CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE FRANCA

“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LINYKEER RENATO SILVA ALMEIDA

VINÍCIUS GARCIA MORGAN DE AGUIAR

Desenvolvimento de Aplicativo Mobile para Segurança do Transporte Escolar

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Ely Fernando do Prado

FRANCA/SP

2020

Ficha catalográfica

S586s Silva, Linykeer Renato Silva Almeida e Vinicius Garcia Morgan de Aguiar
Safe School Bus: desenvolvimento de aplicativo Mobilepara segurança do transporte escolar
Linykeer Renato Silva Almeida e Vinicius Garcia Morgan de Aguiar / / [s.n], 2020

28 f.; 30 cm; il
Trabalho de Graduação (Curso Superior de Análise e Desenvolvimento de Sistemas) Fatec - Faculdade de Tecnologia “Dr. Thomaz Novelino”.
Orientador: Prof. Me. Ely Fernando do Prado

1. Aplicativo. 2. Motorista. 3. Localização. 4.Segurança. 5. Responsável. I. Autor. II. Título.
.

CDD – 005.1

Linykeer Renato Silva Almeida

Vinícius Garcia Morgan de Aguiar

Resumo

Este projeto apresenta o desenvolvimento do aplicativo Safe-School-Bus que foi projetado para aumentar a visibilidade dos motoristas de transporte escolar, e aumentar a segurança de seus passageiros. O aplicativo busca facilitar a vida tanto do motorista tanto do responsável do aluno, para isso o responsável terá acesso a uma lista de motoristas, escolhendo o motorista o aluno será vinculado ao mesmo, e assim tendo acesso a localização em tempo real do aluno. Já o motorista terá acesso a uma lista de todos seus alunos, melhorando o controle do seu negócio, além de poder visualizar a rota de cada aluno, facilitando seu trajeto. Sempre que o motorista embarcar ou desembarcar o aluno o responsável receberá uma atualização do status, melhorando assim a segurança do transporte escolar.

Palavras-chave: Aplicativo, Motorista, Localização, Segurança, Responsável

Abstract

This project presents the development of the Safe-School-Bus application that was designed to increase the visibility of school transport drivers, and to increase the safety of their passengers. The application seeks to make life easier for both the driver and the student's guardian, so the guardian will have access to a list of drivers, choosing the driver the student will be linked to, and thus having access to the student's real-time location. The driver, on the other hand, will have access to a list of all his students, improving control of his business, in addition to being able to visualize the route of each student, facilitating his journey. Whenever the driver embarks or disembarks the student, the guardian will receive a status update, thus improving the safety of school transport.

Keywords: *Application, Driver, Location, Security, Responsible*

1 Introdução

Atualmente o serviço de transporte escolar proporciona uma comodidade para os pais e alunos. Assim, o transporte escolar vem para suprir uma lacuna importante no convívio familiar, que é o de proporcionar maior tempo aos pais, terceirizando o serviço de transporte de seus filhos. Ainda assim o transporte escolar tem um

potencial muito grande a crescer, visto que de 50,5 milhões de matriculados na rede de ensino, apenas 802,8 mil alunos vão para a escola em transportes escolares.

Com a tecnologia podemos ter acesso em tempo real de tudo, principalmente dos locais onde os passageiros desembarcam e onde embarcam, utilizando algumas ferramentas como mapas e geolocalização utilizados em aplicativos como uber podemos aderir mais alunos para os motoristas e aumentar ainda mais a segurança para os alunos e passageiros.

Hoje em dia existem vários problemas em relação a transportes escolares, pode se considerar um deles os motoristas que ficam dependentes de ter alunos e passageiros, isso é relatado por vários motoristas, que alegam ter muita concorrência e pouca demanda, por isso criar um diferencial no seu serviço seria fundamental, o projeto Safe-School-Bus tem o objetivo de criar este diferencial, com o aplicativo o motorista terá muito mais argumentos para convencer o cliente a fechar negócio com ele. Outro problema são os responsáveis preocupados que não deixam seus filhos aderirem o transporte escolar por medo de não desembarcarem nos locais corretos, ou até mesmo no pior dos casos serem sequestrados, novamente o projeto Safe-School-Bus vem ajudar neste ponto, visto que com o aplicativo os responsáveis terão acesso a localização em tempo real do transporte que seu filho está.

Com estás considerações, a solução proposta é de alavancar o mercado de transporte escolar utilizando uma das principais influências atualmente e que está frequentemente em nosso dia a dia, que é o celular. A partir deste ponto desenvolver um aplicativo que acompanha a trajetória do veículo que transporta seus filhos seria a solução ideal para os pais terem mais confiança no serviço, além de ajudar os motoristas na localização de endereços. Hoje em dia, por meio de um aplicativo é possível desenvolver diversas funcionalidades, agilizando o serviço e colaborando com as dificuldades que temos cotidianamente.

