Sistema de Gestão de Viaturas de uma Instituição

João Barbosa	João Marques	João Reis	Nuno Machado
(A82044)	(A84684)	(A84671)	(A68702)
LCC	LCC	LCC	LCC

1 de junho de 2022

Resumo

Temos como principal objetivo, criar um sistema de gestão de viaturas da Instituto Politécnico de Bragança, em aplicação web de forma a otimizar todo o processo de requisição de viaturas para serviços de deslocação, substituindo toda a logística necessária em formato papel, tornando-a mais simples, intuitiva e rápida de processar. Se ainda for possível, tentaremos migrar a mesma aplicação web para uma aplicação mobile.

Conteúdo

Capítulo 1

Introdução

No âmbito da unidade curricular de Projeto foi-nos proposto o desenvolvimento de uma aplicação web para um sistema de gestão de viaturas da instituição de Bragança. Optamos pela sua construção através da linguagem MySQL para implementação da base de dados e da linguagem Java para implementação das funcionalidades que necessitamos para o funcionamento de aplicação. Utilizamos o framework Spring pois permite fazer a "ligação"do Back-end para o Front-end conectando-se à base de dados mencionada anteriormente. Iniciaremos pelo esboço do modelo conceptual de todo o projecto com as respetivas questões necessárias para o levantamento de requisitos que serão listados neste relatório. De seguida iremos assumir o modelo relacional produzindo todas as relações entre entidades/classes necessárias tendo em principal atenção o encapsulamento do mesmo.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Numa instituicão existe um conjunto de viaturas (com ou sem motorista) que podem ser requisitadas para um determinado serviço de deslocação. A ideia seria desenvolver uma aplicação web que permitisse efectuar um pedido de viatura para um determinado serviço (capacidade da viatura, dia de ida, dia de vinda, horas, local de destino). Esse pedido terá que ser validado em termos de disponibilidade pela própria aplicação atribuindo uma viatura ao serviço e, em termos de autorização, por parte de uma entidade superior. Caso o pedido seja aceite é inserido no calendário de requisições autorizadas. O sistema poderá ser otimizado com sugestões de boleias partilhadas (total ou parcialmente), com alternativas de viaturas com menor capacidade ou com horas próximas que permitiriam a requisição da viatura.

2.2 Especificação dos Requisitos

Após definidos os objetivos da nossa aplicação estabelecidos pela cliente, percebemos que teríamos essencialmente de priorizar a organização da aplicação guardando assim vários registos. Assim sendo fomos capazes de identificar os seguintes requisitos:

Veículo: a nossa aplicação terá vários veículos que serão reservados para a realização de viagens solicitados pelo utilizador. Existirão vários veículos registados distinguidos essencialmente pela sua matrícula onde serão gravadas várias características, sendo elas, a marca, o modelo, os quilómetros, o ano, a cor, os lugares, e o seu combustível. Esse veículo poderá ou não ter motorista atribuído e será possível verificar se está em utilização.

Utilizador: a nossa aplicação será gerida para um utilizador que terá a possibilidade de fazer reservas de veículos para fazer deslocações (viagem). Será identificado pelo id, tendo ainda como características o seu nome e apelido, e ainda um código.

Viagem: a principal "mecânica"da nossa app serve em grande parte para facilitar as deslocações. Essas deslocações que serão feitas através de veículos reservados pelos utilizadores serão identificadas pelo seu id, tendo como características o local de inicio, o local do fim, os kms iniciais, os kms finais, a hora de inicio e hora de fim, o nº de passageiros, e o seu motivo.

Gestor: Será a entidade responsável pela gestão dos veículos, será capaz de adicionar ou remover um veículo da app.

Secretaria: Será a entidade responsável pela confirmação e/ou validação das viagens solicitadas.

2.2.1 Entrevista à Cliente

1. Que informacoes é necessario manter sobre cada veiculo?

R: matricula, nº de lugares, escola a que pertence, marca, submarca, anos de vida, cor, tipo de combustível, tipo de carro (ver abaixo), com ou sem motorista.

2. E necessario criar um perfil de cada pessoa? Se sim, que informações deveriamos guardar sobre cada pessoa que reserva um carro?

R: o utilizador deve ser validado por LDAP, ou seja, deve usar o login da instituicao a que pertence e ao fazer a requisicao fica associado ao seu login. Mas isso pode ser visto depois. Neste momento devemos associar a cada requisicao apenas o nome e nº de funcionario que esta a fazer a requisicao.

3. Que informacoes é necessario obter ao final de cada viagem e ao inicio de cada viagem?

R: hora de inicio, local de inicio, nº de quilometros iniciais do carro, nº e nome de passageiros hora fim, local de fim, nº quilometros finais do carro.

4. Que informacoes deveriamos ter sobre cada viagem?

R: Para alem do destino, data e hora talvez o objectivo da viagem e para alem do que esta na questao anterior adicionar no final da viagem o feedback sobre o estado do carro ou de alguma ocorrencia a registar

5. Como e que podemos diferenciar os diferentes veiculos?

R: matricula tipo de carro: carro, jipe, autocarro, carrinha, etc

6. Sobre cada viagem, que informacao e que precisamos ter? Devemos manter o perfil da pessoa que reserva o veiculo para esse utilizador ir informando quantos lugares estao disponiveis?

