Arquitectura de Sistemas
Trabalho Prático

Ivo Gomes
a10700@alunos.ipca.pt

Rui Ferreira
19998@alunos.ipca.pt

Tiago Oliveira
a21585@alunos.ipca.pt

Mestrado em Engenharia Informática
Escola Superior de Tecnologia
Instituto Politécnico do Cávado e do Ave

Janeiro, 2022

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O trabalho prático abordado no presente relatório foi desenvolvido no âmbito da unidade curricular Arquitectura de Sistemas da oitava edição do Mestrado em Engenharia Informática da Escola Superior de Tecnologia do Instituto Politécnico do Cávado e do Ave.

O referido trabalho prático incidiu na análise, desenho e desenvolvimento de um sistema distribuído cujo tema primordial é a micro-mobilidade. Para concretizar o sistema, desenvolveu-se um conjunto de componentes, com diferentes propósitos funcionais, através do uso de múltiplos padrões, tecnologias e ferramentas.

O desenvolvimento do documento aborda o ambiente de desenvolvimento, a análise do problema, a arquitectura técnica e disponibilização da solução e os resultados obtidos.

Capítulo 2

Desenvolvimento

2.1 Ambiente de desenvolvimento

O trabalho prático e correspondente relatório foram desenvolvidos com recurso às tecnologias e ferramentas abaixo listadas:

- Visual Studio Code 1.63.2;
- Node.js 17.3.0;
- Python 3.9.9;
- MongoDB 5.1.1;
- PostgreSQL 13;
- Mosquitto 2.0.13;
- Node-RED 2.1.4;
- GeoServer 2.19.4;
- PostGIS 3.1;
- Docker 20.10.11;
- Postman 9.8.2;
- Git 2.34.1;
- LaTeX.

2.2 Análise do problema

O prelúdio do desenvolvimento do trabalho prático começou com a análise do problema a resolver. Para tal, procedeu-se à interpretação do respectivo enunciado, com vista a identificar os requisitos funcionais e não-funcionais da solução. Foram ainda definidos os devidos constrangimentos e pressupostos.

2.2.1 Requisitos funcionais

A tabela 1 descreve os requisitos funcionais da solução.

Identificador	Descrição
RF01	O sistema deverá suportar dois perfis de utilizador: gestores e clientes.
RF02	O gestor deverá conseguir configurar diferentes tipologias de veículos.
RF03	O gestor deverá ser capaz de definir o preçário para cada tipologia de veículo.
RF04	O gestor deverá conseguir gerir (adicionar, remover, etc.) a frota de veículos.
RF05	O gestor deverá ser capaz de consultar o estado e histórico de utilização do sistema.
RF06	Um novo cliente deverá conseguir registar-se no sistema.
RF07	A idade e género do cliente deverá ser inferido através do carregamento de uma fotografia.
RF08	O cliente deverá ter a possibilidade de carregar o saldo associado ao seu perfil.
RF09	O cliente deverá conseguir consultar as tipologias de veículos e respectivos preçários.
RF10	O sistema deverá permitir que clientes encontrem o(s) veículo(s) mais próximo(s) de si.
RF11	O sistema deverá permitir o aluguer de veículos por clientes.
RF12	O sistema deverá calcular o trajecto mais rápido entre a origem e o destino da viagem.
RF13	O custo de uma viagem deverá ser calculado com base no período de utilização.
RF14	O cliente deverá conseguir consultar as viagens efectuadas e respectivos detalhes.
RF15	O cliente deverá conseguir consultar o seu perfil e respectivos detalhes.

Tabela 1: Requisitos funcionais da solução

2.2.2 Requisitos não-funcionais

A tabela 2 define os requisitos não-funcionais da solução.

