Projeto da Unidade Curricular

Arquiteturas de Sistema

Mobilidade Urbana

MEI

Jéssica Macedo a6835,

Fernando Correia a11199,

Andreia Oliveira a12153

2º Trabalho

Escola Superior de Tecnologia

Instituto Politécnico do Cávado e do Ave

Barcelos, 15 de Dezembro de 2019

Conteúdo

Lista de Tabelas

Introdução

O trabalho abordado no presente relatório foi desenvolvido no âmbito da unidade curricular Arquiteturas de Sistemas do mestrado em Engenharia de Sistemas Informáticos em Desenvolvimento de Aplicações. Tem como fundamental objetivo o desenvolvimento de um sistema distribuído que permite alugar veículos de mobilidade urbana, tendo por base uma API Restful que garante a integração entre a aplicação servidor e as várias aplicações cliente (um cliente agente, um cliente gestor, e um cliente dashboard).

Desenvolvimento

Problema a Resolver

Este capítulo aborda a descrição do problema e os seus objetivos.

Servidor

1. O sistema tem como objetivo agilizar o aluguer de veículos disponíveis, fornecendo:
    - informação sobre os veículos livres;
    - filtros para a localização dos veículos livres;
    - gestão dos dados de cliente;
    - registo do pagamento através de um saldo recarregável.
2. O servidor deverá contemplar a utilização de bases de dados onde toda a informação relacionada com o serviço disponibilizado será guardada;
3. Desenvolver um conjunto de serviços para garantir o acesso à informação da base de dados, de forma a responder aos pedidos dos diferentes clientes;

4. Disponibilizadar uma documentação (Open API) e descrição acerca dos testes realizados à utilização dos serviços;
5. Publicação num ambiente cloud dos diferentes serviços desenvolvidos;
6. Seguir uma arquitetura baseada em micro serviços – containers;
7. Utilizar uma Gateway para facilitar a integração dos vários micro serviços;
8. Disponibilizar um sistema integrado de logging global a todos os micro serviços.

Utilizadores

Neste trabalho estão presentes quatro tipos de utilizadores, dos quais são:

- **Utilizador não registado** - Trata-se de um utilizador sem qualquer registo na plataforma;
- **Cliente** - Trata-se de um utilizador previamente registado, sendo considerado um cliente (do serviço de aluguer de veículos). Tem as mesmas funcionalidades que um utilizador não registado e mais algumas para além deste.
- **Funcionário** - Trata-se de um funcionário da entidade responsável pela gestão dos veículos, que tem a responsabilidade de fiscalizar os estacionamentos.
- **Administrador** - Trata-se da entidade fiscalizadora da aplicação. Consulta métricas, valida registos e configura.

Funcionalidades dos Utilizadores

De seguida apresentamos as funcionalidades de cada utilizador:

1. **Utilizador não registado**
   (a) Permite obter informação dos lugares de estacionamento (latitude e longitude), capacidade, quantidade de veículos;
   (b) Permite registar-se e consequentemente autenticar-se na aplicação.

2. **Cliente**

   (a) Utilizador previamente registado

   (b) Permite obter informação dos lugares de estacionamento (latitude e longitude), capacidade, quantidade de veículos;

   (c) Permite pesquisar veículos detalhando o nome da rua ou raio de pesquisa

   (d) Consulta do saldo atual da conta

   (e) Fazer check-in do veículo (código veículo, método de aluguer [preço por minuto/pacotes de horas], hora inicio, preço estimado, código de aluguer)

   (f) Fazer check-out do veículo (hora fim, verifica posição estacionamento, cálculo aluguer)

   (g) Fazer consulta dos dados relativos ao aluguer ativo (tempo e custo até ao momento)

3. **Funcionário**

   (a) Registo de estacionamentos de veículos em locais impróprios

   (b) Notificar cliente de estacionamento impróprio

4. **Administrador**

   (a) Consultar dashboard com resumo dos dados e histórico de ocupação de lugares

   (b) Permitir a validação do pedido de registo de utilizadores

   (c) Configuração da localização dos lugares de estacionamento

   (d) Nice to have: envio de indicação aos clientes da aproximação do fim do saldo

Plano de Desenvolvimento

Esta secção apresenta os objetivos propostos para as próximas entregas.

Data	**Objetivos**
15/12/2019	Geração dos modelos de dados
5/01/2020	Geração dos serviços CRUD para os diferentes serviços
	Criação da documentação Swagger
	Criação da aplicação frontend em React
17/01/2020	Instalação do sistema em serviço cloud com o Heroku
	Continuação da criação da aplicação frontend

Tabela 1: Objetivos do Plano de Desenvolvimento

Modelo de Dados

Nesta secção apresentamos o modelo de dados definido.

1. **Vehicle**

   De seguida é apresentado o código para construir o schema do modelo de dados do Veículo.

```
1
2  var vehicleSchema = new Schema({
3      code: {
4          type: Number,
5          required: [true,'code of the vehicle']
6      },
7      description: {
8          type: String
9      },
10     place: [placeSchema]
11 });
```

2. **Client**

   De seguida é apresentado o código para construir o schema do modelo de dados do Cliente.

```
var clientSchema = new Schema({
    firstName: {
        type: String,
        required: 'first name of the person'
    },
    lastName: {
        type: String,
        required: 'last name of the person'
    },
    rentals: [rentalSchema],
    balance: {
        type: Number,
    },
    registerBy: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User'
    },
    Created_data: {
        type: Date,
        default: Date.now
    }
});
```

3. **User**

   De seguida é apresentado o código para construir o schema do modelo de dados do Utilizador.

```
var userSchema = new Schema({
    username: {
        type: String,
        unique: true,
        required: true
    },
    email: {
        type: String,
        unique: true,
        index: true,
        required: true
    },
     password: {
type: String,
  required: true,
select: false
    },
role:{
type: String,
 required: true,
default: 'client ,
enum: ['client','employee','admin']
}
});
```

4. **Rental**

   De seguida é apresentado o código para construir o schema do modelo de dados do Aluguer.

```
var rentalSchema = new Schema({
    startDate: {
        type: Date,
        // default: Date.now
    },
    endDate: {
        type: Date,
        // default: Date.now
    },
    status: {
            type: String,
            enum: ['confirmed', 'canceled'],
               default: ['confirmed']
    },
    price: {
        type: Number,
        required: true
    },
paymentMethod:{
type: String,
            enum: ['minutes', 'pack'],
             default: ['minutes']
},
    code: {
        type: Number,
        required: true
```

```
27      },
28      vehicle: [vehicleSchema]
29  });
```

5. **Place**

   De seguida é apresentado o código para construir o schema do modelo de dados do Lugar.

```
1  var placeSchema = new Schema({
2      location: {
3          type: String,
4          coordinates: [Number],
5          required: true
6      },
7      capacity: {
8          type: Number
9      },
10     quantity: {
11         type: Number,
12     }
13 });
```

Bibliografia

[1] Repositório, https://github.com/Knox316/MobilityProject

[2] React, https://reactjs.org/.

[3] Heroku, https://www.heroku.com/