R: ao fazer a requisicao indicar o nome e nº de passageiros, no inicio da viagem confirmar os passageiros

7. Que restricoes uma certa pessoa que queira reservar um veiculo para uma certa viagem pode ter? (Restricoes de horario, restricoes relativas a uma certa autorizacao necessaria) Só docentes podem reservar veiculos ou alunos tambem podem? Que diferenciacao havera entre diferentes docentes e funcionarios da instituicao

R: Os alunos nao podem reservar viaturas, apenas os funcionarios (docentes ou nao-docentes) o poderao fazer e isso vai ter que ser validado de alguma forma.

8. Como e que uma pessoa que queira juntar-se a uma certa viagem deve informar a pessoa que esta encarregue dessa viagem? deve ser guardado o meio de contacto da pessoa responsavel pela viagem e os passageiros interessadas na viagem devem entrar em contacto com essa pessoa?

R: SIM, deve entrar em contacto com a pessoa que requisitou o carro e se se consumar a boleia essa pessoa deve ser inserida pelo condutor como passageiro.

8.1. A pessoa que reserva a viagem fica responsavel de quem vai na viagem? Ou simplesmente a pessoa que reserva o veiculo para a certa viagem apenas fica responsavel pelo carro durante a viagem e as pessoas que se queiram juntar simplesmente "adicionam-se"

9. Caso o veiculo seja utilizado numa visita de estudo, que propriedades são interessantes registar que as diferem de uma viagem normal

R: nada

10. Quando se seleciona um horario de reserva de um veiculo, podem acontecer imprevistos que não possiblitem a utilizacao do automovel nos horarios que seguem, como um atraso. A aplicacao deve ser capaz de avisar ao utilizador seguinte que o horario que selecionou nao tem veiculos disponiveis?

R: Quando o condutor anterior sinalizar um atraso o condutor seguinte devia ser notificado.

11. Dada uma viagem concluida, a aplicação deve ser capaz de marcar um veiculo como livre para outra viagem?

R: Sim, condutor no final da viagem deve fazer o checkout.

12. No evento de um acidente como devemos processar a informacao?

R: no feedback final do condutor deve ficar registado o acidente

13. No evento de um acidente como é que a pessoa responsavel pela viagem e pelo veiculo devem atuar e quem e que devem avisar? Eles tem que avisar no momento do acidente e seriam informados de como proceder?

R: Como qualquer condutor ...

14. Quem fica responsavel pelas chaves dos veiculos ou pelo armazenamento dos veiculos na universidade?

R: A portaria

15. A aplicacao devera manter registo da condicao mecanica dos diferentes veiculos?

R: Os documentos de revisao do carro estão dentro do carro. A aplicação apenas servira para notificar necessidades de manutencao

16. Ha motoristas para autocarros que devam ser adicionados ao sistema?

R: Ha veiculos com motorista e veiculos sem motorista. Ambos são passiveis de requisicao.

17. Como e que os gestores dos veiculos na universidade atuam no caso de certos veiculos necessitarem de manutencao? Todos os veiculos vao para a mesma oficina? Ou criam um registo apenas para saberem que esse veiculo foi para manutencao para uma certa oficina e saberem qual oficina?

R: Nao e necessario gerir essa informacao.

18. Como podemos diferenciar as pessoas que utilizam este serviço?

R: Pelo nº de funcionario

19. A aplicação fica responsavel pela lotacao do veiculo?

R: Cada veiculo tem uma lotacao que nao pode ser excedida. A aplicacao nao deve deixar introduzir um nº de passageiros superior a lotacao do veiculo.

20. Devera haver veiculos disponiveis imediatamente no caso de uma emergencia? Veiculos estes que não necessitem de reservar uma hora para viagem.

R: Nao. Esses nao entram na aplicação

21. Que mais informacao sera guardada momento da reserva do veiculo para alem de horario de reserva, veiculo em questão, nome do condutor, etc?

R: Nº de funcionario requisitante, tipo de carro, nº de passageiros, local de destino, data e hora prevista de saida e chegada, motorista?, motivo da requisicao.

22. O portal de registo da aplicacao deve ser acessivel pela internet?

R: Sim

23. Devemos encriptar os credenciais dos utilizadores da aplicacao?

R: Deviamos usar o sistema de autenticacao LDAP

24. Quem e que determina se as propostas de viagens são viaveis ou nao para serem aceites para a reserva de um certo veiculo? Existe algum moderador para a autorizacao do certo veiculo para a certa viagem?

R: Talvez um elemento da direcao da escola ou da secretaria.

25. Deve haver diferentes tipos de gestores com diferentes responsabilidades?

R: Vamos tentar nao complicar muito a este respeito. Talvez apenas um tipo de gestor que gere a informacao dos veiculos e valida as requisições

26. Na reserva de um veiculo, a pessoa pode selecionar um a escolha, ou e-lhe atribuido um veiculo?
R: Pode selecionar dentro dos disponiveis.

