



# MANUAL DO PROJETO

## Plataforma de conectividade V2C baseada em IoT

## Histórico de Revisões

| Data       | Versão | Descrição                                  | Autor                    |
|------------|--------|--|--------------------------|
| 01/11/2018 | 1.0    | Criação do documento                       | Rafael Gomes de Oliveira |
| 10/12/2018 | 1.1    | Correções                                  | Rafael Gomes de Oliveira |
| 14/12/2018 | 1.2    | Instruções para hospedagem em servidor AWS | Rafael Gomes de Oliveira |

## Índice

|   |           |
|---|-----------|
| <b>OBJETIVO</b>   | <b>5</b>  |
| Objetivo do projeto   | 5         |
| Objetivo do manual  | 5         |
| <b>ARQUIVOS E LINKS</b>   | <b>5</b>  |
| Repositório do GitHub contendo os códigos do aplicativo e da API/Aplicação web: | 5         |
| Link para acesso à API:   | 5         |
| Link para registro de novo usuário na API:                                      | 5         |
| Android Studio:   | 5         |
| Spring Tool Suite:  | 5         |
| MySQL Workbench:  | 5         |
| <b>ESTRUTURA DO PROJETO</b>   | <b>6</b>  |
| OBD2 Scanner  | 6         |
| Aplicativo TCC Conectividade V2C  | 7         |
| New Drive   | 8         |
| Refuel  | 9         |
| Settings  | 10        |
| Instalando o aplicativo no smartphone   | 11        |
| Editando o aplicativo   | 12        |
| API   | 12        |
| Editando a API  | 13        |
| Aplicação web   | 13        |
| Home  | 14        |
| Veículos  | 15        |
| Rodagens  | 16        |
| Abastecimentos  | 18        |
| Editando a aplicação web  | 18        |
| Banco de dados  | 18        |
| <b>COMPLEMENTO</b>  | <b>19</b> |
| Dados de acesso ao banco de dados:  | 19        |
| Instruções para hospedagem em servidor AWS:                                     | 20        |
| Configuração  | 20        |
| Hospedagem da aplicação   | 21        |
| <b>RODAR O PROJETO</b>  | <b>22</b> |

## **1. OBJETIVO**

### **1.1. Objetivo do projeto**

Esse projeto objetiva desenvolver uma solução alternativa para conectividade veicular de baixo custo, possibilitando a aquisição de um grande volume de dados em tempo real, e independente das tecnologias proprietárias das grandes empresas do ramo.

### **1.2. Objetivo do manual**

Este documento tem como objetivo detalhar as funcionalidades do projeto, explicando como utilizá-las, e tudo que é necessário para a execução correta do mesmo.

## **2. ARQUIVOS E LINKS**

### **2.1. Repositório do GitHub contendo os códigos do aplicativo e da API/Aplicação web:**

[https://github.com/EvoSystems-com-br/ResidenciaSW2018\\_IoT\\_Veicular](https://github.com/EvoSystems-com-br/ResidenciaSW2018_IoT_Veicular)

### **2.2. Link para acesso à API:**

<http://ec2-18-188-51-12.us-east-2.compute.amazonaws.com:8080>

### **2.3. Link para registro de novo usuário na API:**

<http://ec2-18-188-51-12.us-east-2.compute.amazonaws.com:8080/registration>

### **2.4. Android Studio:**

O Android Studio 2.3 é necessário para edição e teste do código do aplicativo.

Link para download: <https://developer.android.com/studio/archive>

### **2.5. Spring Tool Suite:**

O Spring Tool Suite 3.9 é necessário para edição e teste do código da API.

Link para download: <https://spring.io/tools3/sts/all>

### **2.6. MySQL Workbench:**

O MySQL Workbench (qualquer versão) é necessário para acessar o banco de dados do projeto.

Link para download: <https://dev.mysql.com/downloads/workbench/>

### 3. ESTRUTURA DO PROJETO

Para o projeto, foram desenvolvidos um aplicativo Android que se conecta a um leitor OBD para receber as leituras dos sensores do veículo em tempo real, um banco de dados MySQL para armazenar os dados obtidos dos veículos, uma aplicação web para registrar, no banco de dados, os dados provenientes dos veículos além de gerar uma visualização desses dados e uma API para realizar a comunicação entre o aplicativo, a aplicação web e o banco de dados. Tanto a API quanto o banco de dados e a aplicação web foram hospedados em um servidor da Amazon Web Services (AWS).

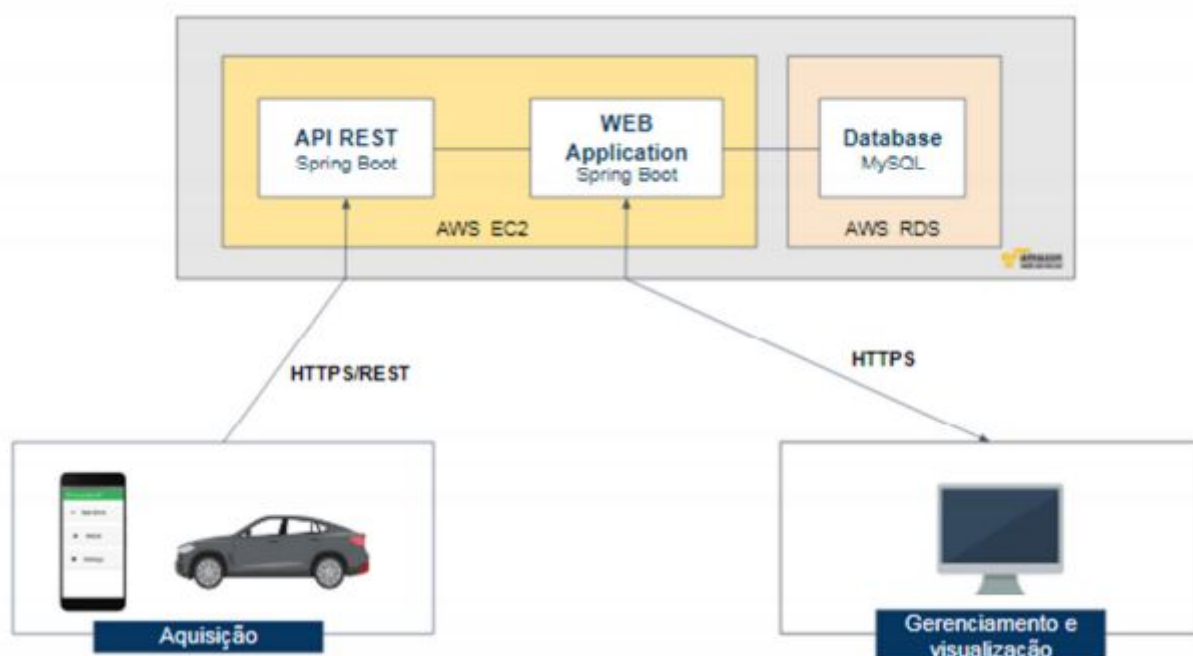


Figura 1 - Infraestrutura do projeto

#### 3.1. OBD2 Scanner

Para que este projeto funcione é necessário utilizar um leitor OBD modelo Elm327 com conexão bluetooth. Esse dispositivo é responsável por ler os dados provenientes dos sensores de um automóvel.



