CENTRO UNIVERSITÁRIO
Fundação Santo André

Arthur Cicuto Pires

Victor Vieira Paulino

Sistema de acompanhamento de transporte público para deficientes visuais

Santo André

2017

Arthur Cicuto Pires
Victor Vieira Paulino

Sistema de acompanhamento de transporte público para deficientes visuais

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Engenheiro Celso Daniel do Centro Universitário Fundação Santo André, como exigência parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Centro Universitário Fundação Santo André – CUFSA

Engenharia de Computação com Ênfase em Software

Orientador: Prof. Dr. Marcos Forte

Santo André

2017

ARTHUR CICUTO PIRES
VICTOR VIEIRA PAULINO

Sistema de acompanhamento de transporte público para deficientes visuais

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Engenheiro Celso Daniel do Centro Universitário Fundação Santo André, como exigência parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Trabalho aprovado. Santo André, 2017:

Prof. Dr. Marcos Forte
CUFSA

Nome 1
CUFSA

Nome 2
CUFSA

Santo André
2017

Agradecimentos

Agradecimentos aqui

Resumo

Colocar o resumo aqui.

Palavras-chave: Acessibilidade, deficientes visuais, transporte público, aplicativo mobile

Abstract

Abstract here. **Keywords**: Acessibility, smartphone, bus, application.

Sumário

Lista de ilustrações

Lista de abreviaturas e siglas

CUFSA Centro Universitário Fundação Santo André

1 Introdução

1.1 Referências do Sistema

Smartphones tem se tornado cada vez mais presentes na vida das pessoas. Uma pesquisa realizada pelo FGV-SP em 2016 [MEIRELLES, 2016] demonstrou que o número de aparelhos chegou a 168 milhões só no Brasil. Com sua facilidade de acesso, surgem inúmeras soluções que resolvem problemas do dia-a-dia dos usuários.

Dentre essas soluções, aplicações para smartphones que ajudam na mobilidade são cada vez mais comuns. Os aplicativos CittaMobi [VIEIRA, 2015] e Moovit [GOMES, 2015] vieram para mostrar que a tecnologia embarcada nos aparelhos podem ajudar a prever quanto tempo falta para o ônibus chegar em um ponto de parada, em tempo real. Eles capturam a geolocalização do usuário para saber qual ponto de ônibus eles estão próximos, possibilitando o usuário dizer de forma mais rápida qual seu ponto. Informando ao aplicativo qual seu ponto, eles podem selecionar um ônibus que passa no ponto selecionado, para saber quanto tempo resta para o veículo chegar.

Isso ajuda os usuários a se programar melhor, possibilitando a pessoa sair em um horário mais oportuno ou deixando ela mais tranquila sabendo que em breve seu ônibus chegará.

1.2 Descrição Geral

Este trabalho visa facilitar a vida de deficientes visuais que utilizam ônibus como meio de transporte. O aplicativo proposto irá possibilitar ao deficiente visual saber quanto tempo falta para seu ônibus chegar, enquanto o sistema se encarrega de avisar o motorista do ônibus qual o próximo ponto onde terá um deficiente visual esperando por aquele ônibus.

Sistemas operacionais de smartphone, como Android e iOS, possuem ferramentas nativas que adaptam o uso de aplicativos para pessoas com deficiências, possibilitando a utilização do aparelho sem grandes dificuldades, mas, nem sempre, criam boas experiências de uso.

O Android possui a ferramenta Talkback, para auxiliar no uso de qualquer aplicativo. Ao desenvolver uma solução para o sistema, é possível colocar tags específicas em cada elemento da

tela da sua aplicação. Isso possibilita o Talkback ler a tela com maiores detalhes para o deficiente visual ou utilizar a função de áudio dele para fazer áudios descrições mais detalhadas sobre o significado de uma tela.

Fazer aplicativos que funcionem em conjunto com essas tecnologias voltadas a deficientes já disponíveis, não é um trabalho difícil, mas criar boas experiências de uso que facilitem a vida de deficientes visuais é uma grande tarefa a ser cumprida.

Por isso é necessário adicionar outras tecnologias que facilitem o uso do app, neste caso, os Beacons. Beacon é um dispositivo que utiliza Bluetooth 4.0 (que tem baixo consumo de energia). Se existe um smartphone próximo a um Beacon, o aplicativo pode informar sua localização com maior precisão que um GPS.

Dessa forma quando um deficiente visual chegar no ponto, ele abre o aplicativo e, com uso do Beacon instalado no ponto, nosso aplicativo sabe em qual ponto o cego está. Sabendo isso, o app lista quais ônibus passam ali. Após o deficiente visual escolher um dos ônibus, o aplicativo vai notificar em intervalos pré-definidos quanto tempo falta para o ônibus chegar, em contrapartida o sistema irá alertar o motorista quando ele estiver próximo ao ponto em que existe um deficiente visual esperando por ele.

1.3 Restrições de projeto

- O smartphone deve ter o sistema operacional Android 4.1 (API Level 16) ou posterior instalado;
- O smartphone deve possuir Bluetooth 4.0 LE ou superior;
- O smartphone deve estar com a função Talkback ativada;
- O ônibus deve prover sinal de rede Wi-Fi para que o módulo do ônibus possa se comunicar.

2 Descrição da Informação

2.1 Visão Geral

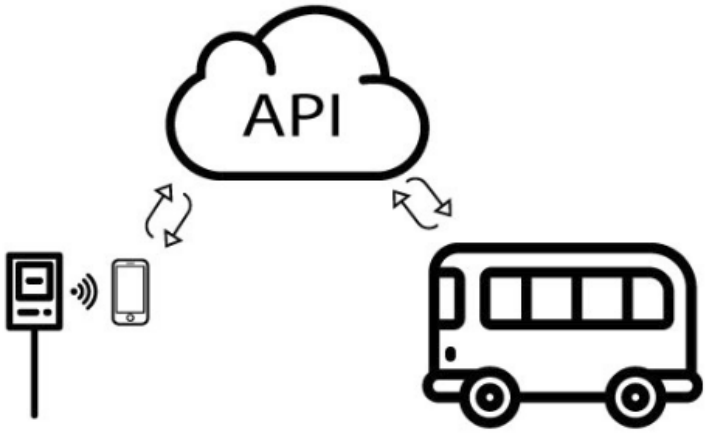

Figura 1 – Visão geral da comunicação dos componentes.

