos HTML, AMQP e MQTT, foram estudados e analisados. Esta análise ajudou a identificar as opções mais adequadas para a resolução do problema.

4. Analisar e decidir, entre as alternativas, quais as arquiteturas e tecnologias que, potencialmente, serão mais adequadas à resolução do problema;

Com base na análise, a decisão tomada foi de utilizar os protocolos AMQP (usando RabbitMQ) e MQTT (usando Mosquitto) para implementar o Sistema de Notificações em Tempo Real. Estas escolhas foram tidas de acordo com os requisitos do projeto e com a eficiência destas.

5. Criar um catálogo de eventos para que o Drive envie mensagens aos consumidores autorizados a subscreverem o evento;

Um catálogo de eventos foi criado em SQL Server, e uma interface na administração dos utilizadores foi disponibilizada para permitir que os utilizadores especifiquem os eventos dos quais desejam ser notificados. Isto facilita a gestão das preferências de notificações por parte dos utilizadores.

6. Implementar um sistema de notificações no produto com a arquitetura e ferramenta escolhida.

O sistema de notificações foi implementado com sucesso, utilizando as tecnologias escolhidas. As notificações foram programadas para serem enviadas ao Broker, assegurando a entrega de notificações aos consumidores autorizados. Além disto, foram desenvolvidos testes para garantir que os requisitos inicialmente propostos foram alcançados.

Em resumo, todos os objetivos inicialmente propostos foram concretizados com êxito. O Sistema de Notificações em Tempo Real foi implementado de acordo com as escolhas da arquitetura e tecnologias feitas após a análise cuidada do contexto e das soluções existentes no mercado.

7.2 Contribuições

Este projeto contribuiu significativamente para uma melhor gestão das infraestruturas. A estrutura e as notificações contruídas fornecem uma base sólida para melhorar a eficiência da gestão das infraestruturas rodoviárias. Este sistema de notificações permite uma comunicação mais ágil entre os diferentes utilizadores do sistema, auxiliando na tomada de decisões mais informadas e permitindo uma resposta mais rápida a situações que o exigem. Além disso, as notificações já implementadas proporcionam informações rápidas sobre as atividades daqueles que estão nas infraestruturas, permitindo um acompanhamento mais eficaz das operações em curso.

7.3 Dificuldades encontradas

Este projeto exigiu a instalação dos Brokers MQTT e AMQP em um servidor remoto para garantir o acesso a ambas as plataformas do Drive. Esta exigência trouxe algumas dificuldades aquando da fase de implementação. Inicialmente, foi configurado e utilizado um servidor no Azure, mas este tronou-se inoperacional devido a problemas de subscrição, resultando em um atraso no projeto. Para superar este obstáculo, foi decidido reinstalar e configurar as tecnologias em outro servidor, desta vez da Armis.

Além disto, a tenologia Mosquitto MQTT apresentou um desafio adicional, uma vez que o autor não tinha experiência anterior com ela, seja em contexto profissional ou académico.

7.4 Trabalho futuro

Apesar da implementação bem-sucedida de todas as tarefas, é importante reconhecer que existe espaço para melhorias contínuas, especialmente para que o Sistema de Notificações em Tempo Real se mantenha alinhado com as necessidades do cliente em contante evolução.

Posto isto, o trabalho futuro incluirá a expansão do catálogo de eventos, adicionando novas categorias e elementos relevantes. Além disso, será necessário programar o sistema associado a esses eventos. Estas notificação permitirão que os eventos sejam enviados com sucesso ao Broker já construído.

7.5 Apreciação final

O projeto corresponde de forma positiva aos requisitos propostos pelo cliente. Todo o trabalho alcançado foi resultado de muita dedicação, uma vez que exigiu um estudo detalhado de todas as tecnologias necessárias para a implementação dos requisitos. Este compromisso e esforço dedicado ao projeto foram essenciais para alcançar os resultados obtidos até ao momento.

