UNIVERSIDADE
LUSÓFONA

Backend BestRide

Trabalho Final de curso

Relatório Intercalar 1º Semestre

Ricardo Faia 21904553

Ângelo Bernardes 21904536

Nome do Orientador: Professor Rui Ribeiro

Trabalho Final de Curso | LEI | Data : 28 de Novembro 2021

www.ulusofona.pt

Direitos de cópia

Resumo

O negócio de Tuk Tuk's está em constante crescimento e também em expansão para diversas cidades deste Portugal. Com esta expansão apareceu a necessidade de organizar o negócio e é esse o principal objetivo desta aplicação chamada BestRide, A aplicação tem como outros objetivos, como por exemplo, garantir a pessoas que não conhecem nada de uma certa cidade marcarem viagens por essas cidades por guias que as conhecem. Com esta aplicação há também uma avaliação e constante mudanças nos roteiros e guias, o que permite oferecer o melhor serviço e conforto para todos os envolvidos, o que incluí as empresas, turistas e guias.

Abstract

The business of Tuk Tuk's is constantly growing and in expansion in several cities in Portugal. With that expansion appeared the need to organize the business and that is the primary goal of this application called BestRide. The app has as other goals, for example, assure to people who don't know anything about the city book trips through these towns with guides who know them. With this app have too a evaluation and constant changes in scripts and guides, what allows offer the better services and comfort for everyone, what includes companies, tourist and guides.

Índice

Lista de Figuras

1 Identificação do Problema

Com o crescimento do turismo em Portugal e o crescente uso de TukTuk's começou a aparecer alguma desorganização no controlo de TukTuk's e também a desregulamentação dos condutores dos mesmos levou ao aparecimento de condutores não qualificados para a tarefa de guias. Apareceu também necessidade de organizar a marcação de viagens devido ao aparecimento da pandemia do Covid.

Com o desenvolvimento da tecnologia houve a hipotese de tratar esse problema, foi assim que surgiu a aplicação "BestRide" que nos foi desafiada a desenvolver na parte do Backend e gerenciamento do servidor.

A "BestRide" consiste na marcação de viagens, por parte do turista de qualquer parte do mundo/nacionalidade, para roteiros já existentes na aplicação e/ou criados pela comunidade de utilizadores. Haverá também uma organização dos condutores dos TukTuk's, esses condutores podem ou não pertencer a empresas que se registam na aplicação.

No decorrer deste desenvolvimento conseguimos ir de encontro a parte do problema fazendo toda a estrutura necessária para a criação de viagens e toda a envolvente da mesma como é a introdução de carros para se efetuar a viagem e também para a criação tanto de empresas como de condutores. Da parte do cliente desenvolvemos toda a estrutura para a visualização de viagens e também para a avaliação das mesmas. Em termos de marcação de viagens propriamente dita não conseguimos proceder ao desenvolvimento ficando assim para um futuro desenvolvimento.

2 Viabilidade e Pertinência

Sendo os tuk tuk's um negócio em ascensão em Portugal, mas com alguma desorganização no negócio como é o caso de alguns condutores que não têm licença de guias turísticos, mas fazem os tours na mesma, surgiu a ideia de fazer a BestRide, esta aplicação irá automatizar, otimizar e organizar este negócio tanto da parte do profissional e empresas como da parte dos turistas.

Como é que há essa otimização? Para mostrar façamos os seguintes exemplos:

Considerando que um turista de nacionalidade inglesa está a planear uma vinda a Lisboa, durante esse planeamento ele pode ver na aplicação quais são os roteiros que pode efetuar em Lisboa e se lhe agradar pode proceder à marcação/reserva do roteiro a partir de casa e ainda antes de chegar a Lisboa. Visto que os guias são todos certificados o turista sabe que o guia que o acompanhar é um profissional certificado.

Considerando agora um condutor com licença de guia turístico que começar a fazer roteiros de Tuk Tuk, ele pode-se inscrever na aplicação, nessa inscrição o guia diz os horários que tem disponíveis, com esta simples inscrição ele pode começar a fazer os roteiros que quiser.

A aplicação tem também uma gestão de guias por parte de uma empresa, isto é a empresa pode gerir os seus profissionais de forma segura e sem problemas, o que facilita a gestão para uma empresa de, por exemplo, 10 empregados.