2.2.2 Levantamento dos Requisitos

1. Devemos manter, de cada veículo, a matricula, número de lugares, escola a que pertence, marca, submarca, anos de vida, cor, tipo de combustível, tipo de carro, com ou sem motorista.
2. Devemos associar a cada requisição apenas o nome e número de funcionário que está a fazer a requisição
3. Ao inicio de cada viagem devemos obter, a hora de inicio, local de inicio, nº de quilometros iniciais do carro e o número de passageiros
4. Ao final de cada viagem devemos obter, a hora do fim, local do fim e o número de quilometros finais do carro
5. Para além do que obtemos no ínicio e final de viagem, devemos obter também, o destino, data, objetivo da viagem e o feedback sobre o estado do carro ou de alguma ocorrência a registar
6. Devemos diferenciar os diferentes veículos pela matricula e pelo tipo de carro: carro, jipe, autocarro, carrinha, etc.
7. No inicio de cada viagem devemos confirmar o id dos passageiros a bordo do veículo
8. Um condutor deve ter a opção de adicionar outros utilizadores a uma reserva
9. Deve ser capaz de notificar o usuário sobre um atraso na disponibilidade do veiculo
10. Dada uma viagem concluida, deve ser capaz de marcar um veiculo como livre para outra viagem, atravez de uma funcionalidade de checkout
11. O condutor deve poder registar um acidente
12. Deve informar sobre a necessidades de manutenção de um veículo
13. Deve poder registar veículos com e sem motorista
14. Cada utilizador deve ter um ID de funcionário
15. Cada veiculo tem uma lotação que não pode ser excedida.
16. Os veículos, para casos de emergência, são tratados à parte e por isso não se mantêm registo sobre eles.
17. No momento de fazer a reserva, são mantidas a matrícula do veículo em questão, nº de funcionário requisitante, o tipo do carro em questão, o nº previsto de passageiros, o local de Destino, a data e hora previstas de saída, a data e hora previstas de chegada e o motivo da requisição.
18. O portal de registo da aplicação deve ser acessível pela internet.
19. Os utilizadores da aplicação são aceites pelo sistema de autenticação LDAP.
20. A secretaria é responsável pela validação das propostas de reserva.

21. Deve haver um gestor que gere as informações dos veículos
22. O condutor deve poder selecionar entre veículos disponíveis
23. Apenas funcionários da universidade podem reservar viagens