À medida que o projeto continua a evoluir, estas melhorias contribuirão para aperfeiçoar ainda mais a qualidade e eficácia do sistema de notificações, mantendo-o alinhado com as espectativas e as demandas em contante mudança.

Referências

Aiyagari, S. *et al.* (2008) 'AMQP Advanced Message Queuing Protocol Protocol Specification A General-Purpose Messaging Standard Technical Contributors Envoy Technologies Rafael Schloming Red Hat Matthias Radestock Rabbit Technologies'.

Al-Masri, E. *et al.* (2020) 'Investigating Messaging Protocols for the Internet of Things (IoT)', *IEEE Access*, 8, pp. 94880–94911. Available at: https://doi.org/10.1109/ACCESS.2020.2993363.

AMQP 0-9-1 Model Explained — RabbitMQ (no date). Available at: https://www.rabbitmq.com/tutorials/amqp-concepts.html (Accessed: 28 January 2023).

Armis | Armis (no date). Available at: https://www.armis.pt/ (Accessed: 19 January 2023).

Armis | Drive (no date). Available at: https://www.armis.pt/intelligent-transport-systems/produtos/drive/ (Accessed: 17 January 2023).

Ascendi (no date). Available at: https://www.ascendi.pt/ (Accessed: 19 January 2023).

ASP.NET MVC | Microsoft Learn (no date). Available at: https://learn.microsoft.com/pt-br/aspnet/mvc/ (Accessed: 10 September 2023).

ASP.NET Web APIs | Rest APIs with .NET and C# (no date). Available at: https://dotnet.microsoft.com/en-us/apps/aspnet/apis (Accessed: 29 January 2023).

Azure DevOps Services | Microsoft Azure (no date). Available at: https://azure.microsoft.com/en-us/products/devops (Accessed: 4 September 2023).

Code of Ethics | IEEE Computer Society (no date). Available at: https://www.computer.org/education/code-of-ethics (Accessed: 16 February 2023).

dTIMS Business Analytics (BA) — dTIMS® | Infrastructure Asset Management Software (no date). Available at: https://www.deighton.com/dtims-business-analytics (Accessed: 31 January 2023).

dTIMS Business Intelligence (BI) — dTIMS® | Infrastructure Asset Management Software (no date). Available at: https://www.deighton.com/dtims-business-intelligence (Accessed: 31 January 2023).

dTIMS Mobile — dTIMS® | Infrastructure Asset Management Software (no date). Available at: https://www.deighton.com/dtims-mobile (Accessed: 31 January 2023).

dTIMS Operations Management (OM) — dTIMS® | Infrastructure Asset Management Software (no date). Available at: https://www.deighton.com/dtims-operations-management (Accessed: 31 January 2023).

Eeles, P. (2001) 'Capturing Architectural Requirements What Is an Architectural Requirement?' Available at: http://www.therationaledge.com/content/nov_01/t_architecturalRequirements_pe.html (Accessed: 5 February 2023).

Ferreira, A. *et al.* (no date) 'MANUAL DE OBSERVAÇÃO E AUSCULTAÇÃO DE INFRAESTRUTURAS GEOTÉCNICAS'. Available at: https://www.ascendi.pt/ (Accessed: 30 January 2023).

Gemirter, C.B., Çenturca, Ş. and Baydere, Ş. (2021) 'A Comparative Evaluation of AMQP, MQTT and HTTP Protocols Using Real-Time Public Smart City Data', *Proceedings - 6th International Conference on Computer Science and Engineering, UBMK 2021*, pp. 542–547. Available at: https://doi.org/10.1109/UBMK52708.2021.9559032.

Gestão da Manutenção | Ascendi (no date). Available at: https://www.ascendi.pt/areas-de-negocio/operacao-manutencao/gestao-da-manutencao/ (Accessed: 30 January 2023).