A aplicação terá margem para progressão para o futuro através da criação de roteiros personalizados por parte dos turistas e/ou por parte dos guias, formação aos guias turísticos e diversas ideias que nos foram apresentadas pela CEO do BestRide, Sónia Gomes, que durante muitos anos trabalhou neste negócio e viu todo o tipo de problemas impercetíveis aos turistas e até mesmo população em geral.

3 Levantamento e análise dos Requisitos

No que diz respeito a levantamento e análise de requisitos foi-nos pedido o desenvolvimento do nosso trabalho seguindo os seguintes tópicos: (para efeitos de simplicidade podemos considerar Turistas, Drivers ou Empresas o **Utilizador**)

- O Utilizador tem que conseguir registar-se, fazer login, fazer update da conta e ter a hipótese de eliminar a conta; (Requisito desenvolvido na sua totalidade)
- O Utilizador pode fazer login usando a rede social *Facebook* ou usando o *Google*; (Requisito foi desenvolvido parcialmente)
- O Utilizador tem que conseguir recuperar a password caso se esqueça da mesma; (Requisito desenvolvido na sua totalidade)
- O Utilizador tem que ter acesso aos seus dados pessoais; (Requisito desenvolvido na sua totalidade)
- A aplicação tem que adaptar o seu idioma com a escolha de idioma do utilizador; (Requisito desenvolvido na sua totalidade)
- As empresas podem criar veículos e roteiros; (Requisito desenvolvido na sua totalidade)
- Os dados dos veículos e dos roteiros têm que incluir a empresa que criou um dado roteiro ou veículo; (Requisito desenvolvido na sua totalidade)
- O Utilizador tem que saber todos os detalhes de um dado roteiro; (Requisito desenvolvido na sua totalidade)
- O utilizador pode realizar uma viagem de qualquer roteiro; (Requisito desenvolvido parcialmente)
- O utilizador tem que ter acesso ao trajeto da viagem que está a marcar; (Requisito desenvolvido na sua totalidade)
- O turista pode comentar uma viagem que efetuou assim como o condutor que fez a viagem; (Requisito desenvolvido na sua totalidade)

Requisitos para uma fase futura da aplicação aplicam-se também os seguintes requisitos:

- Implementação de um sistema de formação para os drivers, o que irá trazer um melhor serviço às viagens;
- Implementação de um sistema de ranking tanto de empresas como de condutor;
- O utilizador pode ter acesso ao caminho que está a fazer em tempo real;

4 Solução Desenvolvida

4.1 Linguagem e Framework

No desenvolvimento do Backend usamos o Python, uma linguagem de alto nivel que está cada vez mais presente no mundo da programação.

Do Python usamos a framework Django que utiliza o padrão MTV (Model-Template-View). Consiste na criação de models, que são objetos que têm o mesmo modelo de dados que as tabelas da base de dados. Depois da criação de models e da receção dos dados da base de dados procede-se à manipulação dos dados através das views que são chamados por url definidos para cada funçãos das views.

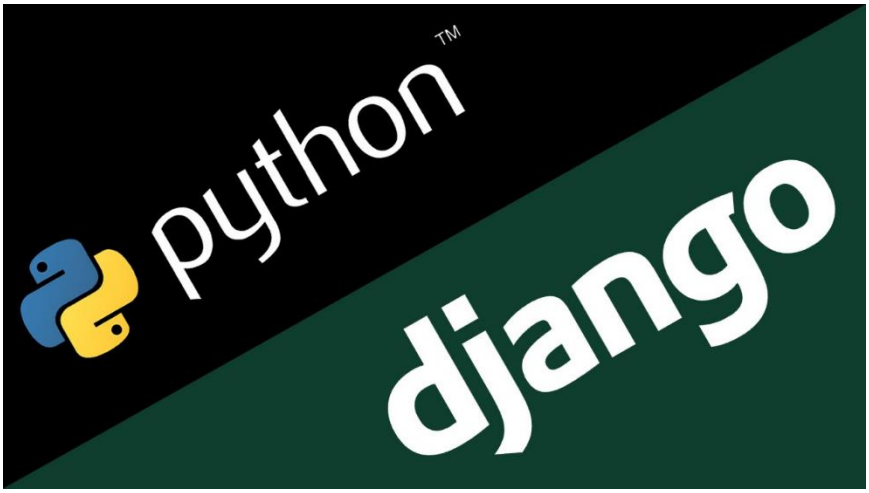

Figura 1: Python/Django

4.2 MySQL

A tecnologia usada na base de dados é o MySQL, que é um sistema de gestão da base de dados que utiliza a linguagem SQL como base para a sua interface.