Figura 2 - Leitor OBD

### 3.2. Aplicativo TCC Conectividade V2C

O aplicativo desenvolvido tem o objetivo de realizar a comunicação bluetooth com o leitor OBD, receber os dados de leitura do mesmo e enviar esses dados para a API. Abaixo segue a descrição do aplicativo:

A tela inicial apresenta três botões: New Drive, Refuel e Settings.

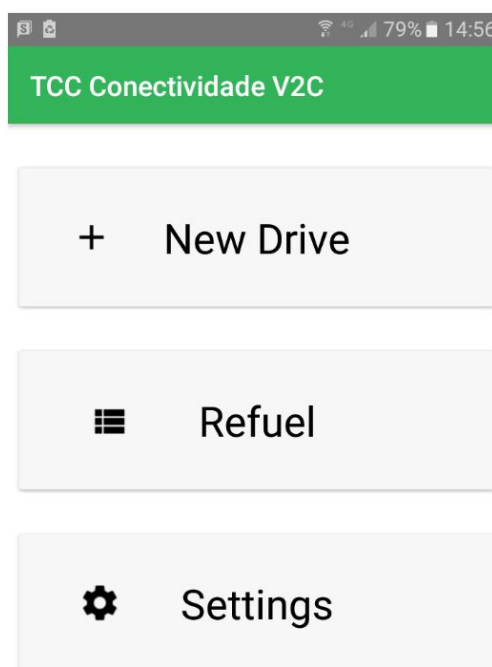
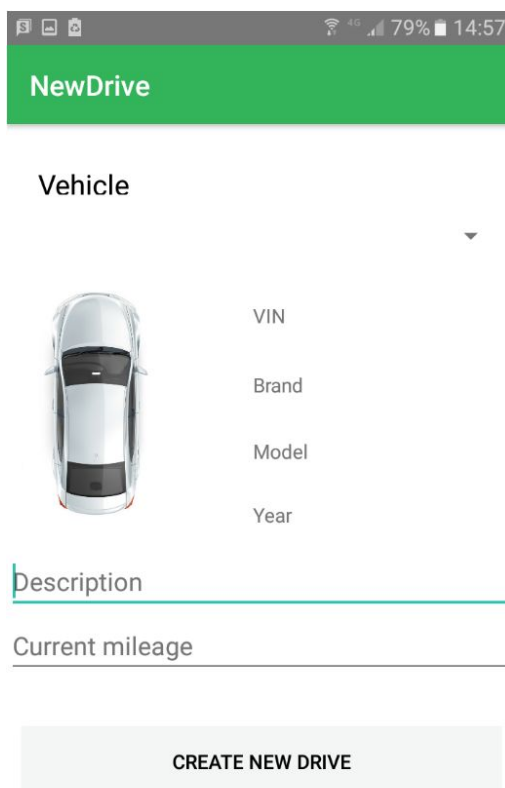


Figura 3 - Tela inicial do aplicativo

### 3.2.1. New Drive

Selecionando a opção New Drive, uma nova tela irá se abrir onde será possível selecionar um dos veículos cadastrados, escrever uma pequena descrição da viagem e informar a quilometragem.



The screenshot shows a mobile application interface for creating a new drive. At the top, there is a green header with the text "NewDrive". Below this, there is a section titled "Vehicle" with a dropdown arrow. Under the "Vehicle" section, there is a car icon and four input fields labeled "VIN", "Brand", "Model", and "Year". Below these fields, there are two more input fields labeled "Description" and "Current mileage". At the bottom of the form, there is a button labeled "CREATE NEW DRIVE".

**Figura 4 - Tela de criação de nova rodagem**

Após preencher esses campos e selecionar CREATE NEW DRIVE, uma nova tela irá se abrir. Essa tela contém os dados do veículo selecionado e, assim que a aquisição de dados começar, ela irá mostrar os dados que estão sendo adquiridos em tempo real.



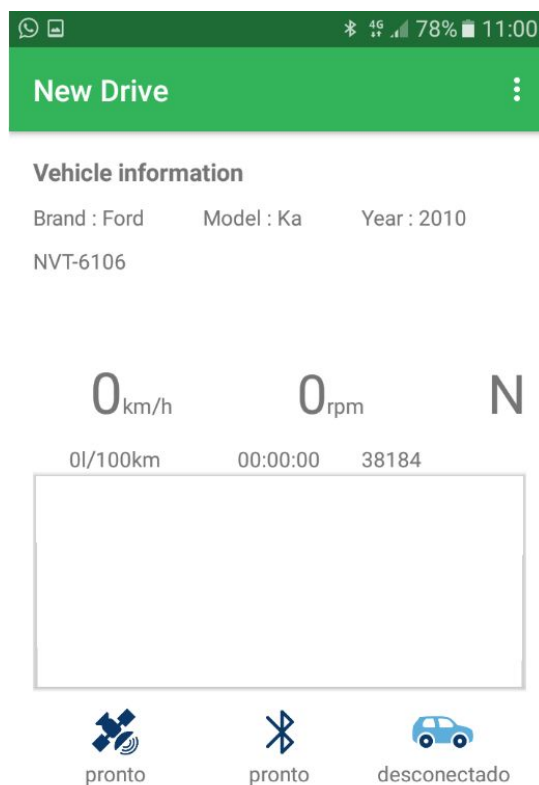



Figura 5 - Tela de aquisição de dados em tempo real

### 3.2.2. Refuel

Voltando à tela inicial e selecionando o botão Refuel, uma nova tela será aberta onde será possível selecionar um veículo cadastrado e inserir dados referentes a um abastecimento feito. Ao selecionar o botão CREATE NEW REFUEL, os dados previamente inseridos nesta página serão salvos como um novo abastecimento.

**NewTank**

Vehicle



FOJ5082

Brand : Chevrolet

Model : Sonic

Year : 2014

**Refuel Information**

Station

Fuel

Volume

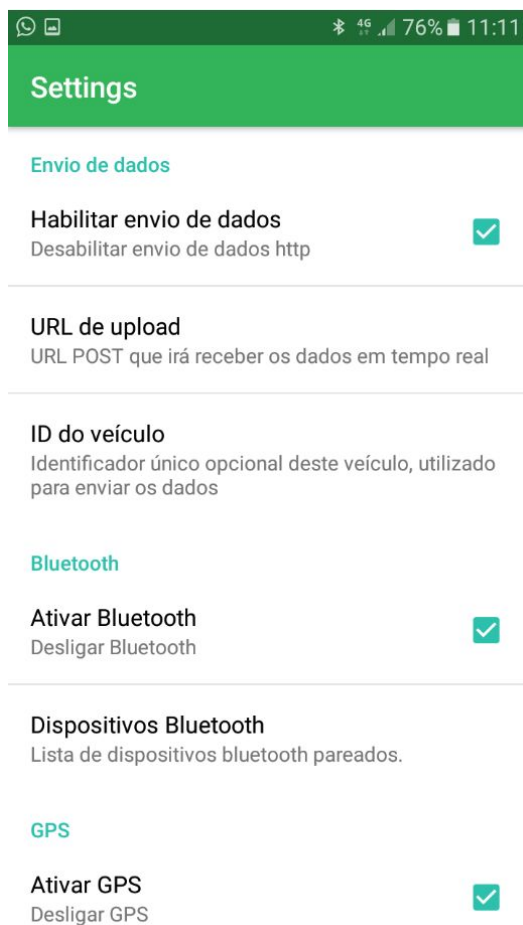
Odometer

CREATE NEW REFUEL

Figura 6 - Tela de registro de abastecimento

### 3.2.3. Settings

Voltando mais uma vez à tela inicial e selecionando Settings, fará com que seja aberta outra tela do aplicativo contendo opções de configurações para o mesmo.