Intelligent Transportation System Market Size Report, 2030 (no date). Available at: https://www.grandviewresearch.com/industry-analysis/intelligent-transportation-systems-industry (Accessed: 11 February 2023).

Jonnatan, L ; *et al.* (2022) 'Mobile Device Usage before and during the COVID-19 Pandemic among Rural and Urban Adults', *International Journal of Environmental Research and Public Health 2022, Vol. 19, Page 8231*, 19(14), p. 8231. Available at: https://doi.org/10.3390/IJERPH19148231.

Jung, H.W., Kim, S.G. and Chung, C.S. (2004) 'Measuring software product quality: A survey of ISO/IEC 9126', *IEEE Software*, 21(5), pp. 88–92. Available at: https://doi.org/10.1109/MS.2004.1331309.

Naik, N. (2017) 'Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP', *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings* [Preprint]. Available at: https://doi.org/10.1109/SYSENG.2017.8088251.

Onfleet | Company (no date). Available at: https://onfleet.com/company (Accessed: 2 February 2023).

Onfleet | Features (no date). Available at: https://onfleet.com/features (Accessed: 2 February 2023).

Overview of ASP.NET Core MVC | Microsoft Learn (no date). Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-3.1#razor-view-engine. (Accessed: 28 January 2023).

Projeto SustIMS (no date). Available at: https://www.eng.uminho.pt/pt/investigareinovar/projetoscomempresas/Paginas/projetosustims.aspx (Accessed: 30 January 2023).

RabbitMQ tutorial - 'Hello world!' — RabbitMQ (no date). Available at: https://www.rabbitmq.com/tutorials/tutorial-one-python.html (Accessed: 28 January 2023).

Razor syntax reference for ASP.NET Core | Microsoft Learn (no date). Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-7.0 (Accessed: 28 January 2023).

Sachdeva, S. (2016) 'Scrum Methodology', *International Journal Of Engineering And Computer Science*, 5, pp. 16793–16799. Available at: https://doi.org/10.18535/ijecs/v5i6.11.

Shahri, E., Pedreiras, P. and Almeida, L. (2021) 'Enhancing MQTT with Real-Time and Reliable Communication Services', *IEEE International Conference on Industrial Informatics (INDIN)*, 2021-July. Available at: https://doi.org/10.1109/INDIN45523.2021.9557514.

Shalini Prasad, Ajith Rao and Eeshoo Rehani (2001) 'DEVELOPING HYPOTHESIS AND RESEARCH QUESTIONS'. 500 RESEARCH METHODS, pp. 1–30.

'SISTEMA INTELIGENTE DE GESTÃO DE INFRAESTRUTURAS RODOVIÁRIAS' (no date). Available at: www.ascendi-group.com (Accessed: 30 January 2023).

StructureMap (no date). Available at: http://structuremap.github.io/ (Accessed: 6 February 2023).

SustIMS | Plataforma Ascendi para Gestão de Infraestruturas - YouTube (no date). Available at: https://www.youtube.com/watch?v=GiUVm99ItE4 (Accessed: 30 January 2023).

THE BUSINESS MODEL ONTOLOGY A PROPOSITION IN A DESIGN SCIENCE APPROACH (2004). Available at: https://d1wqtxts1xzle7.cloudfront.net/30373644/thebusiness-model-ontology-libre.pdf?1391809579=&response-content-disposition=inline%3B+filename%3DThe_Business_Model_Ontology_a_propositio.pdf&Expires=1677015754&Signature=AAlRIhmxvQbvLCFxq0VlljHU0zMo37JayykHRo6NbK6BEz9UGfH84pnwfP5~qodOVyG9k1scaV6DjJEpoDyamHKBdRJC6eQOasCYgHFhopaSL8WFFV9guqWJWP7klngQzTl7cp1TrpGtxsYnT7UghblL~RzxOj-C4b7proofaPyJH9djMKGKjwWMlMZ9MrgIMF79x18sfNK33rvto8PxFD5yt~HFMtuBUOuupjM6VIjGAxtytZT4TxFVGpbVAqUe2BNWqcwz7knh~jtjUdon9uIx5eg~c3WQoPbinZHBP8c3YbV0W9zCEeJZ1cMNhKUdLy2cxLg~zlLY8SvZZ1VKzw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA (Accessed: 21 February 2023).