Figura 2: MySQL

4.3 AWS

O servidor e a base de dados irá estar nos serviços da AWS utilizando diferentes web services fornecidos pela propria AWS.

Figura 3: Amazon Web Services

Os web services da AWS usados:

- **S3**, é um serviço que fornece armazenamento de objetos por meio de uma interface de serviço da web;
- **Cognito**, que serve para encriptar os dados mais pessoais dos utilizadores e assim dar uma segurança maior a esses mesmos dados;
- **RDS**, que nos dá o controlo da base dados e a respetiva manipulação dos dados, como a criação, alteração ou eliminação dos dados;
- **CodeCommit**, que é onde está situado o codigo desenvolvido;
- **CodePipeline**, consiste na compilação do codigo acessando-o graças ao CodeCommit e lança esse codigo para o servidor que faz com que o codigo possa ser acedido pela aplicação final;
- **Translate**, que vai tornar a aplicação internacional, ou seja, faz a tradução automática de algum texto para uma dada linguagem escolhida pelo utilizador;

4.4 Docker

O Docker é um servidor de automação de tarefas que usa a vitualização para executar aplicações através de espaços chamados “containers” para entregar o software em pacotes.

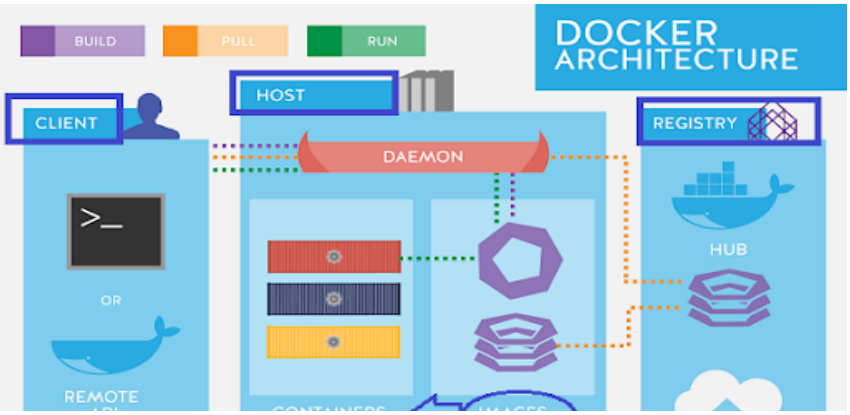

Figura 4: Arquitetura do Docker

Neste desenvolvimento usamos o Docker com a finalidade de preparar o ambiente/servidor onde vai correr o código, ou seja, o Docker ajuda-nos a criar um ambiente virtual pronto para correr o código como se fosse o nosso computador pessoal instalando no servidor tudo o que seja necessário para correr a aplicação.

4.5 Stripe

O sistema de pagamento tem que ser a parte mais segura e eficaz a solução que encontramos para esse problema do pagamento é o Stripe.

Stripe é um software que permite pagamentos entre negocios (neste caso é a BestRide) e clientes (neste caso são os utilizadores da APP que queiram fazer uma viagem). Esta tecmologia também atua muito na prevenção de fraudes bancarias, o que protege tanto a APP que está a vender os roteiros e os clientes que os querem fazer.

Figura 5: Stripe

4.6 Arquitetura da solução

Utilizando os serviços já referidos a Arquitetura proposta baseia-se muito em Cloud Computing, usando como base a AWS.

O código encontra-se escrito em Python, que usa, como já referido, diversos serviços da AWS e também do Stripe. Esse código vai ficar armazenado num serviço da aws que é o CodeCommit. Depois de o código estar “commited” e desenvolvido passa para a fase do codePipeline.

O codePipeline consiste em 3 fases, o ECR que é um serviço de registo de imagens de um determinado contentor, o CodeBuild que compila o código e o CodeDeploy que disponibiliza a ultima versão do código para a Internet em geral através de um IP.