Módulo da parada de ônibus Emite informações de identificação da parada.

Aplicativo Reconhece o ponto de ônibus e solicita informações da API.

API Intermediário entre o aplicativo e o módulo do ônibus.

Módulo do Ônibus Mantém constante comunicação com o API enviando dados de geolocalização. Recebe também informação se deve alertar o motorista sobre deficiente visual na próxima parada.

2.2 Representação do Fluxo da Informação

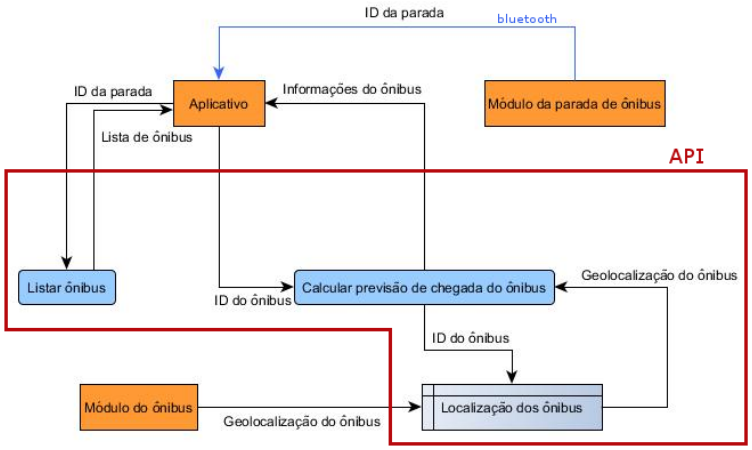

Figura 2 – Diagrama de fluxo de dados.

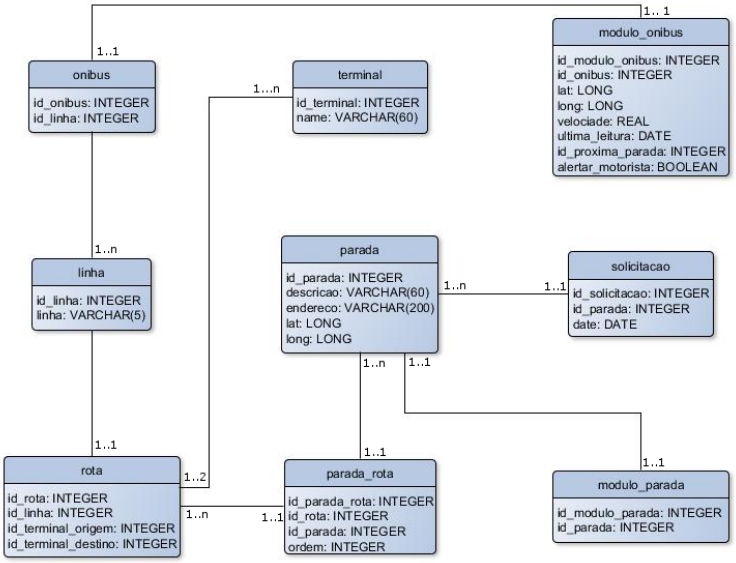

Figura 3 – Diagrama de banco de dados.

2.3 Interfaces com Sistema

2.3.1 Busca por um ponto próximo

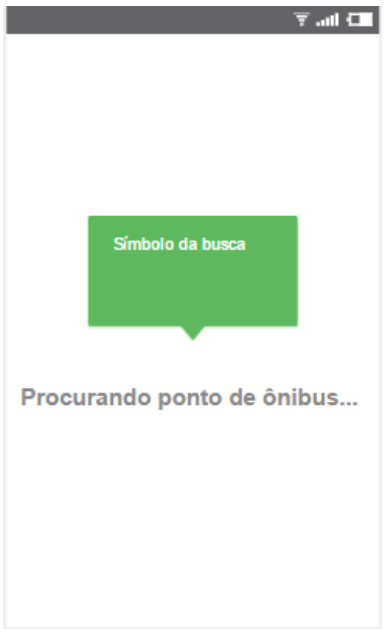

Figura 4 – Tela do aplicativo ao buscar por um ponto de ônibus próximo.

Número	Nome	Descrição	Requisitos	Grupo
1	Símbolo da busca do ponto de ônibus	Indica que o aplicativo está procurando um ponto de ônibus	-	Imagem e texto
2	Áudio sobre busca	Indica ao usuário que está sendo feito uma busca por algum ponto próximo	Função Talkback ativa	Áudio

Tabela 1 – Descrição dos elementos da tela de busca por ponto de ônibus próximo.

2.3.2 Busca por um ponto na API

Figura 5 – Tela do aplicativo ao buscar por um ponto de ônibus na API.

Número	Nome	Descrição	Requisitos	Grupo
1	Símbolo da busca das linhas de ônibus	Indica que o aplicativo procura as linhas de ônibus	O sistema deve ter detectado um ponto de ônibus	Imagem e texto
2	Áudio sobre busca	Indica ao usuário que está sendo feito uma busca dos ônibus disponíveis	Função Talkback ativa	Áudio

Tabela 2 – Descrição dos elementos da tela de busca por ponto de ônibus na API.

2.3.3 Lista de ônibus disponíveis

Figura 6 – Tela do aplicativo com ônibus disponíveis.

Número	Nome	Descrição	Requisitos	Grupo
1	Lista de linhas	Lista de linhas que o usuário pode escolher	Ter recebido uma lista da API	Botão
2	Áudio sobre escolha de um item	Indica que a lista de ônibus já está disponível	Função Talkback ativa	Áudio

Tabela 3 – Descrição dos elementos da tela de busca por ponto de ônibus na API.

2.3.4 Detalhes do ônibus

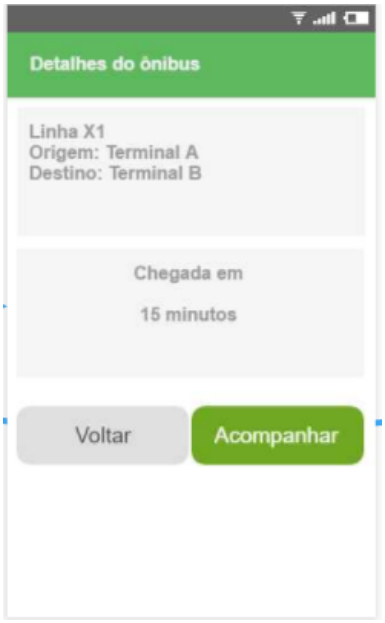

Figura 7 – Tela do aplicativo com detalhes de um ônibus.