Capítulo 3

Concepção/desenho da Resolução

3.1 Modelo Conceptual

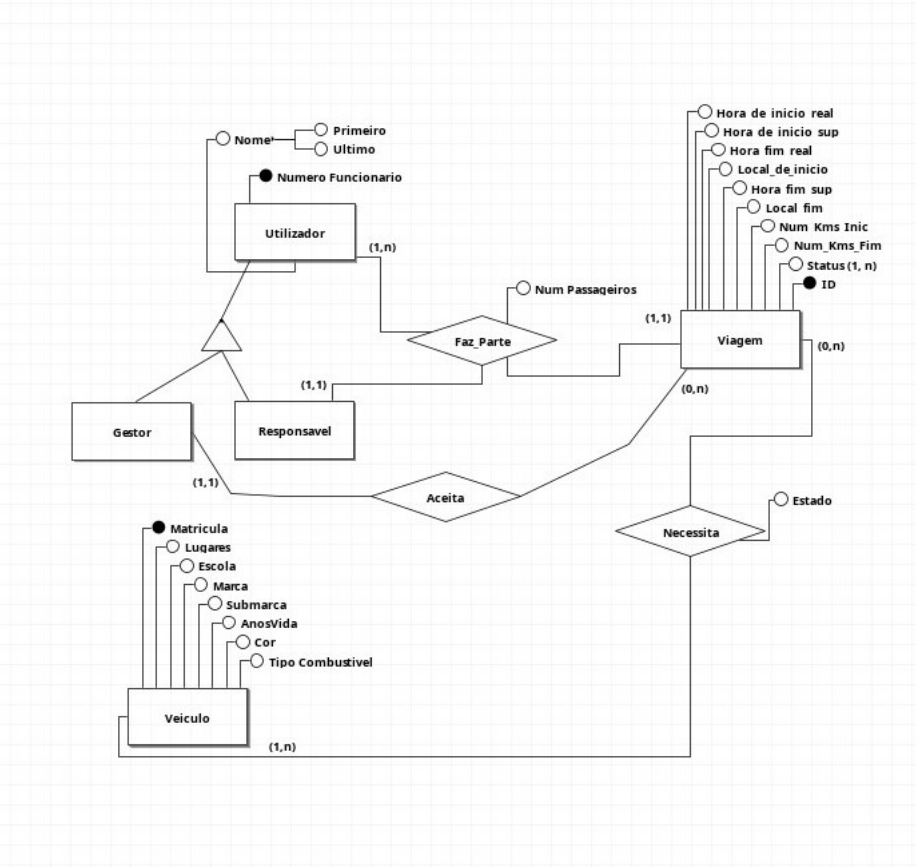


Figura 3.1: Modelo Conceptual da aplicação

3.2 Diagrama UseCase

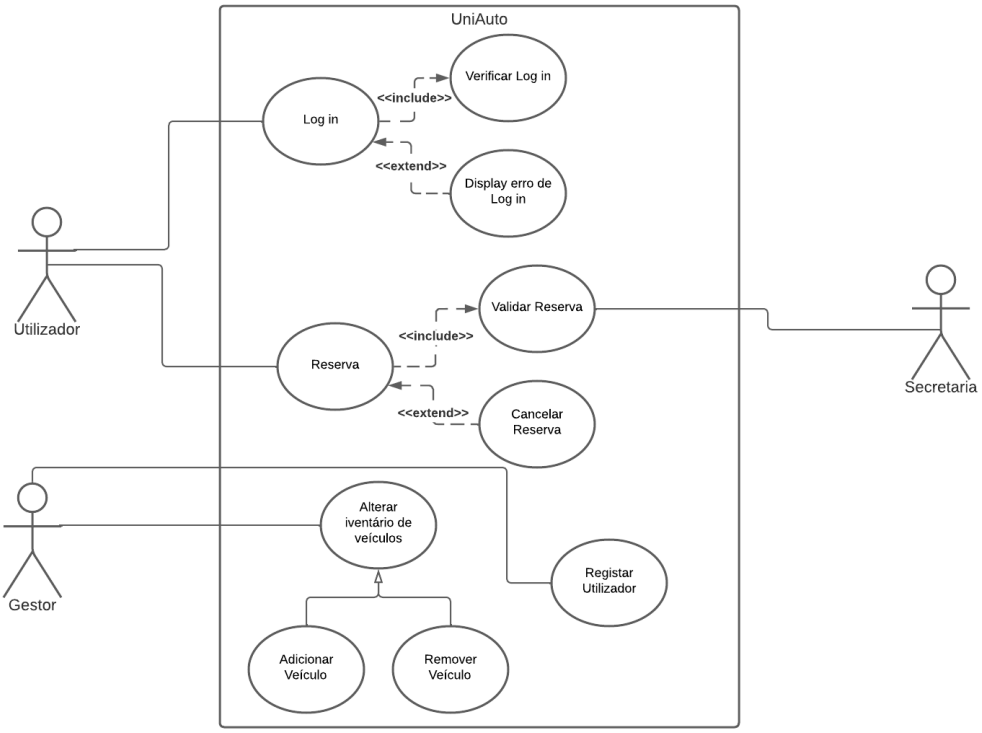


Figura 3.2: Diagrama UseCase

3.3 Diagrame de Classes

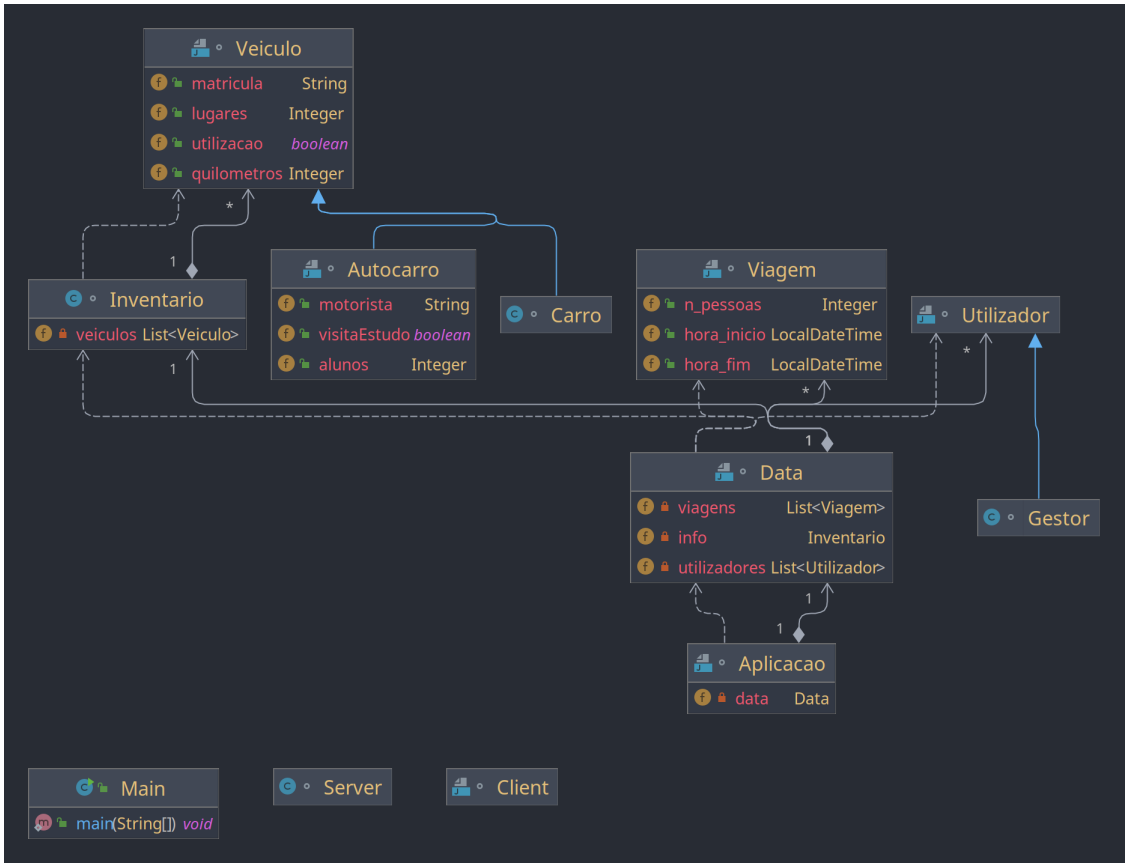


Figura 3.3: Diagrama de classes

Capítulo 4

Implementação

4.1 O Framework

Para o desenvolvimento de uma aplicação web, um dos frameworks mais utilizados é o Spring, mais propriamente o Spring boot, que usa o paradigma de desenvolvimento de software "Model-View-Controller"ou "MVC", utilizado em muitas aplicações orientadas aos objetos. Este framework tem ferramentas de "middleware", que ajudam a conectar elementos do front-end com o back-end, similares ao javascript. Exemplos de tais ferramentas são as tags "@PostMapping"e "@GetMapping"que quando definidas no html e no java, servem como ponte para passar informação de um extremo da aplicação (interface do utilizador no navegador web) para outro (back-end java e seguidamente mysql).
Eis os exemplos destas funcionalidades:

```
@PostMapping("/adduser")
    public String addUtilizador(@RequestParam String first, @RequestParam String last,
    @RequestParam String codigo) {
        Utilizador Utilizador = new Utilizador(first,last,codigo);
        CustomerRepository.save(Utilizador);
        return "Added new Utilizador to repo!";
    }
```