The PDMA ToolBook 1 for New Product Development - Google Livros (no date). Available at: https://books.google.pt/books?hl=pt-BR&lr=&id=kqX5EvT2U8AC&oi=fnd&pg=PA5&dq=Fuzzy+Front+End:+Effective+Methods,+Tools,+and+Techniques&ots=8LpnabuMj9&sig=SHvtfC3NZxgq0K5ogZyDeq8tUVE&redir_esc=y#v=onepage&q=Fuzzy%20Front%20End%3A%20Effective%20Methods%2C%20Tools%2C%20and%20Techniques&f=false (Accessed: 8 February 2023).

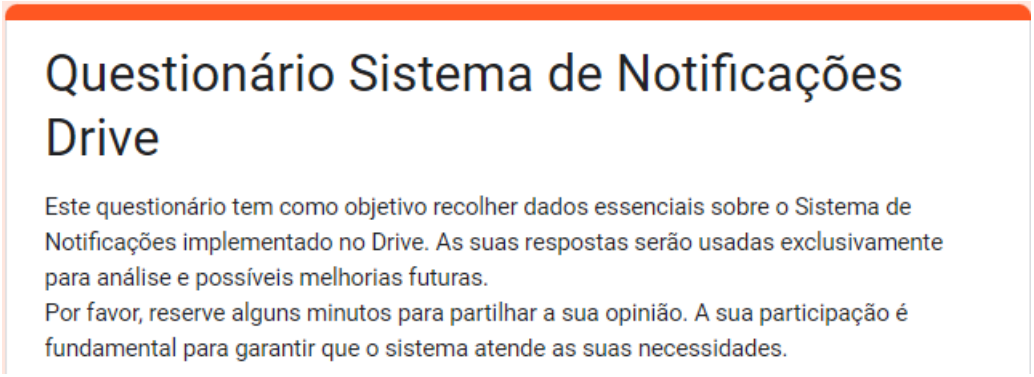
Uy, N.Q. and Nam, V.H. (2019) 'A comparison of AMQP and MQTT protocols for Internet of Things', *Proceedings - 2019 6th NAFOSTED Conference on Information and Computer Science, NICS 2019*, pp. 292–297. Available at: https://doi.org/10.1109/NICS48868.2019.9023812.

Warwick, © (no date) *Quality Function Deployment*.

Xamarin | Open-source mobile app platform for .NET (no date). Available at: https://dotnet.microsoft.com/en-us/apps/xamarin (Accessed: 2 February 2023).

Anexos

Anexo A - Questionário Sistema de Notificações Drive


Questionário Sistema de Notificações Drive

Este questionário tem como objetivo recolher dados essenciais sobre o Sistema de Notificações implementado no Drive. As suas respostas serão usadas exclusivamente para análise e possíveis melhorias futuras.
Por favor, reserve alguns minutos para partilhar a sua opinião. A sua participação é fundamental para garantir que o sistema atende as suas necessidades.