Número	Nome	Descrição	Requisitos	Grupo
1	Linha X1	Mostra a linha selecionada	Receber previsão da API	Texto
2	Origem	Exibe o ponto inicial da linha	Receber previsão da API	Texto
3	Destino	Exibe o ponto final da linha	Receber previsão da API	Texto
4	Chegada em	Exibe a previsão de chegada da linha	Receber previsão da API	Texto
5	Voltar	Volta para a seleção de linhas	Receber previsão da API	Botão
6	Solicitar ônibus	Solicita que o ônibus pare no seu ponto e acionar o acompanhamento dele	Receber previsão da API	Botão
7	Áudio sobre previsão	Alerta ao usuário a previsão do ônibus	Receber previsão da API	Áudio

Tabela 4 – Descrição dos elementos da tela de detalhes do ônibus.

2.3.5 Detalhes do ônibus

Figura 8 – Tela do aplicativo sobre a solicitação de um ônibus.

Número	Nome	Descrição	Requisitos	Grupo
1	Informação de previsão	O sistema irá informar o usuário até a chegada do ônibus	Ter selecionado botão Solicitar ônibus	Texto
2	Favoritar	Adiciona ônibus como favorito	O ônibus não pode estar cadastrado como favorito. Caso esteja o botão não é exibido	Botão
3	Cancelar	Cancela o acompanhamento do ônibus e solicita que não pare mais no ponto	Ter selecionado botão Solicitar ônibus	Botão
4	Áudio sobre o acompanhamento	Informa ao usuário que está sendo feito o acompanhamento do ônibus	Ter escolhido acompanhar um ônibus. Função Talkback ativa	Áudio

Tabela 5 – Descrição dos elementos da tela sobre a solicitação de um ônibus.

2.4 Descrição Funcional

2.4.1 Divisão Funcional

2.4.1.1 Aplicativo para dispositivo móvel

Aplicativo que irá interagir com o deficiente visual. Sua função é verificar qual Beacon está mais próximo para que a API possa saber sua localização, podendo listar, via interface gráfica e áudio, para o usuário, quais linhas passam no ponto de parada que ele está.

2.4.1.2 API

Sistema que recebe informações do aplicativo e do módulo do ônibus. Tem como objetivo acessar os dados gravados no banco de dados para que possa prover informações de previsão ao aplicativo. Também é responsável por verificar se o módulo do ônibus deve alertar a presença de um usuário no próximo ponto. Além de calcular a previsão de um ônibus até o ponto de parada selecionado.

2.4.1.3 Módulo do ponto de ônibus

Dispositivo localizado em um determinado ponto de parada de ônibus. Emite constantemente um sinal ID para a identificação do ponto que ele se refere.

2.4.1.4 Módulo do ônibus

Dispositivo instalado no ônibus. Mantém comunicação constante com a API para informar sua geolocalização. Verifica ao mesmo tempo a necessidade de alertar o motorista se existe um deficiente visual aguardando no próximo ponto de parada.

2.5 Descrição funcional

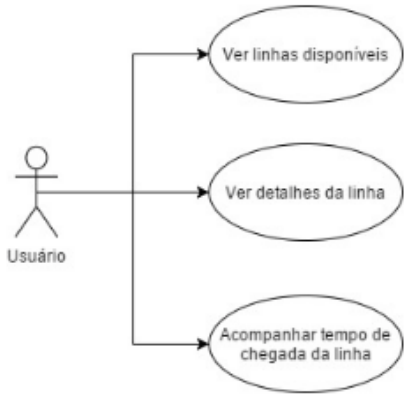

Figura 9 – Diagrama de caso de uso.

2.5.1 Narrativas: Casos de Uso

Solicitar horário do próximo ônibus da linha e sentido escolhido: Este caso de uso acontece quando um usuário solicita qual será a previsão de horário do próximo ônibus, de uma linha e sentido que ele poderá escolher de acordo com o seu ponto de ônibus.

Solicitar parada do ônibus escolhido: Este caso de uso é uma extensão do caso de uso Solicitar horário do próximo ônibus da linha e sentido escolhido, onde depois de escolher uma linha e sentido ele poderá solicitar a parada do próximo ônibus escolhido.

2.5.2 Diagramas de apoio para compreensão funcional

Identificação: UC001	
Nome: Solicitar horário do próximo ônibus	
Atores: Usuário	
Pré-condições: O aplicativo precisa ter lido o ID do módulo do ponto de ônibus	
Pós-condições: Retorno do horário do próximo ônibus e da solicitação de parada	
Fluxo de eventos	
Ator	**Sistema**
1. Usuário chega ao ponto de ônibus	2. Sistema lê o ID do ponto de ônibus e retorna uma lista de linhas
3. Usuário escolhe uma linha	4. Informar constantemente o horário do ônibus
Fluxo alternativo	
Não possui fluxo alternativo	

Tabela 6 – Tabela com caso de uso UC001.

Identificação: UC002	
Nome: Solicitar parada do ônibus escolhido	
Atores: Usuário	
Pré-condições: O usuário precisa ter solicitado o ônibus de uma linha	
Pós-condições: Confirmação de parada	
Fluxo de eventos	
Ator	**Sistema**
1. Usuário confirma solicitação de parada no seu ponto	2. Sistema retorna tela de seleção de ponto de ônibus destino
Fluxo alternativo	
1.a 1. Usuário cancela solicitação de parada	2. Sistema retorna cancela operação
3.a 1. Usuário cancela escolha de ponto de ônibus	2. Sistema retorna cancela operação

Tabela 7 – Tabela com caso de uso UC002.

3 Desenvolvimento

3.1 Protocolos de Comunicação

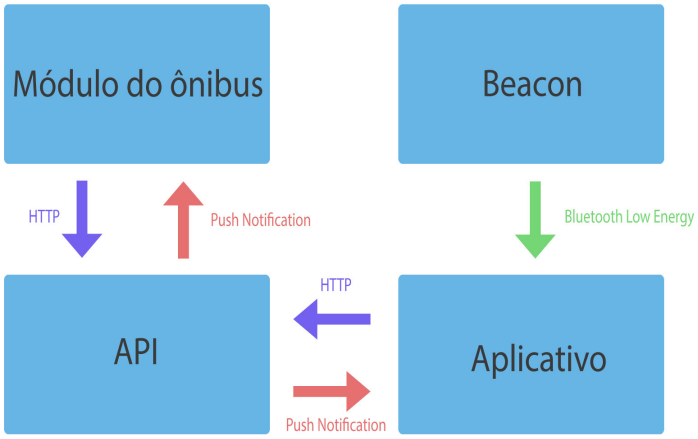

Figura 10 – Protocolos de comunicação utilizados.