Listing 4.1: @PostMapping no java

No HTML, a tag "action"é mencionada com o valor "/adduser"que está definido no java. É também necessário especificar a tag "method"que especifica se o método é de leitura ou de escrita. Neste exemplo queremos adicionar um utilizador à base de dados, logo no HTML definimos o "form"como "POST"já que queremos escrever na base de dados.

```
<form action="/adduser" method="POST" id="user-form">
            <p>Vamos adicionar um utilizador:</p>
            <div>
                <input name="first" id="firstName" type="text" className="form-control"
    placeholder="Nome pr prio" aria-label="Username"
                    aria-describedby="basic-addon1"/>
            </div>
            <div>
                <input name="last" id="lastName" type="text" className="form-control"
    placeholder="Apelido" aria-label="Username"
                    aria-describedby="basic-addon1"/>
            </div>
            <div>
```

```
                <input name="codigo" id="codigo" type="text" className="form-control"
    placeholder="Codigo" aria-label="Username"
                       aria-describedby="basic-addon1"/>
            </div>
            <br/>
            <div  id="user-button">
                <button type="submit" className="btn btn-outline-primary me-2">Adicionar
    Utilizador</button>
            </div>
        </form>
```

Listing 4.2: Método "Post"no Html

4.2 Classes

Como foi referido anteriormente, o back-end da aplicação tem como suporte as classes "Utilizador", "Veículo"e "Viagem", definidas no java convencionalmente.

4.2.1 Utilizador

Cada elemento da base de dados tem um ID único que é independente do tipo do objeto, ou seja se forem adicionados à base de dados um utilizador, um veículo e uma viagem, estes terão id 1,2 e 3 respetivamente. Este processo é automatizado unicamente pelo Spring, e não é necessário que o programador mantenha um contador dos id's da base de dados. Basta que antes da respetiva variável de instância seja assinalada a tag "@Id"tal como a tag "@Generatedvalue"para que o valor seja incrementado automaticamente.

```
@Entity
public class Utilizador {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    private String nome;
    private String apelido;
    private String codigo; // numero de funcionario
```

Listing 4.3: Classe Utilizador

Estas classes são os objetos mais elementares da aplicação, o agrupar de diferentes utilizadores é registado no "CustomerRepository"definido pelo Spring, e onde podemos definir métodos de procura, entre outros. Este repositório utiliza também as operações CRUD (que foram um dos requisitos mencionados pelos docentes) que serão responsáveis pela manutenção da base de dados e especificam a ligação direta entre o Framework e a base de dados mysql.

```
public interface CustomerRepository extends CrudRepository<Utilizador, Integer> {
    Utilizador findUtilizadorById(Integer id);
}
```

Listing 4.4: Repositório dos Utilizadores

4.2.2 Veículo

Da mesma forma que definimos ID's e um repositório para a classe Utilizador, também o fazemos para a classe Veículo. Apresentamos então as variáveis de instância desta classe:

```
@Entity
public class Veiculo {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String matricula; // matricula do veiculo
    private int quilometros; // <-- a debater
    private int lugares; // capacidade maxima do veiculo
    private boolean utilizacao; // esta a ser utilizado, mediador de reserva
    private String escola; // escola a que pertence
    private String marca;
    private String modelo;
    private int ano; // ano de fabrico
    private String cor;
    private String combustivel;
    private boolean motorista; // tem ou n o motorista
    private String tipo; // tipo do veiculo (carro, autocarro, ...)
```

Listing 4.5: Classe Veículo

4.2.3 Viagem

```
public class Viagem {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private Date hora_inicio; // hora do inicio da viagem
    private Date hora_fim; // hora prevista de chegada ao destino
    private String local_de_inicio;
    private String local_de_fim;
    private Integer n_passageiros; // numero de pessoas da viagem ou ent o uma lista com
    as pessoas
    private String utilizadores; // lista dos id dos utilizadores
    private int condutor; // ID do utilizador condutor
    private int kms_iniciais;
    private int kms_finais;
    private int veiculo; // ID do veiculo utilizado
```

Listing 4.6: Classe Viagem

4.3 Página Web

Foi implementado uma versão protótipo da aplicação com objetivo de testar os diferentes componentes necessários, como a reserva de viagens da parte de utilizadores, tal como o registo de veículos pela parte de um gestor da aplicação. Optamos por uma abordagem em profundidade, ou seja, disponibilizar todos os campos que necessitam input do utilizador, para que sejam criados os respetivos objetos java, e estes sejam inseridos na base de dados, de modo a verificar que tudo funciona corretamente. Podemos assim, na fase de desenvolvimento estético da aplicação, reorganizar apenas estes campos na página web.