**Figura 7 - Tela de configurações**

### 3.2.4. Instalando o aplicativo no smartphone

O aplicativo TCC Conectividade V2C foi desenvolvido para Android, portanto será necessário um smartphone com esse sistema operacional.

Para instalar o aplicativo no seu smartphone é necessário transferir o arquivo do tipo .apk do aplicativo para o celular, selecionar esse arquivo e clicar em instalar. Para que isso seja possível é necessário ativar a permissão para instalação de aplicativos de fonte desconhecidas (no seu smartphone, vá em “Configurações>Segurança” e marque a opção “Fontes desconhecidas”).

### 3.2.5. Editando o aplicativo

Para editar o código fonte do aplicativo é necessário fazer o download ou clonar o código do aplicativo presente no repositório do GitHub e utilizar o Android Studio para abrir e editar o código.

## 3.3. API

A API foi desenvolvida para fornecer os serviços necessários para o funcionamento do aplicativo:

1. Requisição de todos os veículos cadastrados;
2. Cadastrar um novo abastecimento;
3. Registrar uma nova rodagem;
4. Registrar as leituras provenientes do OBD na rodagem realizada em tempo real.

Abaixo segue uma imagem da API que mostra os recursos implementados:

```

64 //VEHICLE OPERATIONS
65 @RequestMapping(value = "/api/vehicle", method = RequestMethod.GET)
66 @ResponseBody
67 public List<Vehicle> returnAllVehicles() {
68     return vehicleservice.getAllVehicles();
69 }
70
71 //DRIVE OPERATIONS
72 @PostMapping("/api/vehicle/{vin}/drive")
73 public Drive createdrive(@PathVariable (value = "vin") String vin, @Valid @RequestBody Drive drive) {
74     drive.setVehicle(vehicleservice.findByVin(vin));
75     Calendar cal = Calendar.getInstance();
76     Date date=cal.getTime();
77     drive.setDate(date);
78     driveservice.createNewDrive(drive);
79     return drive;
80 }
81
82
83 //TANK OPERATIONS
84 @PostMapping("/api/vehicle/{vin}/tank")
85 public Tank createTank(@PathVariable (value = "vin") String vin, @Valid @RequestBody Tank tank) {
86     tank.setVehicle(vehicleservice.findByVin(vin));
87     Calendar cal = Calendar.getInstance();
88     Date date=cal.getTime();
89     tank.setDate(date);
90     tankservice.createNewTank(tank);
91     return tank;
92 }
93
94
95 //OBDREADING OPERATIONS
96 @PostMapping("/api/drive/{driveid}/obdreading")
97 public void createreading(@PathVariable (value = "driveid") int driveid, @Valid @RequestBody OBDReading reading) {
98     Calendar cal = Calendar.getInstance();
99     Date date=cal.getTime();
100     reading.setTimestamp(date);
101     reading.setDrive(driveservice.findById(driveid).get());
102     obdreadingservice.createNewReading(reading);
103 }
104
105 ...

```

Figura 8 - Recursos da API

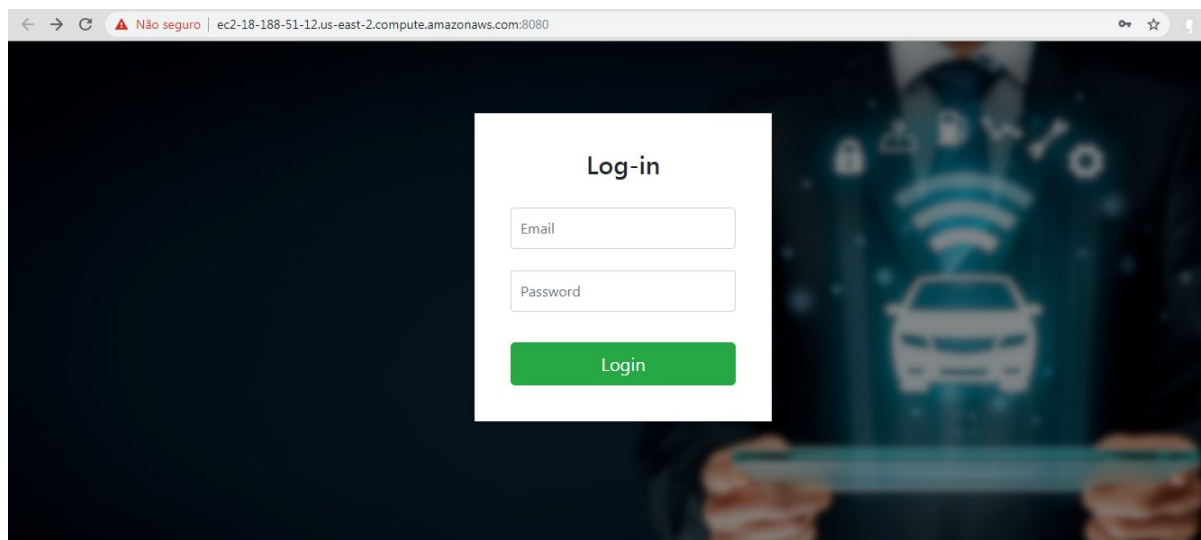
### 3.3.1. Editando a API

Para editar o código fonte da API é necessário fazer o download ou clonar o código da mesma presente no repositório do GitHub e utilizar o Spring para abrir e editar o código.

## 3.4. Aplicação web

A aplicação web desenvolvida para esse projeto tem o objetivo de cadastrar os dados das rodagens e dos registros de veículos e abastecimentos no banco de dados relacional além de gerar uma visualização das informações coletadas.