3.1.1 HTTP

Hypertext Transfer Protocol é um protocolo baseado em requisições. Quando um cliente necessita de uma informação, ele solicita para o servidor que retorna uma resposta. Sua especificação permite requisições do tipo *GET, POST, DELETE*, dentre outros. Sua principal vantagem é não haver uma conexão aberta a todo momento para trafegar mensagens, permitindo que conexões e informações trafeguem apenas quando necessário. O formato para trafego das informações neste trabalho, por meio deste protocolo é a notação *JSON*.

Para o cenário deste projeto, tanto o aplicativo quanto o módulo do ônibus estão em cenários não favoráveis para o trafego de informação em grande escala, tendo em vista a baixa qualidade das redes 3G/4G dos smartphones e das redes WiFi que possuem nos ônibus.

3.1.2 Push Notification

Push Notification é um serviço de entrega de mensagens, parecido com *SMS* (*Shot Message Service*), mas que usa exclusivamente a internet para entregar. Cada plataforma possui

seu próprio serviço *Push*. Um bom uso deste serviço, é quando o emissor precisa enviar algo para o destinatário, sem a necessidade do destinatário ter solicitado antes, como ocorre no *HTTP*.

Neste projeto temos duas situações que a tecnologia é conveniente: primeiro, existe a necessidade de avisar o motorista que é necessário, em um dado momento, parar no próximo ponto para um deficiente visual. Segundo, precisamos avisar ao deficiente visual que seu ônibus já chegou e ele pode se dirigir a ele.

Nestes dois cenários precisamos avisar os dispositivos sobre algum evento e não temos uma conexão aberta constantemente como eles. Fazendo o serviço de *Push Notification* ser a melhor escolha.

Uma alternativa ao uso deste serviço são plataforma de *Realtime Database*. Eles funcionam de forma parecida com o protocolo *MQTT*, quando há alguma alteração em algum nó, os *subscribers* são notificados sobre o novo dado.

3.1.3 Bluetooth Low Energy

O *Bluetooth* é uma tecnologia de transmissão dados. Na sua versão 4.0+ ele se tornou *BLE* (ou *Bluetooth Smart*), trazendo a transmissão de dados com baixo consumo de energia.

Os pontos de ônibus não costumam possuir energia elétrica, com isso, surge a necessidade de uma tecnologia que tenha um moderado consumo de eletricidade. Com a necessidade do baixo consumo de energia e o envio constante, o *BLE* em seu modo *Beacons* ativado, se demonstrou ser a melhor alternativa suprindo todas as necessidades do projeto.

3.2 Módulo do Ponto de Ônibus

3.2.1 Hardware

1. HM-10 - Bluetooth 4.0 BLE module
2. Arduino Uno

Arduino Uno é utilizado apenas como ponte para configurar o módulo HM-10. A ligação deve seguir o diagrama abaixo.

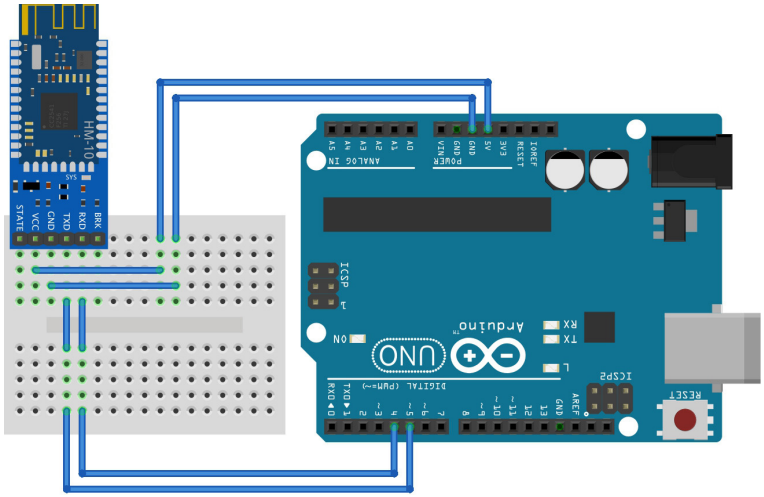

Figura 11 – Módulo Bluetooth HM10.

3.2.2 Software

- Arduino IDE 1.8.3 ou superior.

3.2.3 Configuração

Conecte o arduino Uno ao computador e compile o código abaixo utilizando a IDE do arduino.

```
#include <SoftwareSerial.h>

// SoftwareSerial (TX / RX)
SoftwareSerial mySerial(7, 8);

// Serial value
String val;

// Receive info
boolean receiveInfo = false;

// Bauds for detectBleBaudRate function
long bauds[] = {
    // major
    9600, 57600, 115200,

    // others
```

```
    19200, 38400, 4800, 2400, 1200, 230400
};

// Detect BLE Baud Rate
bool detectBleBaudRate() {
  Serial.println("-----------------------------");
  Serial.println("Detecting BLE baud rate:");

  for (int i=0; i < (sizeof(bauds) / sizeof(long)); i++) {
    Serial.write("Checking baud rate: ");
    // Checking baud rate
    long cur_baud = bauds[i];
    Serial.println(cur_baud, DEC);

    mySerial.begin(cur_baud);
    mySerial.write("AT");
    mySerial.flush();

    delay(50);

    String response = mySerial.readString();

    if (response == "OK") {
      Serial.println("-----------------------------");
      Serial.print("BLE Baud Detected: ");
      Serial.println(cur_baud);
      Serial.println("-----------------------------");

      return true;
    } else {
      mySerial.end();
    }
  }

  detectBleBaudRate();
}

// Uncomment function to wake up bluetooth or discover BLE Baud Rate
void setup() {
```

```
  mySerial.begin(9600);
  Serial.begin(9600);
}

void wakeUp() {
  Serial.println("-----------------------------");
  for (int i = 0; i < 30; i++) {
     mySerial.write("
        123456789012345678901234567890123456789012345678901234567890123456789012345678901
        ");
  }

  Serial.println("Wake up command has been sent to HM-10");
  Serial.println("-----------------------------");
}

// Reads strings that are sent to mySerial
void loop() {
  while (Serial.available() > 0) {
     delay(10); // Delay needed to recognize key and value properly

     String val = Serial.readString();
     Serial.print("Serial sent: ");
     Serial.println(val);

     // Receive info from HM-10
     receiveInfo = true;

     // If you need to wake up the HM-10 sensor, send wake command
     if (val == "wake") {
        wakeUp();
        receiveInfo = false;
     }

     // If you want to detect BLE Baud Rate, send detect command
     if (val == "detect") {
        detectBleBaudRate();
        receiveInfo = false;
     }
```

```
        mySerial.print(val);
        mySerial.flush();

        delay(50);

        if (receiveInfo) {
            // Read string from HM-10 (Software Serial)
            String response = mySerial.readString();

            Serial.print("mySerial received: ");
            Serial.println(response);

            Serial.println("-----------------------------");
        }
    }
}
```