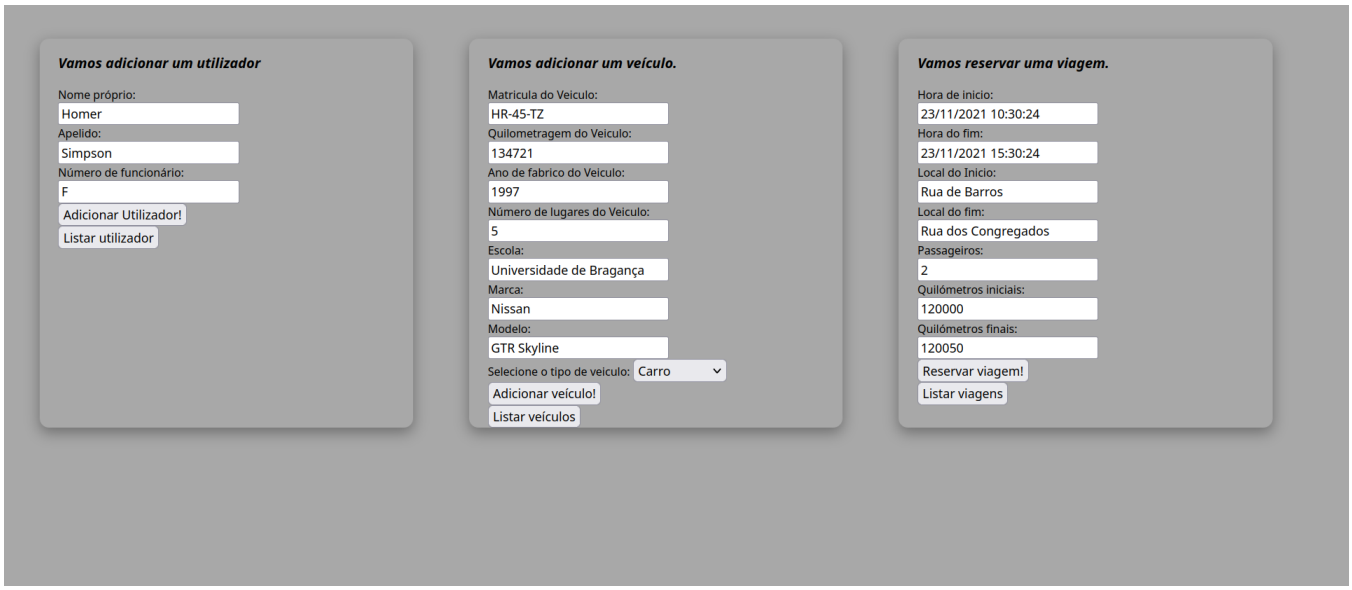


Figura 4.1: Prótótipo do website desenvolvido em html.

O Spring é flexível e é possível integrar varias "Views"e frameworks que melhor se adaptam às necessidades do desenvolvimento web comtemporaneo. Com o pacote "create-react-app"podemos configurar um ambiente de trabalho dentro do Spring que substitui a view tradicional que este usa, e nos deixa desenhar um website muito mais responsivo. **??**

Após algum desenvolvimento da página web em React desenvolvemos um layout mais amigável ao utilizador. No React, os componentes são os "tijolos"que constroem a página web. Na nossa aplicação, "App"é o componente principal, e o que trata do redirecionamento para diferentes páginas dentro do site. Está definido da seguinte forma:

```
function App() {
    return (
        <Router>
            <div className="App">
                <Routes>
                    <Route exact path="/" element={<Home/>}/>
                    <Route path="/veiculos" element={<Veiculos/>}/>
                </Routes>
            </div>
        </Router>
    );
}
```

Listing 4.7: Componente App do React

Figura 4.2: Página Web Temporária

Utilizamos o pacote "react-router-dom"que pode ser facilmente incluido em qualquer aplicação React com o comando:

```
$ npm install react-router-dom
```

Este é o pacote mais utilizado por desenvolvedores de aplicações web em React para redirecionamento.
O próximo componente do site é "Home", que decora a página inicial com uma barra de navegação e um menu para input do utilizador. Mais tarde serão desenvolvidas as funcionalidades de registo de log-in de utilizadores, mas para já os respetivos botões já se encontram na nav-bar. Existem tags "Link"no código fonte da página inicial que pertencem ao pacote react-router-dom, que se encarregam de especificar o novo endereço do website caso ocorra um clique nas abas de "Registo", "Veiculo", etc.

Para uma customização mais profunda do aspeto da página web, foi utilizada a biblioteca "Bootstrap", que tem componentes pré-definidos como botões e menus, que para além de facilitar o desenvolvimento, dão um aspeto profissional à página.