Figura 34 – Descrição do Questionário

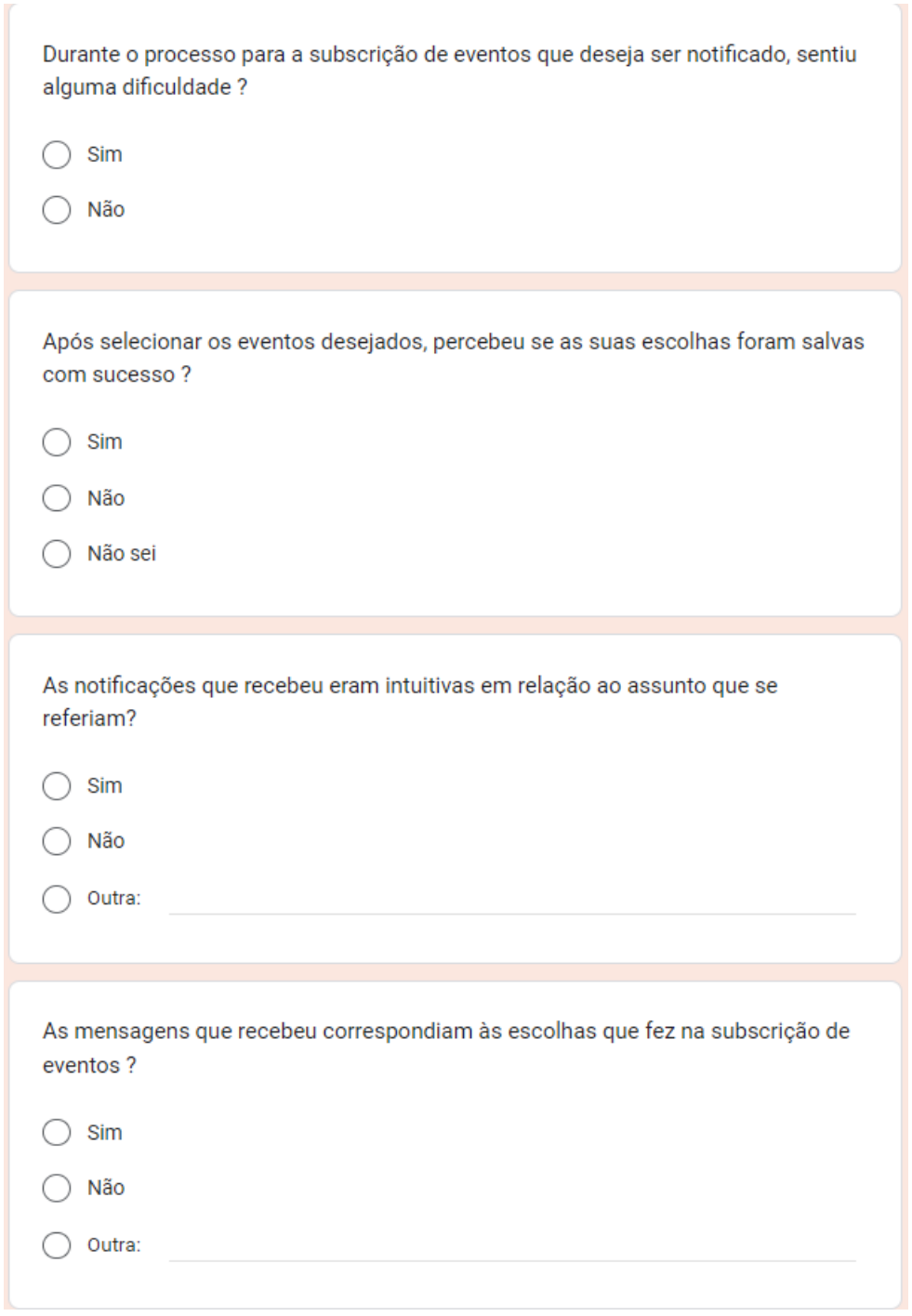


Figura 35 - Questionário Secção da Usabilidade

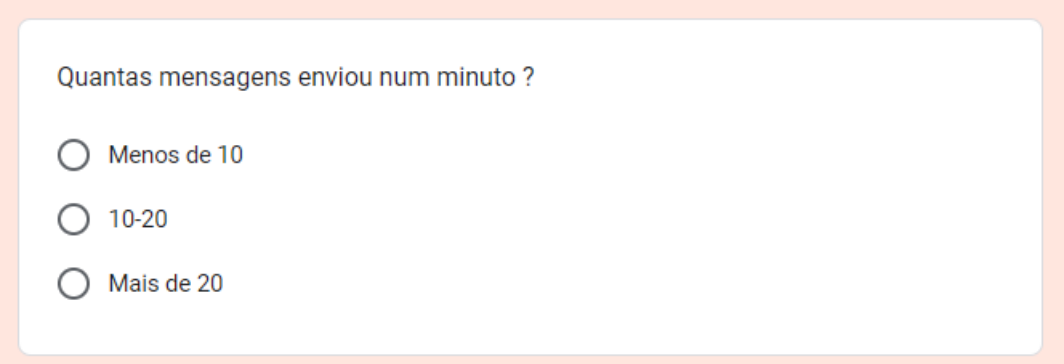


Figura 36 – Questionário Secção do Fluxo de Mensagens