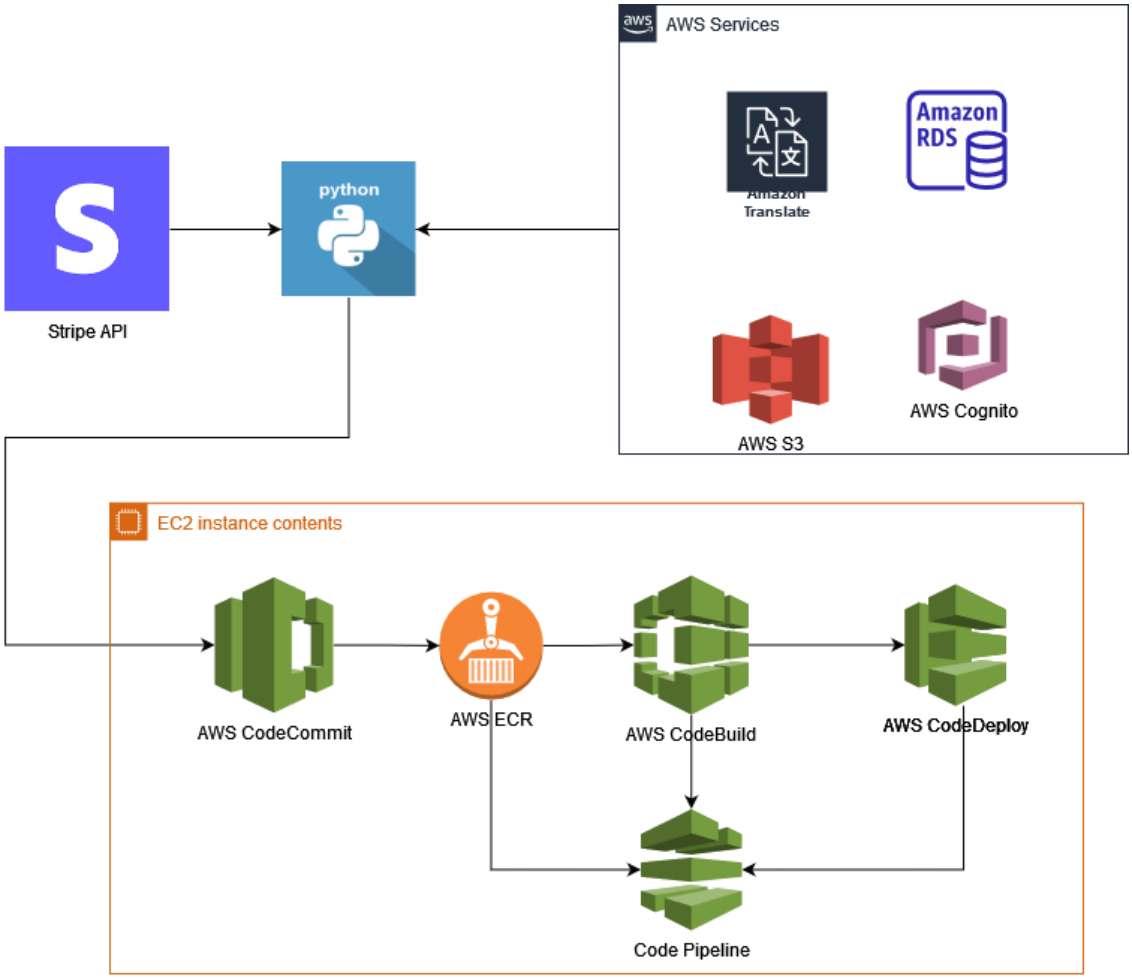

Figura 6: Arquitetura do Backend da aplicação

4.7 Desenho da Base de Dados

A solução final para a base de dados deste projeto é a seguinte:

