Instituto Politécnico do Cávado e do Ave

Escola Superior de Tecnologia

Arquitetura de Sistemas

Mestrado em Engenharia Informática

2º Trabalho Prático

Mobilidade Urbana

Barcelos,15 de Dezembro de 2019

Jéssica Macedo a6835

Fernando Correia a11199

**Atualizações ao documento:**

|  |  |
| --- | --- |
| Alterações | Data |
| Criação do documento | 15/12/2019 |
| Alterações da 2ª entrega | 05/01/2020 |

**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).

**Descrição detalhada do problema a resolver**

**Servidor:**

O objetivo do sistema é agilizar o aluguer de veículos disponíveis, fornecendo:

· filtros para a localização dos veículos livres;

· gestão dos dados de cliente;

· registo do pagamento através de um saldo recarregável;

O servidor deverá contemplar a utilização de bases de dados onde toda a informação relacionada com o serviço disponibilizado será guardada.

Desenvolver um conjunto de serviços para garantir o acesso à informação da base de dados, de forma a responder aos pedidos dos diferentes clientes;

Disponibilizar uma documentação (Open API) e descrição acerca dos testes realizados à utilização dos serviços.

Publicação num ambiente cloud dos diferentes serviços desenvolvidos;

Seguir uma arquitetura baseada em micro serviços – containers

· a utilização de uma Gateway para facilitar a integração dos vários micro serviços;

· a disponibilização de 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 os dados.

**Funcionalidades dos Utilizadores**

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 logar-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 para o desenvolvimento da solução (objetivos para próximas entregas)**

**Até 15 Dezembro:**

1. Geração dos modelos de dados

**Até dia 5 Janeiro**

1. Geração dos serviços CRUD para os diferentes serviços
2. Criação da documentação Swagger
3. Início da criação de algumas funcionalidades da aplicação frontend em React<https://reactjs.org/>

**Até dia 17 Janeiro**

1. Continuação da criação da aplicação frontend
2. Instalação do sistema em serviço cloud com o Heroku<https://www.heroku.com/>
3. Instalação dos micro-serviços em Docker

**Modelo de dados**

**Vehicle**

var vehicleSchema = new Schema({

code: {

type: Number,

required: [true,'code of the vehicle’']

},

description: {

type: String

},

Place: {

type: mongoose.Schema.Types.ObjectId,

ref: ‘Place’

}

});

**Client**

var clientSchema = new Schema({

firstName: {

type: String,

required: 'first name of the person '

},

lastName: {

type: String,

required: 'last name of the person '

},

rentals: [{

type: mongoose.Schema.Types.ObjectId,

ref: ‘Rental’

}],

balance: {

type: Number

},

Created\_data: {

type: Date,

default: Date.now

}

});

**User**

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: [“guest”, “client”,”employee”,”admin”]

},

registeredBy: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User'

},

valid : {

type: Boolean

}

});

**Rental**

var rentalSchema = new Schema({

startDate: {

type: Date,

// default: Date.now

},

endDate: {

type: Date,

// default: Date.now

},

price: {

type: Number,

required: true

},

rentalMethod:{

type: String,

enum: ['minutes', 'pack'],

default: ['minutes']

},

code: {

type: Number,

required: true

},

vehicle: {

type: mongoose.Schema.Types.ObjectId,

ref: ‘Vehicle’

}

});

**Place**

var placeSchema = new Schema({

location: [{

type: String,

coordinates: [Number],

required: true

},

{

range: Number

}

}],

capacity: {

type: Number

},

quantity: {

type: Number,

}

});

Repositório GitHub: <https://github.com/Knox316/MobilityProject>

**2ª entrega:  05/janeiro**

* Ponto de situação

Para esta fase tínhamos definido:

1. Geração dos serviços CRUD para os diferentes serviços
2. Criação da documentação Swagger
3. Início da criação de algumas funcionalidades da aplicação frontend em React<https://reactjs.org/>

Conseguimos concluir com sucesso os três pontos que tínhamos definido. Geramos os serviços CRUD que nos foram propostos no enunciado do trabalho e que podem ser verificados na coleção POSTMAN do projeto *MobilityProject.postman\_collection.json.*

Criamos uma base de dados não-relacional com o Mongodb, que pode ser consultada com o link: *mongodb+srv://admin:admin@cluster0-krbnl.mongodb.net/MobilityProject?retryWrites=true&w=majority.*

Criamos a documentação Swagger que pode ser consultada em http://{host}:3000/api-docs.

Fizemos algumas alterações aos modelos de dados previamente criados, pois sentimos necessidade de alterar alguns pontos à medida que íamos construindo os serviços.

No schema Rental adicionamos os campos “finalCost”,”previewCost”, “timeSpent” e alteramos a estrutura para “start” e “end”, onde no primeiro indicamos a localização e data do checkin e no segundo os dados de checkout.

No schema “user” adicionamos os campos “waitValidation” (que vai indicar se ainda necessita de validação do administrador), “firstname” e “lastname”.

No entanto nesta fase não criamos ainda funcionalidades da aplicação front-end em React.

* Plano para próximas entregas

Para as próximas entregas, iremos focar-nos na aplicação frontend, de forma a utilizar os serviços criados e construir o interface. Se necessário iremos ajustar os serviços.

Vamos também dividir os serviços criados em diferentes micro serviços, para depois podermos criar diferentes containers em Docker e publicar no Heroku.

* Código desenvolvido até ao momento

O código pode ser visto no Repositório GitHub: <https://github.com/Knox316/MobilityProject>.