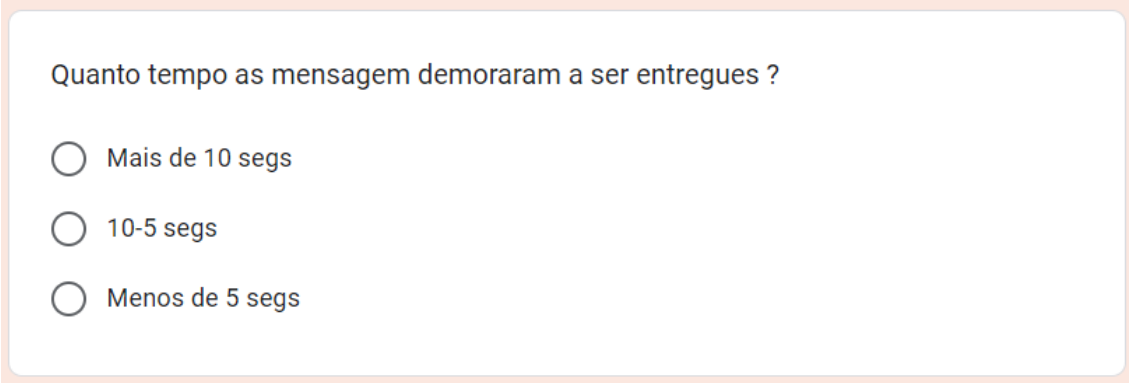


Figura 37 – Questionário Secção da Latência

Figura 38 - Questionário Secção da Confiabilidade

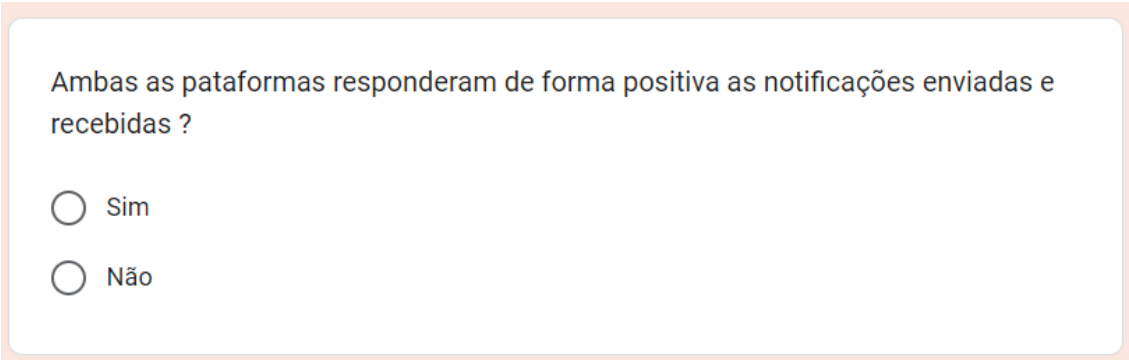
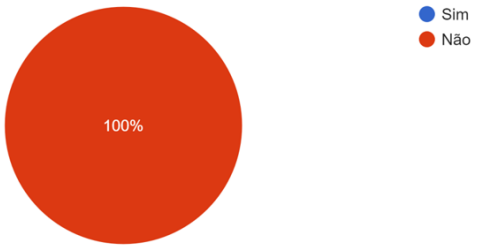