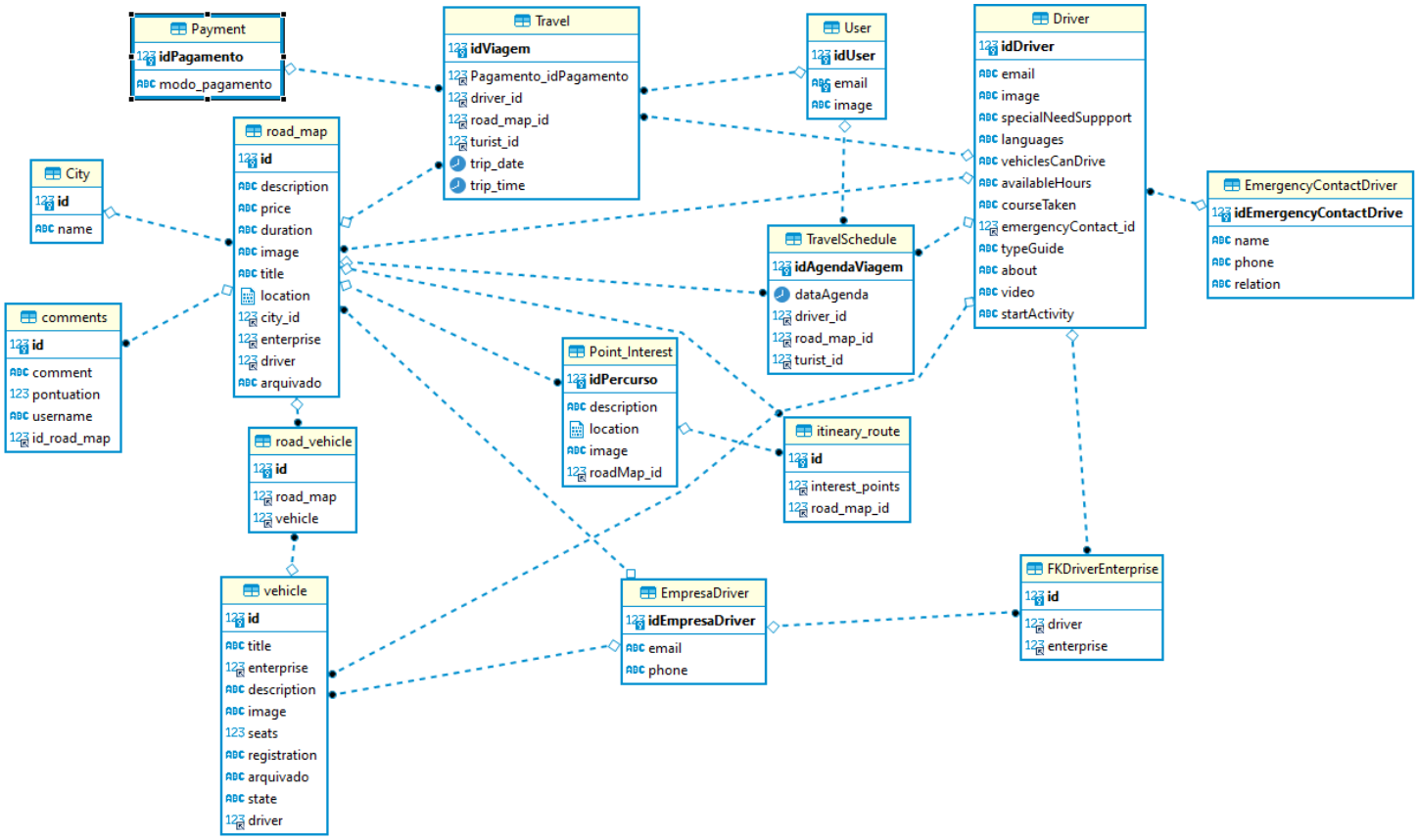

Figura 7: Desenho atual da Base de dados

Neste projeto usamos muito do conhecimento adquirido de diversas disciplinas do curso de LEI, são elas:

1. Engenharia de Requisitos e Teste
2. Engenharia de Software
3. Fundamentos de Programação
4. Linguagens de Programação I e II
5. Algoritmia e Estrutura de Dados
6. Base de Dados
7. Computação Distribuída
8. Sistemas de Informação na Nuvem

5 Benchmarking

Depois de uma análise ao mercado podemos concluir que a aplicação BestRide é inovadora e única na parte do turismo, o que garante uma grande vantagem para a aplicação. Encontra-se é aplicações que têm uma ideia idêntica, mas para serviços diferentes, é o exemplo da aplicação "*TripAdvisor*", "*AirBnB*", estas são aplicações que têm o propósito de apresentar pontos de interesses de determinadas cidades. As aplicações "*Tours & Travel*" e "*Walkbox – Roteiros Autoguiados*" oferecem roteiros sem guias para os turistas fazerem certos roteiros. Em comparação com estas aplicações a "*BestRide*" oferece roteiros com guia e de TukTuk, o que faz desta aplicação pioneira nos roteiros com guias e também com transportes.

Visto como uma aplicação inovadora e que fazia falta no turismo, a parceria com negócios que tenham uma elevada influência no turismo, como a publicidade podem fazer do BestRide uma aplicação vencedora no seu ramo.