Após compilador, utilizando o Serial Monitor da IDE, execute os comandos AT na seguinte ordem:

Obs: Quanto menor o tempo de envio, maior a economia de energia.

1. AT+RENEW //Coloca nos padrões de fábrica
2. AT+RESET //Reinicia para aplicar os padrões de fábrica
3. AT+MARJ0xNNNN //Define o valor Marjor
4. AT+MINO0xNNNN //Define o valor Minor
5. AT+NAMEMeuBeacon //Define o nome do Beacon
6. AT+ADVI5 //Define tempo de envio. 5 = 546.25 millisegundos
7. AT+ADTY3 //Define como não pareável
8. AT+IBEA1 //Habilita como Beacon
9. AT+DELO2 //Configura para apenas emitir sinal
10. AT+PWRM0 //Habilita auto-sleep para economizar energia

11. AT+RESET

Após configurado, pode ser ligado em uma bateria 3v para utilização.

3.2.4 Referências

HM-10 Bluetooth 4.0 BLE module Datasheet

Arduino IDE

Repositório da Metractive - Como construir Beacons

3.3 Módulo do Ônibus

3.3.1 Hardware

- Intel Edison
- Raspberry Pi 3
- Tela LCD 7"(em breve)
- NEO u-blox 6 GPS Modules

3.3.1.1 Intel Edison

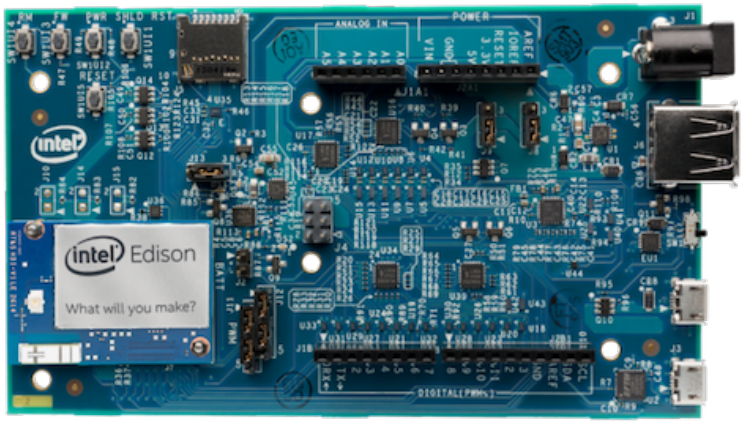

Figura 12 – Intel Edison.

Inicialmente foi adotado o Intel Edison com placa de expansão arduino. Foi escolhido devido a fácil acesso a um exemplar e ótimo hardware. Ele conta com WiFi, Bluetooth, portas

I/O, processador Intel Atom de 500 MHz, 1GB de memória RAM DDR3 e 4GB eMMC.
Sua utilização foi fácil e não obtivemos nenhuma dificuldade em instalar o sistema que escolhemos.

Problemas encontrados em adotar como solução:

Preço

Embora tenha um ótimo hardware e uma empresa séria por trás da sua construção, o preço, em 07/2017, que gira em torno de R$ 600,00, não justifica sua adoção como a melhor solução para o projeto já que existem alternativas com preços melhores e bom desempenho.

Ausência de controlador gráfico

Uma das features do projeto é emitir alertas visuais para o motorista por meio de telas LCDs. A placa Intel Edison nos permite fazer alertas visuais utilizando LEDs e afins.

Descontinuidade da placa pela Intel

em 07/2017, a Intel anunciou a descontinuidade do desenvolvimento de algumas placas que fabrica. O Intel Edison foi uma delas.

3.3.1.2 Raspberry Pi 3

Testes realizados no Raspberry Pi 3 demonstraram ser uma boa alternativa ao Intel Edison. Foi fácil a instalação do sistema e a placa vem com saída HDMI permitindo utilizar telas LCD para fazer os alertas visuais. Seu preço, em 08/2017, gira em torno de R$ 150,00, 1/4 do preço do Intel Edison. Seu hardware contém boas especificações:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports

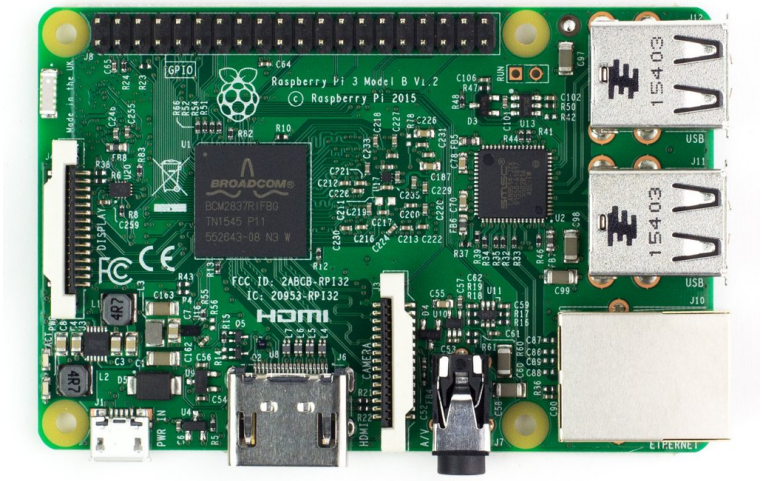

Figura 13 – Raspberry 3.

- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

Embora tenha um hardware com especificações superiores ao Intel Edison, não houve ganho de desempenho ao rodar o sistema, devido a ausência de algoritmos complexos no sistema. Assim, a grande vantagem de se utilizar o Raspberry Pi 3 ao invés do Intel Edison, é seu baixo custo e recurso de chip gráfico.

3.3.1.3 Módulo NEO u-blox 6 GPS

Para realizar o rastreamento do ônibus foi adotado o módulo NEO u-blox 6 GPS Modules, devido a compatibilidade com as placas que contém o sistema embarcado e preço acessível.

Figura 14 – Módulo NEO u-blox 6 GPS.

Localização

Uma característica desse módulo é trabalhar com GPS, fazendo comunicação direta com no mínimo 3 satélites para triangular sua posição com mais precisão. Alguns módulos disponíveis no mercado trabalham com A-GPS, que usam torres de telefonia móvel para conhecer sua posição. O uso do GPS trás maior precisão, porém demora mais para estabelecer conexão com satélites. O A-GPS fornece a localização com menor tempo, porém com menor precisão e a um custo mais alto.

Comunicação

O módulo realiza comunicação UART (Universal Asynchronous Receiver/Transmitter), o que permite fácil comunicação com as placas utilizadas para testes.

Preço

Seu preço, em 08/2017, gira em torno de R$ 60,00 e pode ser encontrado com facilidade na internet para venda.

3.3.2 Software

3.3.2.1 Sistema Operacional

Android Things Em 2016 o Google anunciou o Android Things, uma versão do Android voltada para IoT (Internet of Things). Ele é, atualmente, uma versão do Android Marshmallow reduzida. Sua escolha foi devido a facilidade de embarcar em placas como o Raspberry Pi e Intel Edison, e a variedade de recursos que já estão disponíveis no SO que facilitam o desenvolvimento do módulo, como o recurso LocationManager. **[Detalhar mais essa parte]**

3.3.2.2 IDE

Foi escolhido o Android Studio como IDE do projeto. Ela é desenvolvida pela IntelliJ e tida pelo Google como ferramenta oficial de desenvolvimento para aplicativos Android.

3.3.2.3 Linguagem

O Google tem duas linguagens de primeiro nível para desenvolvimento Android: Java e Kotlin. Para esse projeto adotamos a linguagem Kotlin, que possui sintaxe muito simplificado em comparação ao Java. Embora Java tenha sido a primeira linguagem oficial para desenvolvimento, Kotlin oferece acesso aos mesmo recursos do sistema. Algumas bibliotecas disponíveis, desenvolvidas por terceiros, ainda não migraram para o Kotlin, obrigando a implementar algumas classes em Java. Como Kotlin tem interoperabilidade com Java, não existe nenhum impeditivo de utilizar Kotlin e eventualmente alguma classa Java.

3.3.2.4 Arquitetura

Para desenvolvimento do software, foi adotado o padrão *Clean Archtecture*. É um padrão que visa um maior desacoplamento das classes e distruibui bem as responsabilidades.

Core Não contém nenhuma lógica de negócio. Esta camada provê informações comuns, como configurações estáticas da placa a toda a aplicação. Possui também algumas classes e interfaces bases.

Data Responsável por prover dados para toda aplicação. Ela adota o Padrão de Arquitetura *Repository*, tendo uma interface de acesso aos dados. Uma grande vantagem em utilizar essa camada com esse padrão de arquitetura, é o respeito a responsabilidade única, um

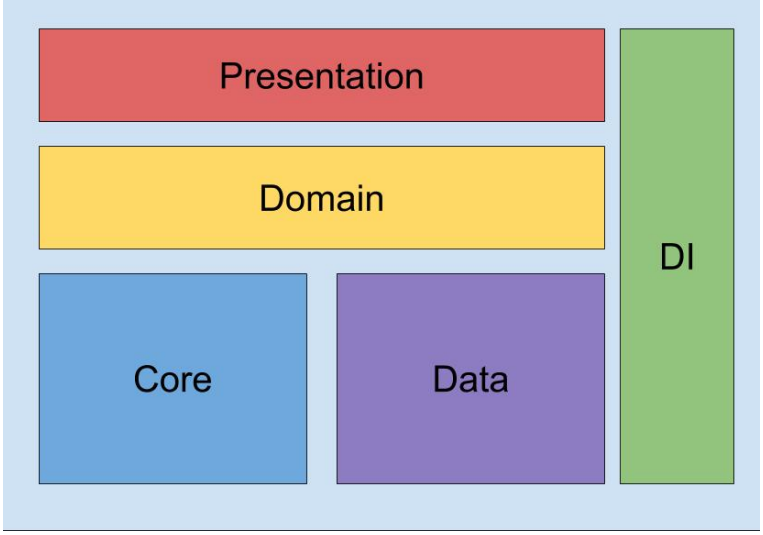

Figura 15 – Diagrama de bloco do módulo do ônibus.

dos princípios do *SOLID*. Ela encapsula toda lógica de busca de dados, assim, caso uma classe precise de algum dado específico, ela solicita através da interface de comunicação e a classe que implementa a interface, cuida de toda lógica de busca de dado, seja um dado armazenado localmente, em cache ou em um servidor remoto. Tudo fica transparente para a classe que solicitou o dado.

Domain Esta camada encapsula toda regra de negócios da aplicação. Toda vez que é necessário realizar processamentos em dados para satisfazer funcionalidades, é feito por esta camada.

Presentation Responsável por toda interface gráfica. Toda lógica de criação de telas e interceptação de interações do usuário com o aplicativo, é feito aqui. Quando é necessário procurar dados para exibir ao usuário, é feito solicitações deles para a camada Domain ou Data para que seja possa exibir os dados.

3.3.2.5 Animações

O sistema operacional provê uma *API* para animações que herdou da versão do *Android* de smartphone. Foi utilizado algumas animações para melhorar a experiência de uso dos motoristas com o módulo. Porém, por utilizar placas de baixo custo, no caso deste trabalho o *Raspberry Pi 3*, é visível a baixa qualidade de transições. O *Raspberry Pi 3* entrega 10 a 15 FPS, enquanto o aconselhável para ter uma boa fluidez é 30 FPS.