Identificador	Descrição
RNF01	O cliente deverá encontrar-se autenticado no sistema para que goze das suas funcionalidades.
RNF02	O sistema deverá ser constituído por serviços com propósitos funcionais bem definidos.
RNF03	A API do sistema deverá estar documentada através da especificação Open API.
RNF04	Deverá ser assegurada a resiliência, escalabilidade e disponibilidade do sistema.
RNF05	O sistema deverá ser disponibilizado através da orquestração de contentores.
RNF06	Os contentores deverão ser concebidos com recurso a imagens oficiais.

Tabela 2: Requisitos não-funcionais da solução

2.2.3 Constrangimentos e pressupostos

A tabela 3 identifica os constrangimentos e pressupostos da solução.

Identificador	Descrição
CP01	Para realizar uma viagem, o cliente deverá dispor do saldo necessário.
CP02	O cliente apenas poderá realizar viagens se tiver, pelo menos, 16 anos de idade.
CP03	Se o cliente ficar sem saldo no decorrer de uma viagem, o veículo deverá parar após um minuto.

Tabela 3: Constrangimentos e pressupostos da solução

2.3 Desenho da solução

O desenho da solução envolveu fundamentalmente duas etapas: a elaboração da arquitectura do sistema e a arquitectura dos serviços. Esta secção discute ambas.

2.3.1 Arquitectura do sistema

A solução é baseada num sistema distribuído que segue uma arquitectura orientada a micro serviços. Neste, existe um conjunto de serviços vagamente acoplados, com propósitos funcionais bem definidos, que comunicam entre si, com vista a concretizar diferentes casos de uso.

A figura 1 ilustra um diagrama de componentes que especifica a arquitectura do sistema. Em seguida, é apresentada uma síntese de cada um dos componentes.

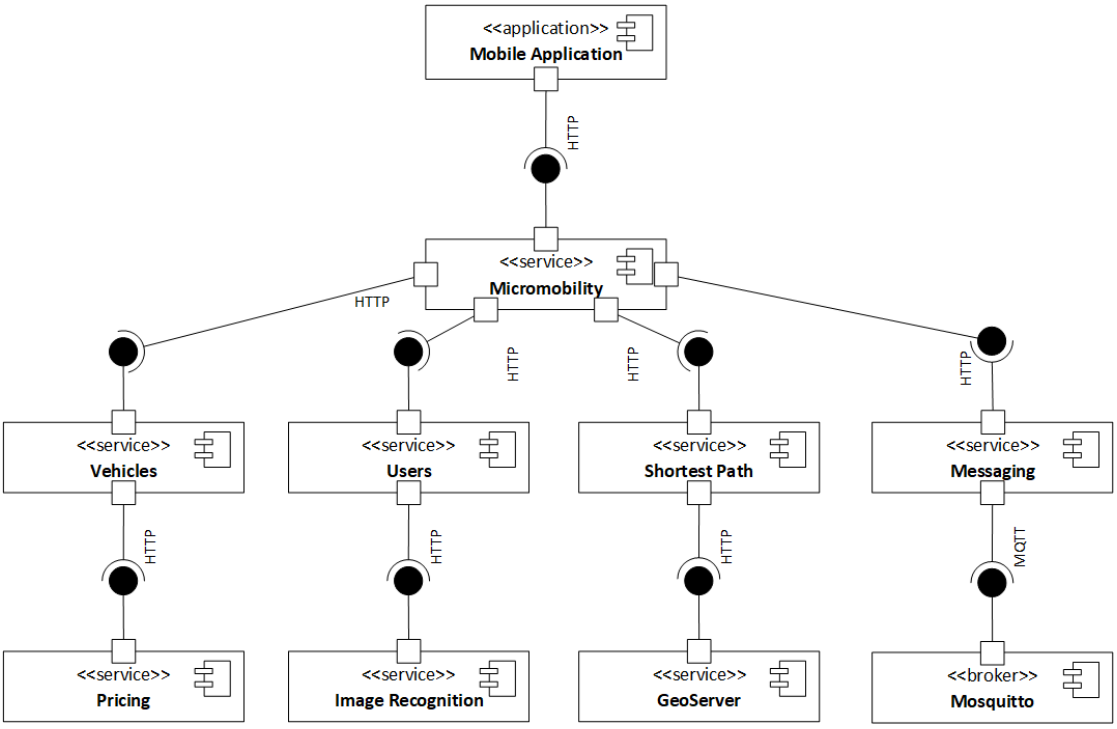

Figura 1: Arquitectura do sistema

Pricing

Serviço orientado à gestão e armazenamento de tipologias de veículos e preçários. É independente na medida em que não depende de nenhum outro serviço.