Com base na solução desenvolvida o BestRide está em vantagem pois será uma aplicação de fácil utilização e com uma elevada margem de progressão em termos de utilizadores pois podendo escolher roteiros criados e guiados por profissionais é vantajoso em relação às aplicações já apresentadas, pois fazer uma viagem com um guia é sempre uma experiência mais enriquecedora. O BestRide tem a vantagem de não ter que ser usado no local, ou seja, um utilizador pode escolher uma determinada cidade que vai visitar e marcar um tour para uma determinada data no conforto da sua casa, mas também o pode fazer minutos antes de ir fazer o tour.

Figura 8: AirBnB

Figura 9: TripAdvisor

6 Método e planeamento

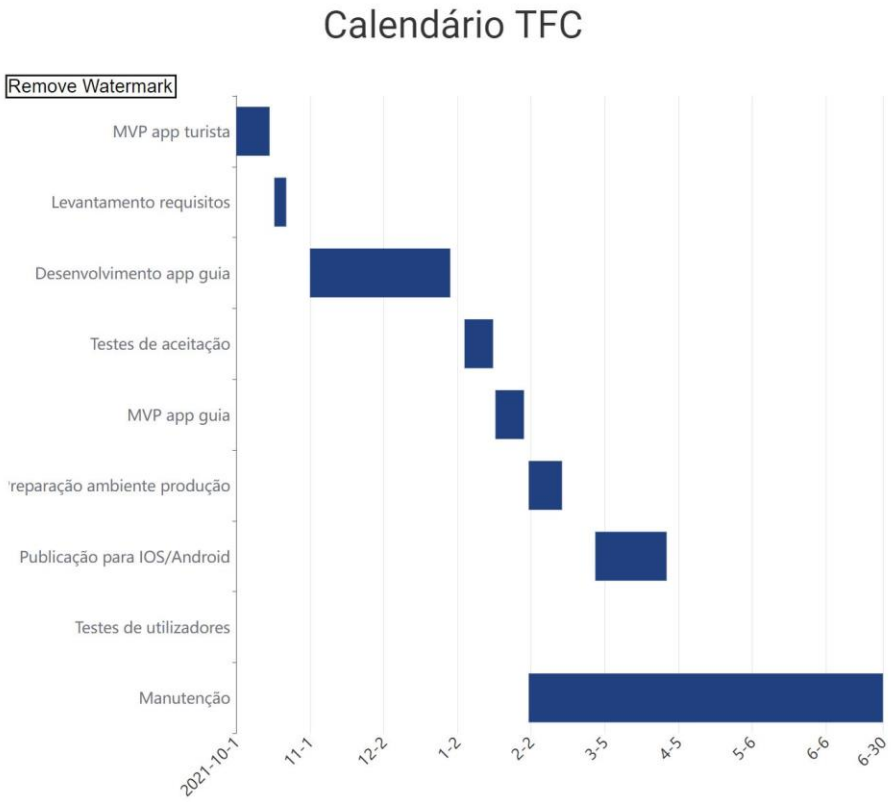

Figura 10: Calendário TFC

O método de trabalho para o desenvolvimento da aplicação foi de uma comunicação ativa com o cliente e também com os colegas do frontend.

Da nossa parte durante o desenvolvimento do backend optamos pela divisão de tarefas, o que facilitou algumas dificuldades que aparecessem pois não teríamos nada por fazer sem ser aquela tarefas, mas mantivemos também uma política de entreajuda entre os 2.

No decorrer deste trabalho existiram alguns atrasos, nomeadamente no desenvolvimento da app do guia, com esse atraso o restante do calendário ficou comprometido, principalmente no MVP do Guia. Em suma, podemos concluir que existiu alguma dificuldade em cumprir as datas tendo em conta o calendário previsto.

7 Resultados

O backend da aplicação BestRide tem o objetivo de tratar os dados, que são passados pelo/para frontend, basicamente esses dados do frontend são escritas ou leituras na base de dados. A comunicação entre o frontend e o backend é sempre feita através de Web Services e os dados são sempre passados através de JSON.

A escrita de dados é feita através do método de POST e é enviado um JSON para o backend, consoante o Web Service que é chamado. O backend faz a leitura do JSON e introduz os dados na base de dados.

A leitura de dados é feita através do método GET e é enviado para o frontend um JSON que terá os dados consoante o Web Service chamado. O backend basicamente faz a leitura dos dados da base de dados e envia os mesmo para o frontend.

Há também a web services que permitem eliminar dados da base de dados usando o método DELETE, neste caso só é preciso enviar um parâmetro com a chamada do web service.