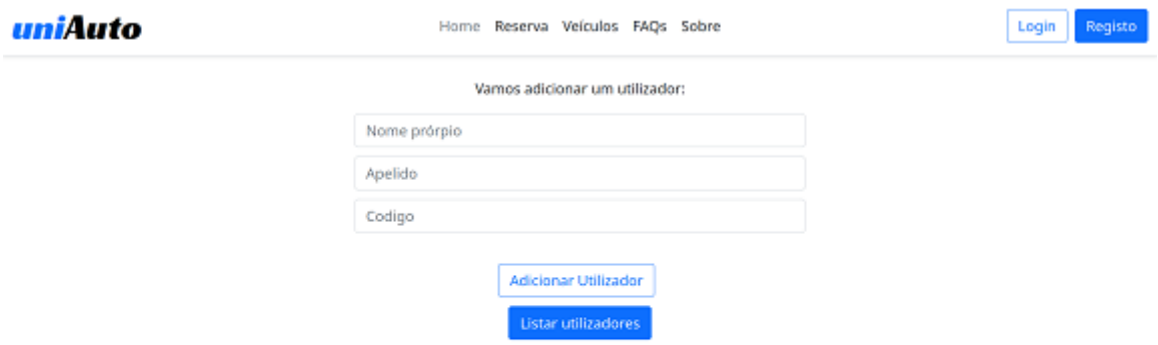


Figura 4.3: Página Atualizada

4.4 Alternativas, Decisões e Problemas de Implementação

```
MariaDB [uniauto]> show tables;
+--------------------+
| Tables_in_uniauto  |
+--------------------+
| hibernate_sequence |
| utilizador         |
| veiculo            |
| viagem             |
+--------------------+
4 rows in set (0.001 sec)

MariaDB [uniauto]> select * from utilizador;
+----+---------+--------+-------+
| id | apelido | codigo | nome  |
+----+---------+--------+-------+
| 20 | Simpson | F      | Homer |
| 21 | Simpson | H      | Marge |
+----+---------+--------+-------+
2 rows in set (0.061 sec)

MariaDB [uniauto]> select * from veiculo;
+----+------+-------------+------+---------------------------+---------+--------+-----------+-------------+-----------+-------------+-----------+------------+
| id | ano  | combustivel | cor  | escola                    | lugares | marca  | matricula | modelo      | motorista | quilometros | tipo      | utilizacao |
+----+------+-------------+------+---------------------------+---------+--------+-----------+-------------+-----------+-------------+-----------+------------+
| 51 | 1997 | gasolina 95 | azul | Universidade de Bragança  |       5 | Nissan | HR-45-TZ  | GTR Skyline |           |      134721 | carro     |            |
| 52 | 1997 | gasolina 95 | azul | Universidade de Bragança  |       5 | Nissan | HR-45-TZ  | GTR Skyline |           |      134721 | autocarro |            |
+----+------+-------------+------+---------------------------+---------+--------+-----------+-------------+-----------+-------------+-----------+------------+
2 rows in set (0.001 sec)

MariaDB [uniauto]> select * from viagem;
+----+---------------------+---------------------+------------+-------------+--------------------+-----------------+-------------+
| id | hora_fim            | hora_inicio         | kms_finais | kms_iniciais | local_de_fim       | local_de_inicio | passageiros |
+----+---------------------+---------------------+------------+-------------+--------------------+-----------------+-------------+
| 22 | 2021-11-23 15:30:24 | 2021-11-23 10:30:24 |     120050 |       120000 | Rua dos Congregados | Rua de Barros   |           2 |
+----+---------------------+---------------------+------------+-------------+--------------------+-----------------+-------------+
1 row in set (0.001 sec)

MariaDB [uniauto]>
```

Figura 4.4: Exemplo primitivo da configuração da base de dados no mysql.

4.5 Testes Realizados e Resultados

```
@SpringBootTest
class UniAutoApplicationTests {

    @Test
    public void whenCustomSerializingAndDeserializing_ThenObjectIsTheSame()
            throws IOException, ClassNotFoundException {
        Person p = new Person();
        p.setAge(20);
        p.setName("Joe");


        FileOutputStream fileOutputStream
                = new FileOutputStream("yourfile2.txt");
        ObjectOutputStream objectOutputStream
                = new ObjectOutputStream(fileOutputStream);
        objectOutputStream.writeObject(p);
        objectOutputStream.flush();
        objectOutputStream.close();

        FileInputStream fileInputStream
                = new FileInputStream("yourfile2.txt");
        ObjectInputStream objectInputStream
                = new ObjectInputStream(fileInputStream);
        Person p2 = (Person) objectInputStream.readObject();
        objectInputStream.close();

/*
        System.out.println("p nome: "+p.getName());
        System.out.println("p idade: "+p.getAge());
        System.out.println("p2 nome: "+p2.getName());
        System.out.println("p2 idade: "+p2.getAge());
*/
        assertTrue(
                p2.getName().equals( p.getName()));
        assertTrue(
                Objects.equals(p2.getAge(), p.getAge()));
    }
}
```