Image Recognition

Como o nome sugere, trata-se de um serviço focado no reconhecimento de imagem. Através deste, com recurso a inteligência artificial, é possível estimar a idade e género de um determinado indivíduo. Não apresenta nenhuma dependência.

GeoServer

Serviço de código aberto cujo propósito é a partilha, processamento e edição de dados geoespaciais. Suporta múltiplos formatos de dados.

Mosquitto

Serviço de código aberto responsável por mediar mensagens entre processos através do protocolo MQTT. Considerando a sua natureza leve, possibilita uma integração eficiente dos serviços do sistema com os veículos associados.

Vehicles

Serviço que está a cargo de gerir e armazenar os veículos. Depende de consultas ao serviço de gestão e armazenamento de tipologias de veículos e preçários.

Users

Serviço capaz de gerir e armazenar os utilizadores, independentemente de estes serem gestores ou clientes. Depende do serviço de reconhecimento de imagem para, no momento de registo de um cliente, estimar o seu género e validar a sua idade.

Shortest Path

Serviço que se apresenta como uma abstracção ao serviço de processamento de dados geoespaciais. Permite, essencialmente, calcular o caminho mais curto entre dois pontos geográficos.

Messaging

Serviço cujo intuito é interpretar as mensagens provenientes dos veículos e desencadear as operações adequadas nos serviços. Para tal, subscreve os tópicos existentes no serviço capaz de mediar eventos, previamente descrito.

Micromobility

Serviço incumbido de definir as operações de alto nível do sistema. Concretiza o comportamento funcional do sistema por meio da orquestração dos restantes serviços. Em termos conceptuais, este serviço assume o papel de gateway, isolando clientes (e, por consequência, utilizadores) dos restantes serviços.

Mobile Application

Aplicação móvel com a qual o utilizador interage para satisfazer os casos de uso previstos na etapa de análise do problema. Esta comunica exclusivamente com o serviço de micromobilidade. O componente encontra-se fora do âmbito do projecto, pelo que é considerado trabalho futuro.

2.3.2 Arquitectura dos serviços

Os serviços do sistema seguem uma arquitectura de referência. Pretende-se, com esta decisão, promover a uniformidade, velocidade de desenvolvimento e facilidade de manutenção da solução. A única excepção é o serviço de reconhecimento de imagem, dadas as suas especificidades técnicas.

Internamente, a arquitectura de referência é composta por quatro camadas: rotas, controladores, modelos e serviços. Cada uma das camadas, assim como respectivos tipos, têm responsabilidades claramente definidas. Uma determinada camada apenas poderá invocar a camada que se encontra imediatamente abaixo.

Adicionalmente, considerando as boas práticas das arquitecturas orientadas a micro serviços, cada um dos serviços tem a sua própria base de dados.

A figura 2 apresenta a arquitectura dos serviços. Depois, é exposta uma breve descrição de cada um dos seus componentes internos.