Tendo em conta os requisitos levantados criámos diversos Web Services para o satisfazer os pedidos. No anexo “Manual de utilização” iremos mostrar todos os Web Services criados da nossa parte e o objetivo de cada um deles.

Conclusão e trabalhos futuros

Glossário

LEI	Licenciatura em Engenharia Informática
TFC	Trabalho Final de Curso
AWS	Amazon Web Services
MVP	Minimum Viable Product (Produto Mínimo Viável)
JSON	JavaScript Object Notation

Web Services AWS usados neste projeto (links):

- **S3** - https://aws.amazon.com/s3/
- **Cognito** - https://aws.amazon.com/cognito/
- **RDS** - https://aws.amazon.com/rds/
- **CodeCommit** - https://aws.amazon.com/codecommit/
- **CodePipeline** - https://aws.amazon.com/codepipeline/
- **Translate** - https://aws.amazon.com/translate/

Manual de Utilização

Web Services utilizados:

Login:

Recebe como argumento JSON com o username e a password;

Envia em JSON os dados do User (Turista);

GetUser:

Recebe como argumento o token do user;

Envia em JSON os dados do User (Turista);

RecoverUser:

Recebe como argumento JSON com o email;

Envia em JSON a reposta se é possível efetuar a recuperação;

UpdateUser:

Recebe como argumento o token do user;

Envia em JSON os dados do User (Turista);

ChangePassword:

Recebe como argumento um JSON com a password antiga, a nova e o token do user;

Envia em JSON com a resposta se foi ou não possível mudar a password;

SaveUser:

Recebe como argumento um JSON com o id do user, o email e a imagem;

Envia em JSON os dados do User (Turista);

ConfirmRecoverUser:

Recebe como argumento um JSON com o email, o código (que foi enviado por e-mail) e a password nova;

Envia em JSON se foi possível efetuar a recuperação com ou sem sucesso;

VerifyAccount:

Recebe como argumento um JSON com o email e o código que foi enviado para verificar a conta apos o login;

Envia em JSON os dados do User (Turista);

ResendCode:

Recebe como argumento um JSON com email;

Envia em JSON a resposta do sucesso da operação

CancelAccount:

Recebe como argumento um JSON com o token do user;

Envia em JSON como sucesso ou insucesso da operação;

Translate:

Recebe como argumento um JSON com o texto e com a linguagem de output;

Envia em JSON com o texto traduzido para determinada lingua;

ShowRoadVehicles

Recebe como argumento o id do roteiro;

Envia em JSON os dados da(s) viatura(s);

ShowRoadMap:

Recebe como argumento um JSON com a distância máxima de uma determinada latitude e longitude (ambos os argumentos passados por JSON);

Envia em JSON os dados do(s) roteiros que cumpram a distância máxima passada no argumento;

ShowInterestPoints:

Não recebe argumentos;

Envia em JSON com os dados de todos os pontos de interesse existentes de um determinado roteiro;

ShowRoadMapsCity:

Recebe como argumento a cidade da qual se pretende os roteiros;

Envia em JSON os dados de todos os roteiros dessa cidade;

GetRoadMapsByEnterprise:

Recebe como argumento o id da empresa;

Envia em JSON os dados de todos os roteiros criados pela empresa;

GetRoadMapsById:

Recebe como argumento o id do roteiro;

Envia em JSON os dados do roteiro;

CreateRoute:
Recebe como argumento um JSON com os dados do roteiro;
Envia em JSON os dados do roteiro (incluindo o id do roteiro);

DeleteRoute:
Recebe como argumento o id do roteiro;
Envia em JSON o sucesso ou insucesso da operação;

GetVehicle:
Não recebe argumentos;
Envia em JSON os dados de todos os veículos;

PostVehicle:
Recebe como argumento em JSON os dados do veículo;
Envia em JSON os dados do veículo (incluindo o id do veículo);

DeleteVehicle:
Recebe como argumento o id do veículo;
Envia em JSON o sucesso ou insucesso da operação;

GetVehicleByEnterprise:
Recebe como argumento o id da empresa;
Envia em JSON os dados do(s) veículo(s) da empresa;

GetVehicleById:
Recebe como argumento o id do veículo;
Envia em JSON os dados do veículo;

UpdateVehicle:
Recebe como argumento o id do veículo e um JSON com os dados do veículo;
Envia em JSON os dados atualizados do veículo;