3.3.2.6 Referências

Site Oficial Intel Edison

Datasheet Intel Edison

Anúncio do fim da produção do Intel Edison

Site Oficial Raspberry Pi

Datasheet Raspberry Pi 3

Datasheet NEO u-blox 6 GPS Modules

Site Oficial Android Things

Configuração do Android Things no Intel Edison

Configuração do Android Things no Raspberry Pi 3

3.4 Aplicativo

3.4.1 Telas

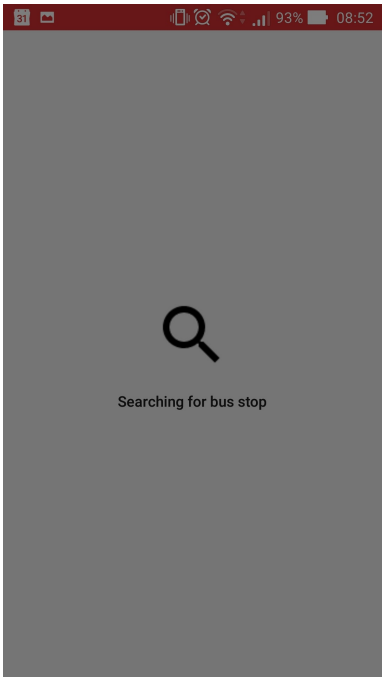

Figura 16 – Tela de busca por um ponto de ônibus do aplicativo móvel.

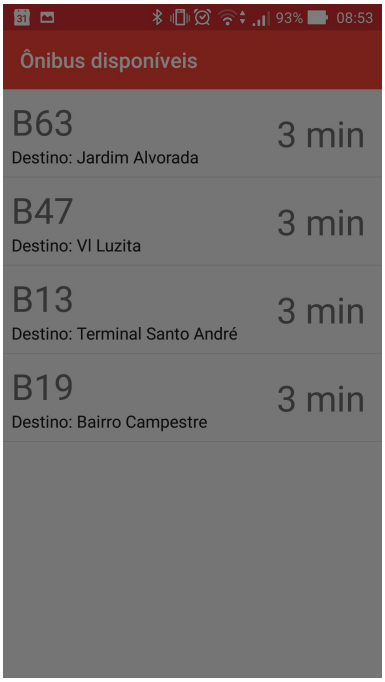

Figura 17 – Tela com lista de ônibus disponíveis do aplicativo móvel.

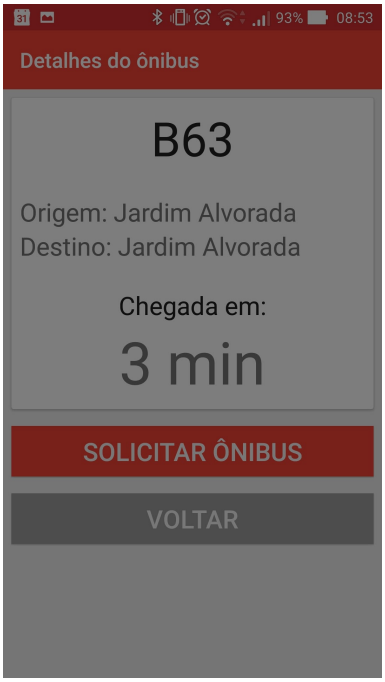

Figura 18 – Tela com detalhes do ônibus do aplicativo móvel.

3.4.2 IDE

Foi escolhido o Android Studio como IDE do projeto. Ela é desenvolvida pela IntelliJ e tida pelo Google como ferramenta oficial de desenvolvimento para aplicativos Android.

3.4.3 Linguagem

O Google tem duas linguagens de primeiro nível para desenvolvimento Android: Java e Kotlin. Para esse projeto adotamos a linguagem Kotlin, que possui sintaxe muito simplificado em comparação ao Java. Embora Java tenha sido a primeira linguagem oficial para desenvolvimento, Kotlin oferece acesso aos mesmo recursos do sistema. Algumas bibliotecas disponíveis, desenvolvidas por terceiros, ainda não migraram para o Kotlin, obrigando a implementar algumas classes em Java. Como Kotlin tem interoperabilidade com Java, não existe nenhum impeditivo de utilizar Kotlin e eventualmente alguma classa Java.

3.4.4 Arquitetura

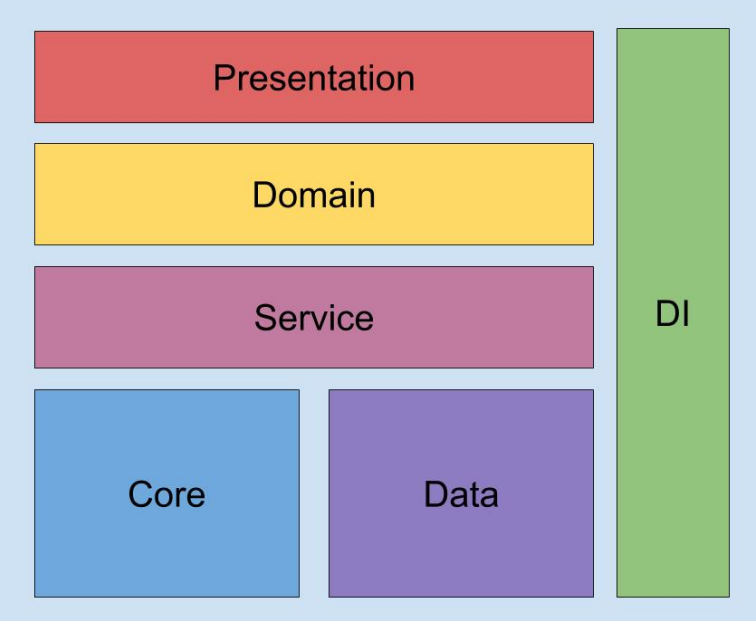

Figura 19 – Diagrama de blocos do aplicativo móvel.

Para desenvolvimento do software, foi adotado o padrão *Clean Architecture*. É um padrão que visa um maior desacoplamento das classes e distruibui bem as responsabilidades.

Core Não contém nenhuma lógica de negócio. Esta camada provê informações comuns, como configurações estáticas da placa a toda a aplicação. Possui também algumas classes e interfaces bases.

Data Responsável por prover dados para toda aplicação. Ela adota o Padrão de Arquitetura *Repository*, tendo uma interface de acesso aos dados. Uma grande vantagem em utilizar essa camada com esse padrão de arquitetura, é o respeito a responsabilidade única, um dos princípios do *SOLID*. Ela encapsula toda lógica de busca de dados, assim, caso uma classe precise de algum dado específico, ela solicita através da interface de comunicação e a classe que implementa a interface, cuida de toda lógica de busca de dado, seja um dado armazenado localmente, em cache ou em um servidor remoto. Tudo fica transparente para a classe que solicitou o dado.