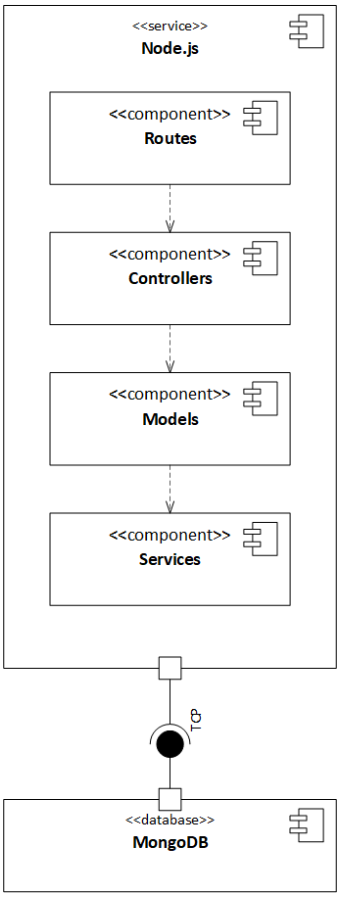

Figura 2: Arquitectura dos serviços

Routes

Camada de apresentação do serviço. Expõe um conjunto de rotas responsáveis por concretizar as funcionalidades dispostas no serviço. A camada implementa um conjunto boas práticas características do domínio dos micro serviços:

- Implementa a especificação REST e segue os princípios RESTful;
- Todas as rotas existentes encontram-se devidamente versionadas;
- Disponibiliza uma rota de monitorização que facilita eventuais automatismos inerentes à infraestrutura;
- Expõe um portal de documentação Swagger que descreve as rotas e modelos existentes no serviço;
- As respostas são enviadas no formato JSON. As mencionadas respostas são também comprimidas;

- Os pedidos, e respectivas respostas, estão sujeitas a um middleware dedicado a operações de logging;
- Emite, aquando das respostas, os códigos de estado HTTP adequados.

Controllers

Camada que define e reúne a lógica de negócio de alto nível do serviço. Seguindo um adequado isolamento e segmentação de responsabilidades, a camada que define as rotas, limita-se a invocar e a interpretar os resultados provenientes das rotinas presentes nesta camada. Por sua vez, esta camada invoca rotinas dispostas na camada imediatamente abaixo: a camada de modelos.

Models

Camada encarregue de definir e abstrair o modelo de dados utilizado na base de dados. Além disto, esta camada concentra os detalhes de execução das operações CRUD existentes no serviço, através de recursos definidos, configurados e expostos pela camada de mais baixo nível: a camada de serviços.

Services

Camada que estabelece a comunicação com a base de dados e restantes serviços do sistema. Esta camada faz uso extensivo de um ficheiro de configuração e define várias rotinas que definem eventos de logging que incentivam a uma adequada monitorização da infraestrutura.

2.4 Disponibilização da solução

A solução é disponibilizada através da conteinerização do sistema, onde cada um dos serviços até então descritos, assim como o motor de base de dados que sustenta os serviços, corresponde a um contentor específico.

A opção de decompor a solução em diversos contentores, fundamenta-se no facto de esta decomposição permitir escalar individualmente cada um dos serviços, consoante necessidades particulares. Consequentemente, de um ponto de vista de infraestrutura, esta decisão sustenta adequadamente a escalabilidade e disponibilidade da solução, sobretudo quando aliada a sistemas de orquestração.

Adicionalmente, com vista a promover a resiliência da solução, os contentores do sistema estão configurados para serem automaticamente reiniciados, em caso de falha não tratada.

2.5 Resultados

2.5.1 Objectivos alcançados

Considerando os requisitos funcionais, requisitos não-funcionais, constrangimentos e pressupostos oriundos da etapa de análise do problema, assim como os artefactos resultantes da etapa de desenho e desenvolvimento da solução, conclui-se que a basta maioria dos objectivos foram alcançados e que o sistema satisfaz os casos de uso inicialmente propostos.

Não obstante, tratando-se de uma primeira versão do sistema, existe espaço para melhorias. Estas melhorias consistem, fundamentalmente, no refinamento de pormenores não-funcionais, tais como o uso de um sistema de orquestração de contentores avançado (Kubernetes) ou a implementação de uma camada de autenticação e autorização em todos o serviços.