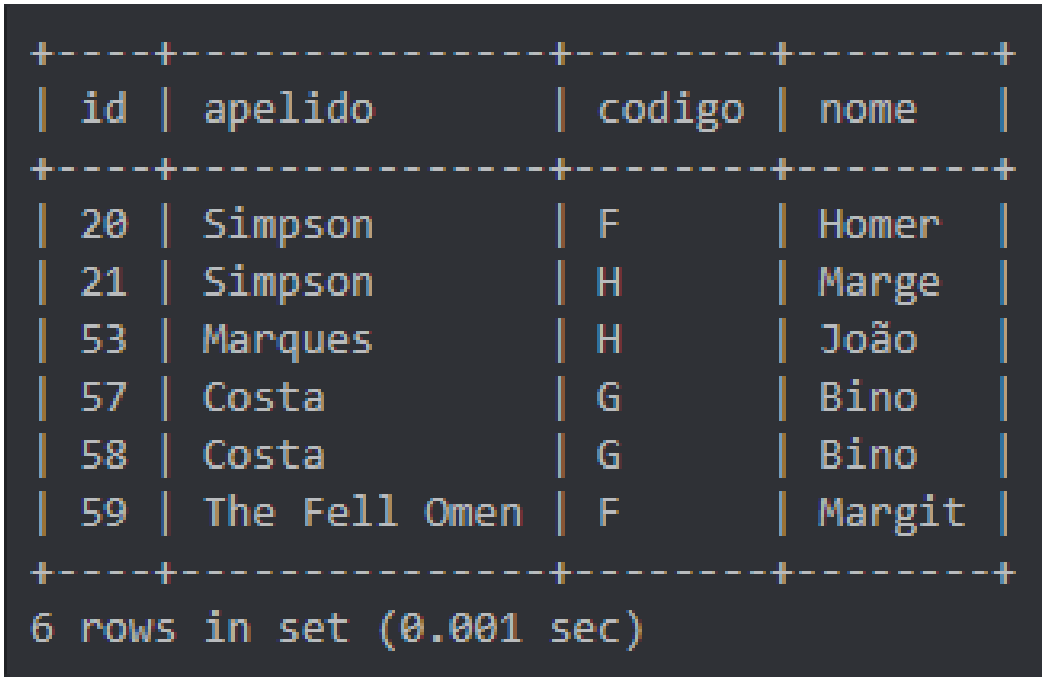
Listing 4.8: Teste em java

Capítulo 5

Manual de Utilização

5.1 Adicionar Utilizador

Figura 5.1: Adicionar Utilizador através do React



```
+----+---------------+--------+--------+
| id | apelido       | codigo | nome   |
+----+---------------+--------+--------+
| 20 | Simpson       | F      | Homer  |
| 21 | Simpson       | H      | Marge  |
| 53 | Marques       | H      | João   |
| 57 | Costa         | G      | Bino   |
| 58 | Costa         | G      | Bino   |
| 59 | The Fell Omen | F      | Margit |
+----+---------------+--------+--------+
6 rows in set (0.001 sec)
```

Figura 5.2: Utilizador Adicionado com sucesso à BD

Bibliografia

[AHM17] Cristiana Araújo, Pedro Rangel Henriques, and Ricardo G. Martini. Automatizing Ontology Population to drive the navigation on Virtual Learning Spaces. In Álvaro Rocha, Bráulio Alturas, Carlos Costa, Luís Paulo Reis, and Manuel Pérez Cota, editors, *Atas da 12ª Conferência Ibérica de Sistemas e Tecnologias de Informação, CISTI'2017*, pages 781–786. AISTI–Associação Ibérica de Sistemas e Tecnologias de Informação, June 2017.

[AHM18] Cristiana Araújo, Pedro Rangel Henriques, and Ricardo G. Martini. Virtual Learning Spaces Creation Based on the Systematic Population of an Ontology. *Journal of Information Systems Engineering & Management (JISEM)*, 3(1):1–11, Feb 2018.

[ANT16] ANTLR. ANTLR. `http://www.antlr.org/`, 2016. Accessed: 2016-09-14.

[Ara16] Cristiana Araújo. Building the Museum of the Person Based on a combined CIDOC-CRM/ FOAF/ DBpedia Ontology. Master's thesis, Universidade do Minho, Braga, Portugal, December 2016. MSc dissertation.

[Ara17a] Cristiana Araújo. Automatizing Ontology Population to drive the navigation on Virtual Learning Spaces. In Álvaro Rocha, Bráulio Alturas, Carlos Costa, Luís Paulo Reis, and Manuel Pérez Cota, editors, *Atas da 12ª Conferência Ibérica de Sistemas e Tecnologias de Informação, CISTI'2017*, pages 781–786. AISTI–Associação Ibérica de Sistemas e Tecnologias de Informação, June 2017.

[Ara17b] Cristiana Araújo. Norma – Simplex Project . Technical Report, UNU-EGOV, United Nations University, 2017.

[Cax93] Peter Caxton. The title of the work. How it was published, The address of the publisher, 7 1993. An optional note.

[Dam99] Luís Damas. *Linguagem C*. FCA, 1999.

[DJ90] P. Deransart and M. Jourdan, editors. *Attribute Grammars and their Applications*. INRIA, sv, Sep 1990. Lecture Notes in Computer Science, nu. 461.

[FK88] Drew Fudenberg and David M. Kreps. A theory of learning, experimentation, and equilibrium in games. 1988.

[HMU06] John E. Hopcroft, Rajeev Motwani, and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation*, chapter 5 – Context-Free Grammars and Languages. Addison-Wesley, 3rd ed. edition, 2006.

[Knu99] Donald Knuth. Knuth: Computers and typesetting, 1999.

[Mar18] Ricardo Giuliani Martini. *Formal Description and Automatic Generation of Learning Spaces based on Ontologies*. PhD thesis, University of Minho, Sep 2018.

[Mas85] Eric S. Maskin. The theory of implementation in Nash equilibrium: a survey. In Leonid Hurwicz, David Schmeidler, and Hugo Sonnenschein, editors, *Social Goals and Social Organization*, pages 173–204. Cambridge University Press, Cambridge, 1985.

[PH91] Luis Filipe Pinto and Pedro Rangel Henriques. *Animador de Especificações OBLOG*. gdcc, 1.st edition, Sep 1991.