Service Provê serviços para qualquer camada. No caso do aplicativo, a implementação do serviço de voz fica neste pacote e é injeta pelo pacote de Injeção de Dependências.

Domain Esta camada encapsula toda regra de negócios da aplicação. Toda vez que é necessário realizar processamentos em dados para satisfazer funcionalidades, é feito por esta camada.

Presentation Responsável por toda interface gráfica. Toda lógica de criação de telas e interceptação de interações do usuário com o aplicativo, é feito aqui. Quando é necessário procurar dados para exibir ao usuário, é feito solicitações deles para a camada Domain ou Data para que seja possa exibir os dados.

DI Este projeto utiliza o padrão de arquitetura *Injeção de Dependências*. Esta camada provê todas dependências, fazendo a implementação mais limpas nas outras classes, já que não precisam saber como instânciar uma classe, apenas usam.

3.4.5 Áudio Descrição

Uma das funcionalidades do aplicativo é descrição da tela que o deficiente está. O *TalkBack* fala para o usuário em qual componente ele está tocando, porém, não descreve em qual tela ele acabou de entrar. A implementação por áudio descrição foi simples com uso da API nativa *TextToSpeech*, onde podemos passar textos personalizados e o serviço se encarrega de sintetizar a voz.

O uso de uma camada de DI (Injeção de Dependências) facilitou o processo de implementação, fazendo ela na camada de serviço e configurando a instanciação no padrão *Singleton* para que todos que vão utilizar (nesse caso são os *presenters*), apenas solicitem a instância sendo passada por construtor.

3.4.6 Usabilidade

É uma boa prática no desenvolvimento de softwares, sempre confirmar se o usuário tem certeza que deseja executar alguma alteração que possa ter algum impacto no sistema ou em funcionalidades. Inicialmente foi pensado em usar um *dialog* para que o usuário confirme a ação de adicionar um ônibus como favorito, na tela de confirmação de acompanhamento. Ao testar a aplicação funcionando com *Talkback*, foi observado que o sistema descreve o botão com o seguinte texto: *"Favoritar botão, para acionar toque duas vezes"*. Este texto já faz o usuário se assegurar da sua ação, fazendo a prática de colocar um *dialog* ser algo desnecessário que só faz o usuário ter um trabalho a mais de deslizar o dedo pela tela para encontrar os botões de *OK* e cancelar do *dialog.*

3.5 Web service

3.5.1 Linguagem

Para o desenvolvimento do *web service* foi escolhida a utilização da pilha MEAN, que engloba quatro tecnologias para desenvolvimento *web* que possuem como base a linguagem JavaScript.

3.5.1.1 JavaScript

JavaScript é uma linguagem de programação *client-side*, utilizada para manipular os comportamentos de uma página, controlando o HTML e o CSS. Outra característica dela, é que ela é uma linguagem orientada à eventos. Para explicar melhor o que são eventos, é importante citar que uma página HTML utiliza tags para representar seus elementos, podendo conter menus, botões e formulários em seu corpo. Cada elemento possui alguns atributos, sendo alguns desses atributos de eventos, como por exemplo o *onClick* que realiza alguma função caso o elemento referente seja clicado pelo usuário. Tais funções podem ser desenvolvidas em JavaScript, entrando aqui para dizer qual comportamento a página terá ao disparo do evento.

3.5.2 MEAN stack

A pilha *MEAN* é um conjunto de *frameworks* desenvolvidos em JavaScript, que englobam os lados do cliente, do servidor e do banco de dados. Por possuírem a mesma linguagem como base, os elementos dessa pilha contam com uma maior produtividade no desenvolvimento. MEAN é um acrônimo para MongoDB, Express, Angular e Node.

3.5.2.1 MongoDB

É um banco de dados não relacional com uma escalabilidade muito boa. Ele utiliza conceitos de *collections* e *documents* em sua construção. As *collections* são equivalentes aos bancos de um ambiente que utiliza o SQL. Já os *documents*, se equivalem aos registros de cada banco. Os dados são guardados em arquivos similares aos de formato JSON (*JavaScript Object Notation*). Outro item importante sobre o MongoDB é o fato de ser schemaless, tornando-o bem flexível em relação a inclusão de dados diferentes em uma mesma *collection*, fazendo com que a validação de dados fique nas mãos dos desenvolvedores.

3.5.2.2 Express

É um framework que ajuda na organização de sua aplicação, caso use a arquitetura MVC, no lado do servidor. Uma de suas funções é a de facilitar a criação e manutenção de rotas, realizando uma configuração inicial com os caminhos para os *controllers, models* e *views* utilizados pela sua aplicação, além de informar os dados de configuração do servidor.

3.5.2.3 Angular

Framework utilizado no lado do cliente. Possui um conjunto adicional de atributos para as páginas HTML, passando parte do processamento dos dados da página para o lado do cliente. Isso possibilita a criação de interfaces dinâmicas e assíncronas além de diminuir a carga de processamento do servidor.

3.5.2.4 Node

Plataforma principal para o funcionamento da pilha MEAN. Ele utiliza o gerenciador de pacotes npm para organizar as bibliotecas utilizadas pela sua aplicação. É ele que realiza a conexão com servidores e diz quais bancos de dados serão utilizados pela aplicação.

3.5.3 Dificuldades

Apesar de conter conceitos simples de aprender, devido à grande quantidade de métodos para se realizar os mesmos processos, fica um pouco difícil para assimilar quais os arquivos que devem ser modificados para o funcionamento adequado da aplicação. Primeira dificuldade surgiu ao utilizar o mongoose, uma solução baseada em *schemas* para o banco de dados MongoDB que cuida de validações e tipagem de dados, resolvido ao criar arquivos separados para cada *collection* do banco.

3.5.4 Referências

Atributos de eventos

Guia introdutório sobre JavaScript

ALMEIDA, Flávio. MEAN - Full stack JavaScript para aplicações web com MongoDB, Express, Angular e Node. ed. Casa do Código, 2016.

Falar sobre o MEAN stack e sobre as dificuldades que estou encontrando sobre como trabalhar com cada tecnologia da pilha.