Figura 39 - Questionário Secção da Compatibilidade

Anexo B - Questionário Sistema de Notificações Drive - Respostas

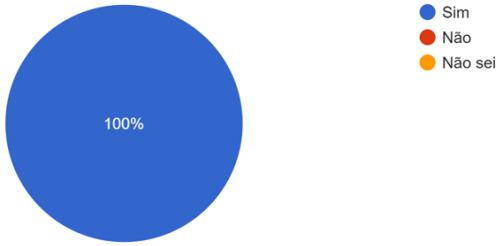
Durante o processo para a subscrição de eventos que deseja ser notificado, sentiu alguma dificuldade ?

3 respostas



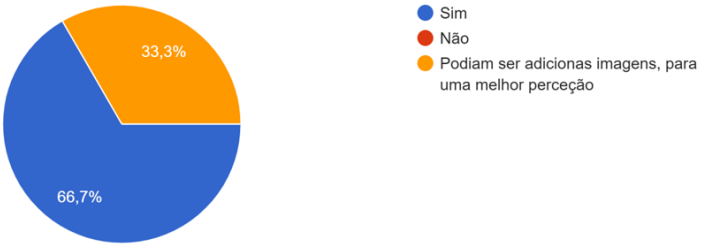
Após selecionar os eventos desejadas, você percebeu se as suas esolhas foram salvas com sucesso ?

3 respostas



As notificações que você recebeu eram intuitivas em relação ao assunto que se referiam?

3 respostas



As mensagens que você recebeu correspondiam às escolhas que fez na subscrição de eventos ?