Funcionamento da aplicação web:

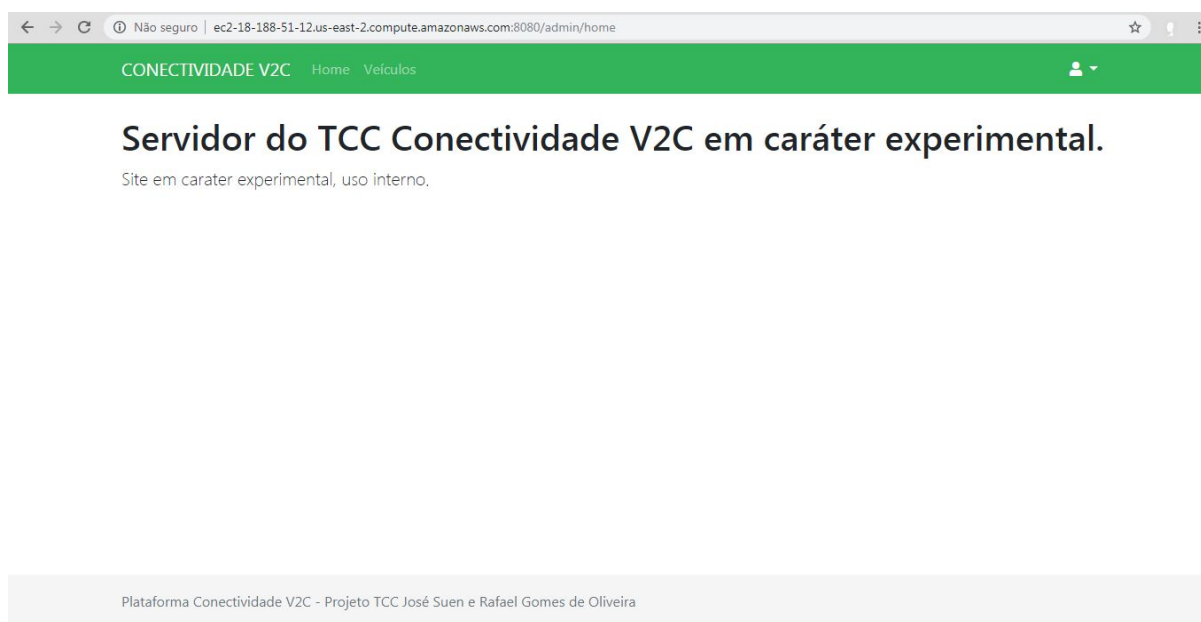


**Figura 9 - Tela de login**

Para acessar a aplicação, é necessário antes realizar um login. Na tela inicial na parte superior é possível selecionar dois botões diferentes: Home e Veículos.

### 3.4.1. Home

O botão Home leva o usuário à tela inicial:



**Figura 10 - tela inicial**

### 3.4.2. Veículos

Selecionar Veículos fará com que seja aberta uma nova tela onde é possível selecionar qualquer veículo cadastrado ou cadastrar um novo veículo.



**Veículos cadastrados no sistema**

Abaixo estão os veículos já cadastrados no sistema.

| VIN       | Marca     | Modelo | Ano  | Potência | Torque | Cilindrada |
|-----------|-----------|--------|------|----------|--------|------------|
| FOJ5082   | Chevrolet | Sonic  | 2014 | 116.0    | 12.0   | 1.6        |
| DeckerCar | Peugeot   | 307    | 2008 | 96.0     | 128.0  | 1.6        |
| FHO7880   | Honda     | Civic  | 2016 | 155.0    | 20.0   | 2.0        |
| PYW2126   | Audi      | A3     | 2017 | 150.0    | 250.0  | 1.4        |
| NVT-6106  | Ford      | Ka     | 2010 | 69.3     | 8.9    | 1.0        |
| ATX3526   | Chevrolet | Prisma | 2011 | 80.0     | 10.0   | 1.0        |

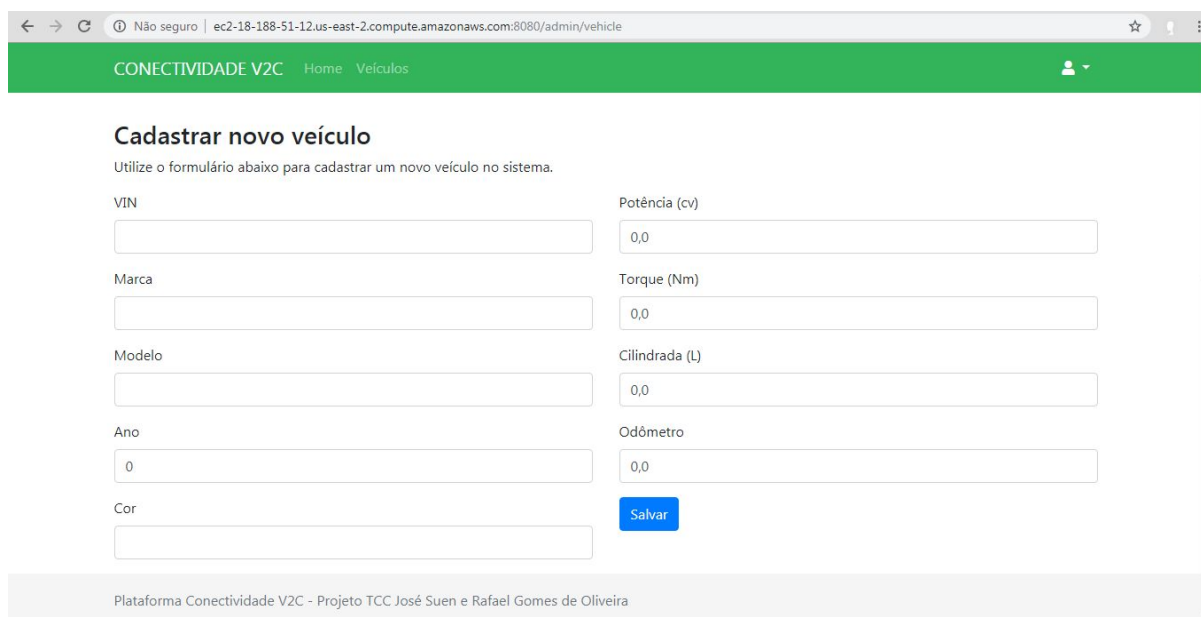
**Cadastrar novo veículo**

Utilize o formulário abaixo para cadastrar um novo veículo no sistema.

VIN

Potência (cv)

Figura 11 - Veículos cadastrados



**Cadastrar novo veículo**

Utilize o formulário abaixo para cadastrar um novo veículo no sistema.

VIN

Marca

Modelo

Ano

Cor

Potência (cv)

Torque (Nm)

Cilindrada (L)

Odômetro

**Salvar**

Plataforma Conectividade V2C - Projeto TCC José Suen e Rafael Gomes de Oliveira

Figura 12 - Cadastro de novo veículo



Para cadastrar um novo veículo basta preencher os campos e clicar no botão “salvar” na parte inferior da tela. Ao selecionar um dos veículos cadastrados, será aberta uma nova página contendo os detalhes do veículo e dois novos botões: “RODAGENS” e “ABASTECIMENTOS”.