2.5.2 Bonificações

No decorrer da construção da solução, fez-se por implementar boas práticas de desenvolvimento de software, além das dispostas no enunciado do trabalho prático. Examine-se algumas das práticas empregues:

1. O desenvolvimento da solução foi planeado e gerido através de metodologias de desenvolvimento ágeis, nomeadamente a framework Scrum, sendo que cada sprint teve a duração de uma semana, decorrida entre aulas dedicadas à unidade curricular;

2. As operações de versionamento do sistema, recorrem ao sistema de controlo de versões git, cujo repositório encontra-se alojado na plataforma GitHub. Os elementos do grupo de trabalho têm o devido acesso ao repositório e podem, consequentemente, trabalhar em formato colaborativo;

3. De modo a detectar erros e garantir a uniformidade do código fonte, fez-se uso de um linter, direccionado à linguagem de programação JavaScript, que se encontra convenientemente integrado nos serviços do sistema. O mesmo é apelidado de eslint e encontra-se ao abrigo de uma licença de código aberto;

4. Com vista a promover a manutenção e simplicidade do código fonte, fez-se por seguir, na medida do possível, os princípios DRY e KISS.

2.5.3 Desafios de implementação

O principal desafio de implementação da solução, foi o facto de alguns elementos do grupo de trabalho, terem pouco conhecimento ou experiência prévia na construção de sistemas distribuídos. Como expectável, este facto traduziu-se numa velocidade de implementação inicialmente reduzida, que foi progressivamente aumentando, à medida que o grupo de trabalho adquiria conhecimento, experiência e agilidade nas tecnologias e ferramentas utilizadas.

Capítulo 3

Conclusão

Concluído o trabalho prático e considerando a natureza empírica do mesmo, cremos que o trabalho desenvolvido foi essencial para a consolidação dos conhecimentos transmitidos pelos professores no decorrer das aulas dedicadas aos diversos temas abordados. Acreditamos também que o desenvolvimento deste permitiu amadurecer as nossas competências técnicas na área de arquitectura de sistemas. Tivemos ainda oportunidade de explorar este domínio através de tecnologias e ferramentas até então não empregues, o que foi uma experiência altamente gratificante.

Relativamente ao trabalho desenvolvido, julgamos que este cumpre competentemente os requisitos mencionados no desafio que nos foi lançado pelos professores. Gostaríamos de ter desenvolvido um trabalho mais aprofundado - particularmente na disponibilização da solução através do sistema de orquestração Kubernetes - contudo, a carga de trabalho característica de uma pós-graduação, aliada às nossas obrigações profissionais, limitou o tempo passível de ser despendido a trabalhar para cada uma das unidades curriculares.

Bibliografia

[1] *Node.js v17.3.0 Documentation*. OpenJS Foundation, 2022.
URL: `https://nodejs.org/docs/v17.3.0/api`.

[2] *Python 3.9.9 Documentation*. Python Software Foundation, 2022.
URL: `https://docs.python.org/3.9`.

[3] *The MongoDB 5.1 Manual*. MongoDB Inc, 2022.
URL: `https://docs.mongodb.com/v5.1`.

[4] *PostgreSQL 13.5 Documentation*.
The PostgreSQL Global Development Group, 2022.
URL: `https://www.postgresql.org/docs/13`.

[5] *Mosquitto Documentation*. Eclipse Foundation, 2022.
URL: `https://mosquitto.org/documentation`.

[6] *Node-RED Documentation*. OpenJS Foundation, 2022.
URL: `https://nodered.org/docs`.

[7] *GeoServer Documentation*. Open Source Geospatial Foundation, 2022.
URL: `https://docs.geoserver.org`.

[8] *Docker Reference Documentation*. Docker Inc, 2022.
URL: `https://docs.docker.com/reference`.

[9] *Express 4.x API Reference*. OpenJS Foundation, 2022.
URL: `https://expressjs.com/en/4x/api`.

[10] *Mongoose v6.1.1 API Docs*. Automattic, 2022.
URL: `https://mongoosejs.com/docs/api`.

[11] Ó. Ribeiro e B. Lima, *Arquitectura de Sistemas*.
Instituto Politécnico do Cávado e do Ave, 2022.