GetComments:
Recebe como argumento o id do comentário;
Envia em JSON os dados do comentário;

PostComments:
Recebe como argumento um JSON com os dados do cometário;
Envia em JSON os dados do comentário (incluindo o id do comentario);

GetAverageComments:
Recebe como argumento o id do roteiro;
Envia em JSON a média de todos os comentários de um determinado roteiro;

GetTravels:
Não recebe argumentos;
Envia em JSON com todas as viagens existentes;

Travels:
Recebe como argumento o id do turista;
Envia em JSON os dados de todas as viagens do turista;

CreateTravel:
Recebe como argumento um JSON com os dados de uma nova viagem;
Envia em JSON os dados da viagem (incluindo o id da viagem);

CreatePointInterest:
Recebe como argumento um JSON com os dados de um novo ponto de interesse;
Envia em JSON os dados do ponto de interesse (incluindo o id do ponto de interesse);

GetPointInterest:
Não recebe argumentos;
Envia em JSON os dados de todos os pontos de interesse;

MakePayment:
Recebe como argumento um JSON com o preço a pagar e com o token do turista;
Envia em JSON o sucesso ou insucesso da operação;

UploadImage:
Recebe como argumento um JSON com o nome do ficheiro;
Envia o sucesso ou insucesso da operação;

LoginDriver:

Recebe como argumento JSON com o username e a password;

Envia em JSON os dados do condutor;

GetDriver:

Recebe como argumento o token do condutor;

Envia em JSON os dados do condutor;

RecoverDriver:

Recebe como argumento JSON com o email;

Envia em JSON a reposta se é possível efetuar a recuperação;

UpdateDriver:

Recebe como argumento o token do condutor;

Envia em JSON os dados do condutor;

ChangePasswordDriver:

Recebe como argumento um JSON com a password antiga, a nova e o token do condutor;

Envia em JSON com a resposta se foi ou não possível mudar a password;

SaveDriver:

Recebe como argumento um JSON com o id do condutor, o email e a imagem;

Envia em JSON os dados do condutor;

ConfirmRecoverDriver:

Recebe como argumento um JSON com o email, o código (que foi enviado por e-mail) e a password nova;

Envia em JSON se foi possível efetuar a recuperação com ou sem sucesso;

VerifyAccountDriver:

Recebe como argumento um JSON com o email e o código que foi enviado para verificar a conta apos o login;

Envia em JSON os dados do condutor;

ResendCodeDriver:

Recebe como argumento um JSON com email;

Envia em JSON a resposta do sucesso da operação

CancelAccountDriver:

Recebe como argumento um JSON com o token do condutor;

Envia em JSON como sucesso ou insucesso da operação;

PostEmergencyContact:

Recebe como argumento um JSON com os dados do contacto de emergência;

Envia em JSON os dados do contacto de emergência;

PostFKDriverEnterprise:

Recebe como argumento um JSON com o id do condutor e da empresa;

Envia em JSON os dados do condutor e da empresa;

LoginEnterprise:

Recebe como argumento JSON com o username e a password;

Envia em JSON os dados da empresa;

GetDriverEnterprise:

Recebe como argumento o token da empresa;

Envia em JSON os dados da empresa;

RecoverDriverEnterprise:

Recebe como argumento JSON com o email;

Envia em JSON a reposta se é possível efetuar a recuperação;

UpdateDriverEnterprise:

Recebe como argumento o token da empresa;

Envia em JSON os dados da empresa;

ChangePasswordDriverEnterprise:

Recebe como argumento um JSON com a password antiga, a nova e o token da empresa;

Envia em JSON com a resposta se foi ou não possível mudar a password;

SaveDriverEnterprise:

Recebe como argumento um JSON com o id da empresa, o email e a imagem;

Envia em JSON os dados da empresa;

ConfirmRecoverDriverEnterprise:

Recebe como argumento um JSON com o email, o código (que foi enviado por e-mail) e a password nova;

Envia em JSON se foi possível efetuar a recuperação com ou sem sucesso;

VerifyAccountDriverEnterprise:

Recebe como argumento um JSON com o email e o código que foi enviado para verificar a conta apos o login;

Envia em JSON os dados da empresa;

ResendCode:

Recebe como argumento um JSON com email;

Envia em JSON a resposta do sucesso da operação

CancelAccount:

Recebe como argumento um JSON com o token da empresa;

Envia em JSON como sucesso ou insucesso da operação;