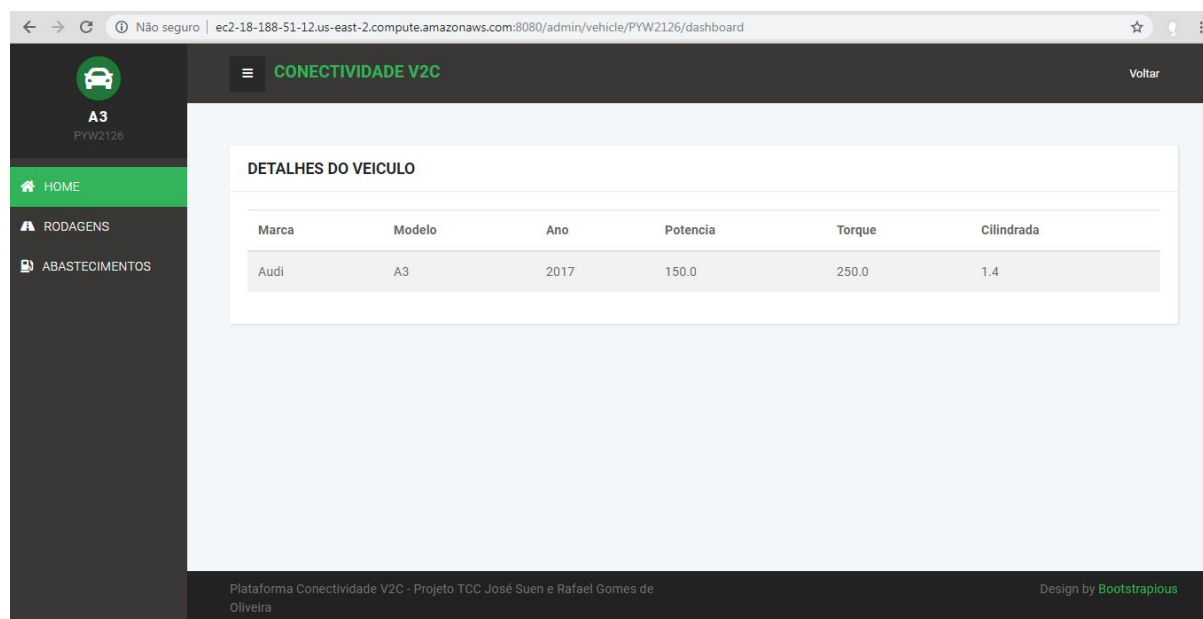
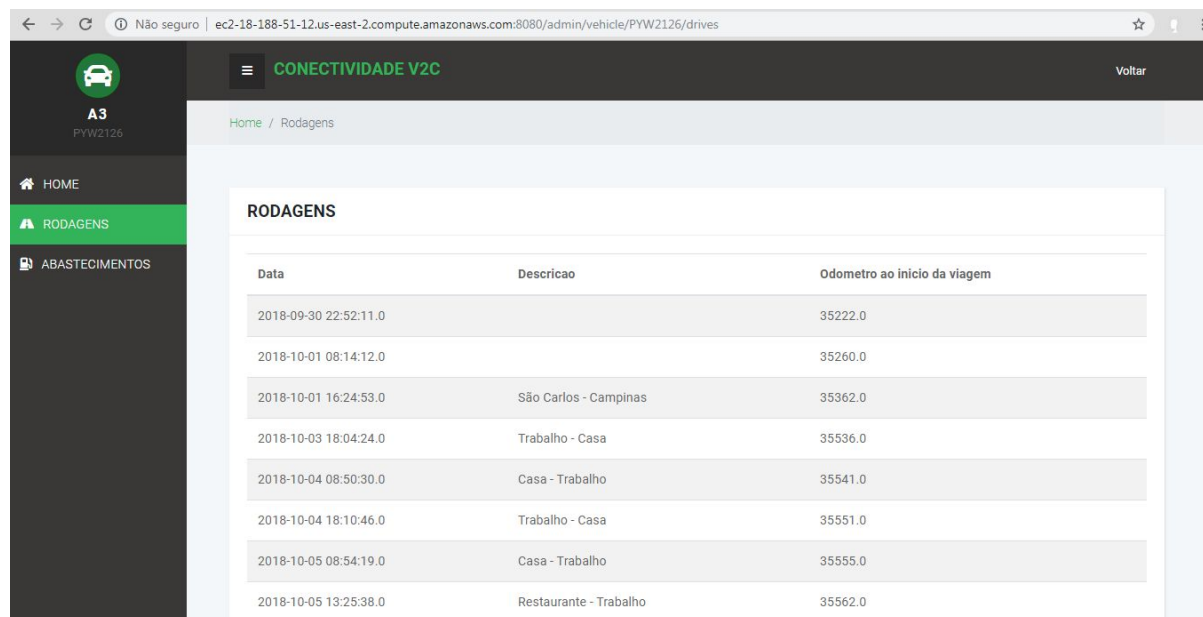


Figura 13 - Tela de detalhes do veículo

#### 1. Rodagens

Selecionar “RODAGENS” fará com que seja aberta uma nova página contendo todas as aquisições feitas no veículo selecionado.

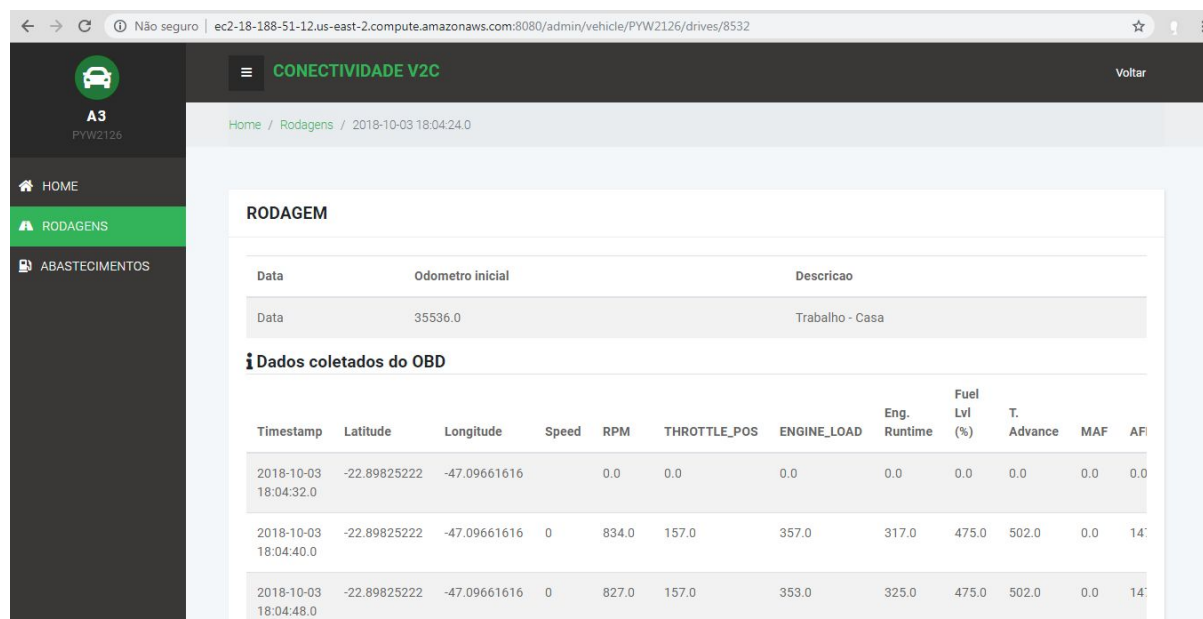




| Data                  | Descricao              | Odometro ao inicio da viagem |
|-----------------------|------------------------|------------------------------|
| 2018-09-30 22:52:11.0 |                        | 35222.0                      |
| 2018-10-01 08:14:12.0 |                        | 35260.0                      |
| 2018-10-01 16:24:53.0 | São Carlos - Campinas  | 35362.0                      |
| 2018-10-03 18:04:24.0 | Trabalho - Casa        | 35536.0                      |
| 2018-10-04 08:50:30.0 | Casa - Trabalho        | 35541.0                      |
| 2018-10-04 18:10:46.0 | Trabalho - Casa        | 35551.0                      |
| 2018-10-05 08:54:19.0 | Casa - Trabalho        | 35555.0                      |
| 2018-10-05 13:25:38.0 | Restaurante - Trabalho | 35562.0                      |