3 respostas

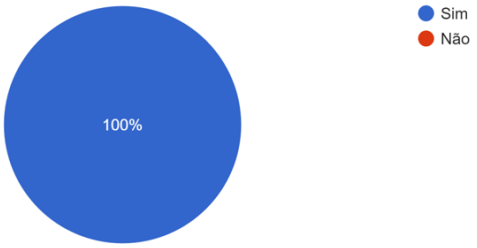


Figura 40 – Questionário Secção da Usabilidade - Respostas

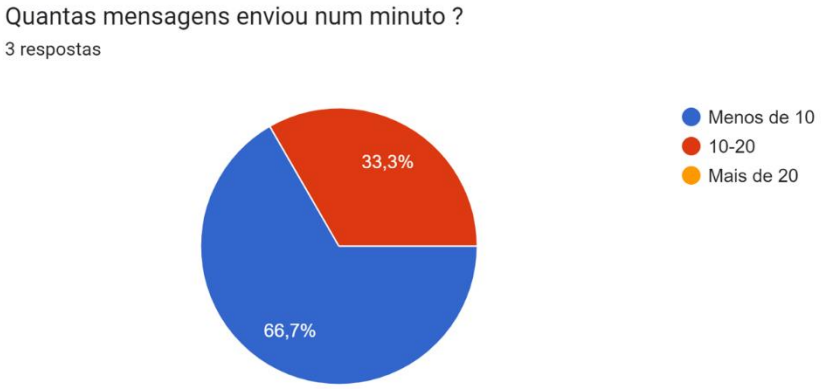


Figura 41 – Questionário Secção do Fluxo de Mensagens - Respostas

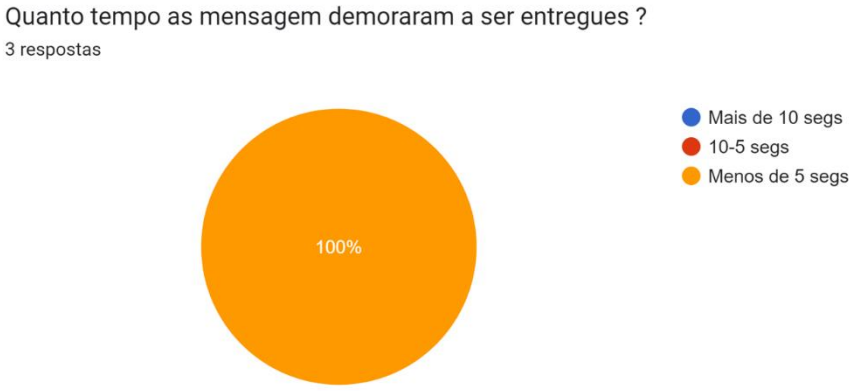


Figura 42 – Questionário Secção da Latência - Respostas

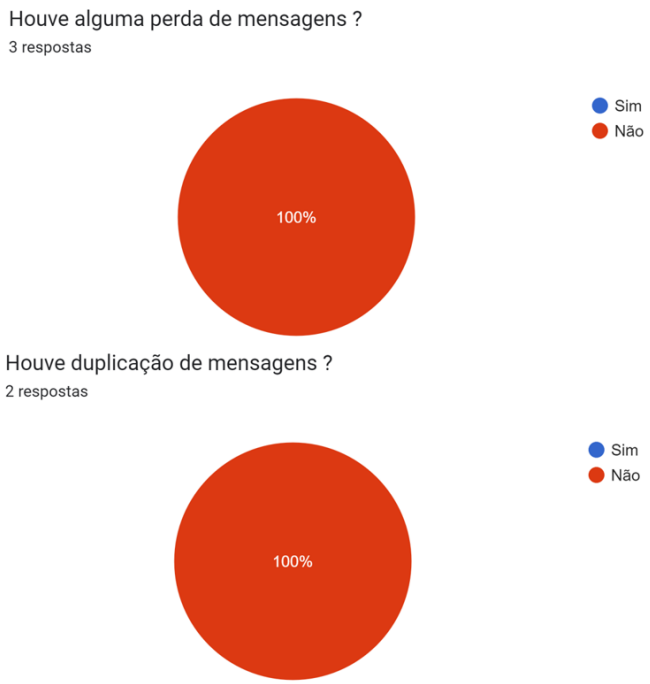


Figura 43 - Questionário Secção da Confiabilidade - Respostas

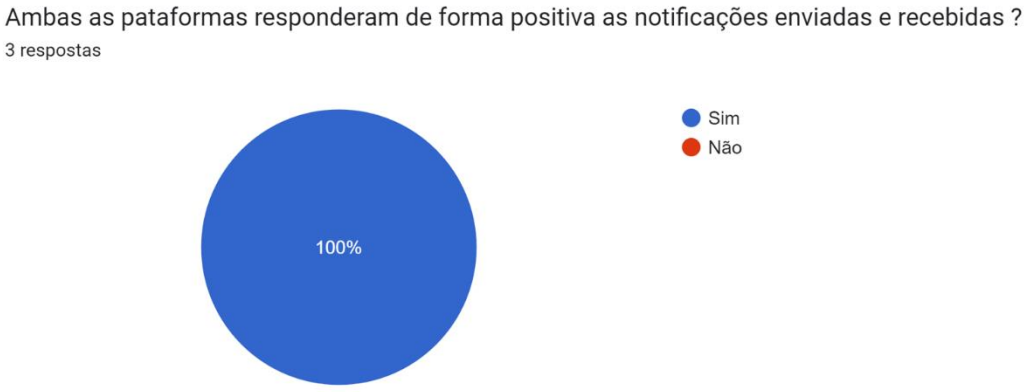


Figura 44 - Questionário Secção da Compatibilidade - Respostas