Figura 14 - Tela de rodagens do veículo

É possível selecionar qualquer uma das rodagens para visualizar os dados coletados durante a mesma.



| Data | Odometro inicial | Descricao       |
|------|------------------|-----------------|
| Data | 35536.0          | Trabalho - Casa |

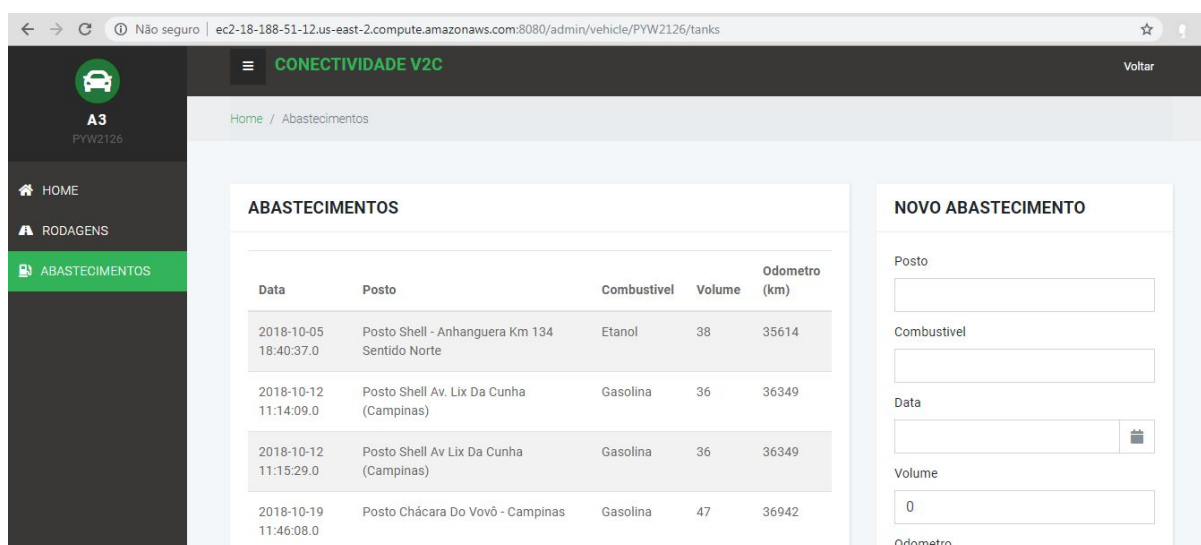
**Dados coletados do OBD**

| Timestamp             | Latitude     | Longitude    | Speed | RPM   | THROTTLE_POS | ENGINE_LOAD | Eng. Runtime | Fuel Lvl (%) | T. Advance | MAF | AFI |
|-----------------------|--------------|--------------|-------|-------|--------------|-------------|--------------|--------------|------------|-----|-----|
| 2018-10-03 18:04:32.0 | -22.89825222 | -47.09661616 | 0.0   | 0.0   | 0.0          | 0.0         | 0.0          | 0.0          | 0.0        | 0.0 | 0.0 |
| 2018-10-03 18:04:40.0 | -22.89825222 | -47.09661616 | 0     | 834.0 | 157.0        | 357.0       | 317.0        | 475.0        | 502.0      | 0.0 | 14' |
| 2018-10-03 18:04:48.0 | -22.89825222 | -47.09661616 | 0     | 827.0 | 157.0        | 353.0       | 325.0        | 475.0        | 502.0      | 0.0 | 14' |

Figura 15 - Dados de uma rodagem

## 2. Abastecimentos

Selecionar “ABASTECIMENTOS” irá abrir uma página com as informações de todos os abastecimentos registrados para o veículo selecionado e na qual também é possível registrar um novo abastecimento.




The screenshot shows a web application interface for vehicle management. The left sidebar contains navigation links: HOME, RODAGENS, and ABASTECIMENTOS (highlighted). The main content area is titled 'CONECTIVIDADE V2C' and 'ABASTECIMENTOS'. It features a table of refueling records and a form for adding a new record.

| Data                     | Posto  | Combustível | Volume | Odômetro (km) |
|--------------------------|--|-------------|--------|---------------|
| 2018-10-05<br>18:40:37.0 | Posto Shell - Anhanguera Km 134<br>Sentido Norte | Etanol      | 38     | 35614         |
| 2018-10-12<br>11:14:09.0 | Posto Shell Av. Lix Da Cunha<br>(Campinas)       | Gasolina    | 36     | 36349         |
| 2018-10-12<br>11:15:29.0 | Posto Shell Av Lix Da Cunha<br>(Campinas)        | Gasolina    | 36     | 36349         |
| 2018-10-19<br>11:46:08.0 | Posto Chácara Do Vovô - Campinas                 | Gasolina    | 47     | 36942         |

**NOVO ABASTECIMENTO**

Posto:

Combustível:

Data:  

Volume:

Odômetro:

Figura 16 - Tela de abastecimentos do veículo

### 3.4.3. Editando a aplicação web

Para editar o código fonte da aplicação web é necessário fazer o download ou clonar o código da aplicação web presente no repositório do GitHub e utilizar o Spring para abrir e editar o código.

## 3.5. Banco de dados

O banco de dados relacional foi construído a partir da modelagem das classes feitas para a API. Utilizando-se o framework do Spring Boot, foi possível através da dependência Hibernate realizar o mapeamento automático das classes nas tabelas, assim como as relações entre si. Dessa forma, não se foi necessário implementar o banco de dados manualmente.

Abaixo encontra-se uma figura gerada com o Mysql Workbench da estrutura final do banco de dados gerado com o Hibernate.

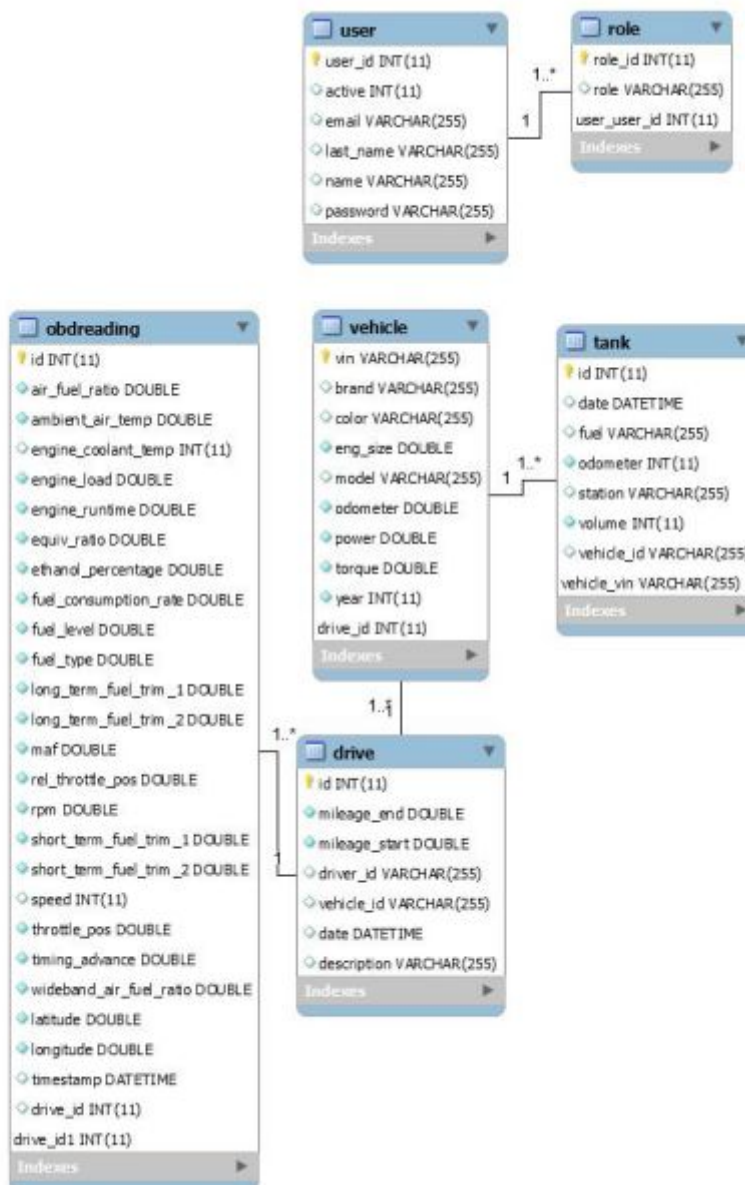


Figura 17 - Estrutura do banco de dados

## 4. COMPLEMENTO

### 4.1. Dados de acesso ao banco de dados:

Utilizando o MySQL Workbench, utilize os seguintes dados para ter acesso ao banco de dados do projeto:

Hostname: `iot-tcc-db.caopgydhbswt.us-east-2.rds.amazonaws.com`

Port: 3306

Username: `josesuen`

Senha: `kop99jk3`

## 4.2. Instruções para hospedagem em servidor AWS:

A implementação da plataforma pode ser feita em qualquer serviço de cloud ou hospedagem que ofereça acesso a uma máquina virtual com sistema operacional Linux, e um banco de dados Mysql na versão 5.8. Como neste projeto foi utilizado o Amazon Web Services (AWS), seguem abaixo as instruções para a hospedagem da plataforma nesse serviço (as instruções foram feitas considerando um usuário com conta criada e logado no serviço AWS):

### 4.2.1. Configuração

1. VPC, subredes e grupos de segurança:

Inicialmente, é necessário criar um ambiente virtual de nuvem privada (VPC) , três subredes (uma pública e duas privadas) e dois grupos de segurança.

- A subrede pública será responsável por hospedar a máquina virtual Linux, que por sua vez terá um acesso à internet.
- As subredes privadas serão dedicadas para hospedar o banco de dados e isolá-lo do ambiente externo, de modo que o acesso seja apenas possível através da máquina virtual.
- Um dos grupos de segurança deve ser criado para um servidor web público e deve ter uma regra de entrada que libera a porta 8080 para todos os IPs.
- O outro grupo de segurança deve ser criado para uma instância de banco de dados privado do Amazon RDS.

Para isso, siga o tutorial do site da AWS presente no seguinte link:

[https://docs.aws.amazon.com/pt\\_br/AmazonRDS/latest/UserGuide/CHAP\\_Tutorials.WebServerDB.CreateVPC.html](https://docs.aws.amazon.com/pt_br/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateVPC.html)

2. RDS:

Agora é possível criar um módulo RDS. Para isso , siga as instruções contidas no tutorial da AWS presente no seguinte link:

[https://docs.aws.amazon.com/pt\\_br/AmazonRDS/latest/UserGuide/CHAP\\_Tutorials.WebServerDB.CreateDBInstance.html](https://docs.aws.amazon.com/pt_br/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateDBInstance.html)

Lembre-se de usar as subredes privadas e o grupo de segurança para uma instância de banco de dados privado do Amazon RDS criados na etapa anterior.

3. EC2:

Após as configurações anteriores é possível instanciar um módulo EC2. Para isso, vá para o console Amazon EC2 em <https://console.aws.amazon.com/ec2/>

Clique no botão “Launch Instance” e, na página de criação da EC2 faça o seguinte:

- Step 1: selecione uma Amazon Machine Image (AMI) Ubuntu;
- Step 2: selecione o tipo t2.micro e clique em “next”;
- Step 3: Em “Network” selecione a VPC que você criou, em “Subnet” selecione a subrede pública também criada por você. Depois clique em “next” três vezes para chegar no Step 6;
- Step 6: Marque a opção “Select an existing security group” e então selecione o grupo de segurança para servidor web público que você criou. Depois clique em “Review and Launch”;
- Step 7: Revise os dados da instância criada e clique em “Launch”;
- Na página “Select an existing key pair or create a new key pair” escolha “Create a new key pair” e defina um nome para seu par de chaves em “Key pair name”. Selecione “Download Key Pair” e salve o arquivo de par de chaves em sua máquina local. Use este arquivo de par de chaves para se conectar à sua instância do EC2.

#### 4.2.2. Hospedagem da aplicação

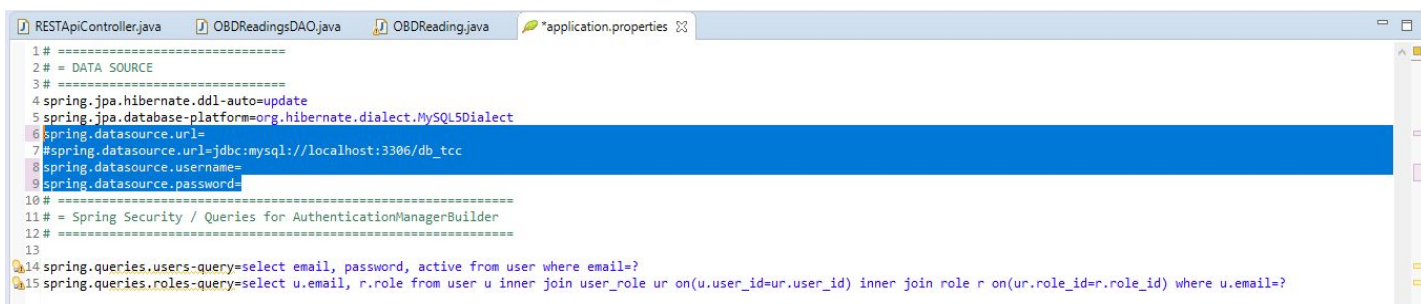
Com a cloud configurada, é possível hospedar a aplicação e rodá-la. Inicialmente deve-se gerar o arquivo compactado do tipo *fat jar*, que permite um deploy rápido por já possuir internamente o servidor Tomcat, e dessa forma a configuração do servidor web não precisa ser feita manualmente.

Para isso, inicialmente, instale o Apache Maven utilizando o seguinte tutorial:

<https://maven.apache.org/install.html>

Depois edite o arquivo `application.properties` encontrado, na pasta do projeto, em: `Código-fonte/tccserver-master/src/main/resources/application.properties`

Abaixo segue uma imagem do arquivo, com os campos que devem ser editados destacados, e as instruções de como editar esses campos:



```
1 # =====
2 # = DATA_SOURCE
3 # =====
4 spring.jpa.hibernate.ddl-auto=update
5 spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect
6 spring.datasource.url=
7 #spring.datasource.url-jdbc:mysql://localhost:3306/db_tcc
8 spring.datasource.username=
9 spring.datasource.password=
10 # =====
11 # = Spring Security / Queries for AuthenticationManagerBuilder
12 # =====
13
14 spring.queries.users-query=select email, password, active from user where email=?
15 spring.queries.roles-query=select u.email, r.role from user u inner join user_role ur on(u.user_id=ur.user_id) inner join role r on(ur.role_id=r.role_id) where u.email=?
```

Figura 18 - Arquivo `application.properties`

```
spring.datasource.url = ENDEREÇO_DO_BANCO_DE_DADOS  
(Exemplo de endereço: jdbc:mysql://iot-tcc-db.caopgvdhbswt.us-east-2.rds.amazonaws.com/tccdb)  
spring.datasource.username=Seu_Username  
spring.datasource.password=Sua_Senha
```

Feito isso, no terminal, navegue até a pasta do projeto e digite os seguintes comandos:

```
terminal > mvn clean  
terminal > mvn install  
target/build / arquivo.jar
```

O arquivo *fat jar* será gerado na pasta do projeto e poderá ser encontrado em `target/build / arquivo.jar`. Com o *fat jar* da aplicação em mãos, e com a aplicação devidamente configurada para comunicar com o módulo RDS, realiza-se o upload da aplicação para a máquina virtual linux por meio do protocolo SFTP.

Com o arquivo localizado na máquina virtual, navega-se até a pasta destino via SSH, e digita-se o seguinte comando que inicializará a aplicação e a manterá rodando:

```
nohup java-jar NOME_DO_ARQUIVO.jar
```

## **5. RODAR O PROJETO**

### **5.1. Aquisição de dados**

- 1 - Cadastre um veículo utilizando a aplicação web, na página de veículos.
- 2 - Conecte o leitor OBD à porta OBD2 do veículo. A porta OBD2, na maioria dos veículos, fica abaixo do volante e próxima aos pedais, como mostra a figura abaixo:





**Figura 19 - Porta OBD2 do veículo**

3 - Habilite o Bluetooth no seu celular, encontre o dispositivo OBD2 e pareie com ele. A senha normalmente é 1234.

4 - Abra o aplicativo, e vá na página de configurações:

1. Selecione o dispositivo pareado;
2. Habilite o GPS;
3. Habilite o upload de dados via HTTP;
4. Defina o intervalo de aquisição do OBD para 1 segundo.

5 - Volte para a página inicial, vá em New Drive, selecione o seu veículo, e preencha os campos com o odômetro atual e uma pequena descrição da rodagem (ex: De casa para o trabalho). Clique em "Create new drive".

6 - Clique no ícone de três pontos no canto superior direito da tela e em seguida em "Iniciar monitoramento em tempo real". A aquisição deverá começar automaticamente e os dados recebidos pelo aplicativo poderão ser visualizados na presente tela.

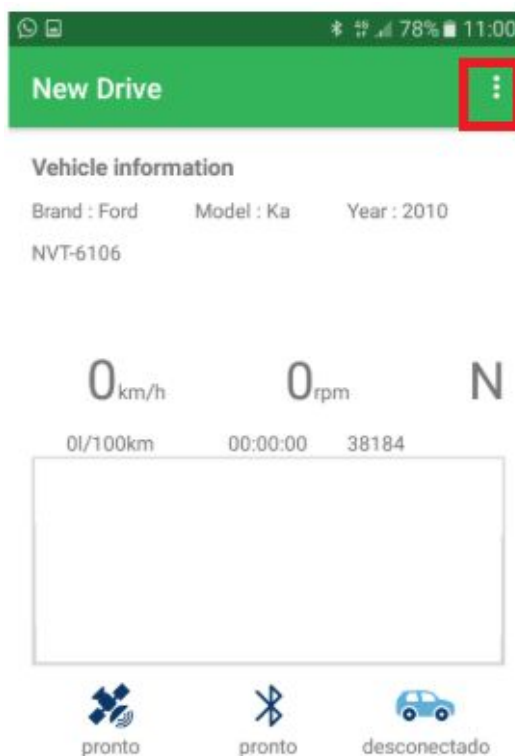


Figura 20 - ícone de três pontos destacado em vermelho

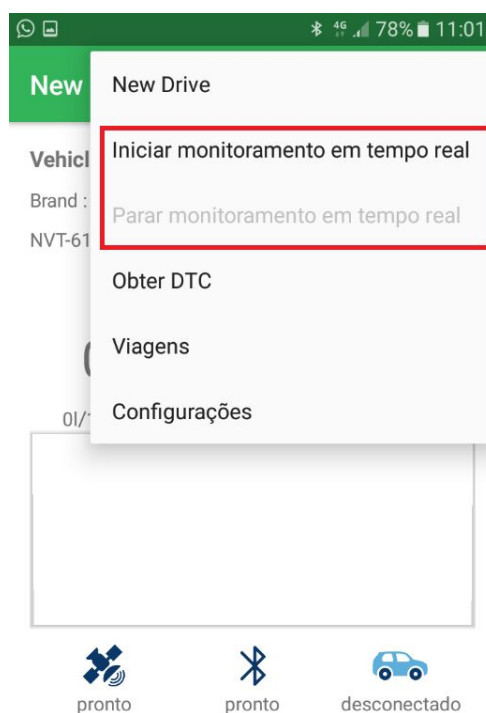


Figura 21 - Opções "Iniciar monitoramento em tempo real" e "Parar monitoramento em tempo real" destacadas em vermelho



7 - Para finalizar a aquisição, clique novamente no ícone de três pontos no canto superior direito da tela e selecione a opção "Parar monitoramento em tempo real".

8 - Para visualizar os dados da aquisição, acesse a aplicação web e vá na página de veículos, selecione o veículo desejado, abra a aba de rodagens e selecione a rodagem realizada.

## 5.2. Registro de abastecimento

1 - Para registrar um novo abastecimento, abra o aplicativo no seu celular e selecione o botão Refuel.

2- Selecione o seu veículo e preencha os campos vazios com, respectivamente, nome do posto, tipo de combustível, quantidade (em litros) de combustível abastecido e a quilometragem atual do veículo.

3 - Clique no botão CREATE NEW REFUEL e seu abastecimento será salvo.

4 - Para visualizar os dados do abastecimento, acesse a aplicação web e vá na página de veículos, selecione o veículo desejado, abra a aba de abastecimentos e selecione o abastecimento realizado.