2 Levantamento de Requisitos

2.1 Elicitação e especificação dos Requisitos

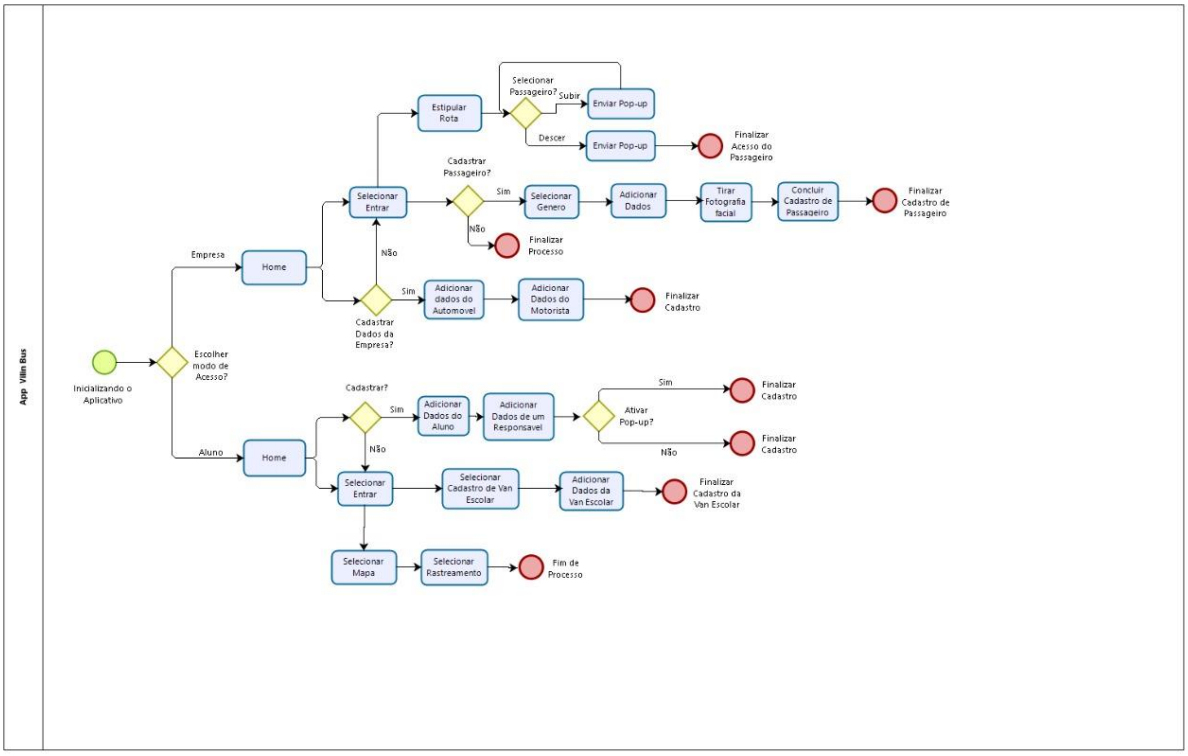
A elicitação foi realizada por meio de pesquisas em sala de aula, e com entrevistas de pais e motoristas de transportes públicos. Foi identificado como principal dificuldade dos motoristas em sempre terem número de alunos suficiente para lotar a capacidade do veículo. Outro problema identificado foi em relação aos pais que ficam sempre preocupados com a segurança de seu filho, sem saber exatamente onde ele está e por onde está indo. Durante a entrevista com pais e motoristas foi identificado que um aplicativo que fosse capaz de mostrar a posição atual do aluno traria maior segurança para os pais e consequentemente faria com que o motorista tivesse uma maior procura por seus serviços.

2.2 BPMN

A notação BPMN (**Business Process Model and Notation**) é formada por ícones que servem para desenhar o fluxo do processo. Ou seja, retratam a forma como o seu processo funciona. Ela é como a notação de fluxograma, o que muda em relação as duas são os ícones e algumas regras do desenho.

Na imagem abaixo (Figura 1), podemos ver como representamos o processo do BPMN.

Figura 1 -BPMN



2.3 Requisitos Funcionais

Quadro 1 – Requisitos Funcionais do sistema

RF001-Cadastro de Motorista	Categoria: () Oculto (X)Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: O sistema deve receber o cadastro da empresa (motorista), com os dados do veículo. E-mail, senha, confirmação de senha, número de telefone, nome do motorista, data nascimento, cpf, tipo de transporte, placa, ano, e a capacidade		
RF002-Cadastro de Responsável	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: O sistema deve receber o cadastro do responsável e do aluno, utilizando e-mail, senha e confirmação de senha, número de telefone, nome do aluno, data nascimento, o endereço do aluno e o endereço da escola.		
RF003- Login	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: O sistema deve receber o e-mail e senha do usuário. Validando se o usuário consta na base dados, se sim, o sistema o redirecionará para Home.		
F004-Visualizar alunos	Categoria:	Prioridade:

	() Oculto (X)Evidente	(X) Altíssima () Alta () Média () Baixa
Descrição: O sistema deverá mostrar uma lista para o motorista de todos os alunos filiados, contendo as informações do mesmo. Podendo controlar a ordem das rotas, e excluir os alunos indesejados.		
RF005-Atualizar Passageiro	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: O motorista irá atualizar o status do aluno sempre que o mesmo desembarcar do veículo.		
RF008-Visualizar percurso (motorista)	Categoria: () Oculto (X)Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: O motorista terá acesso ao mapa para verificar as rotas do aluno selecionado		
RF009-Visualizar percurso (responsável)	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: O responsável terá acesso ao mapa, podendo visualizar o status do aluno e onde se encontra em tempo real		
RF010-Vincular motorista	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: O responsável terá uma listagem de todos os motoristas disponíveis. Podendo vincular o motorista desejado		

2.4 Requisitos Não Funcionais

Quadro 2 – Requisitos Não Funcionais do sistema

RNF001-Cores	O sistema possui cor degrade, roxo e branco	Tipo	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF002-Menu	O sistema apresenta um menu lateral com um card. branco aonde fica os dados do usuário	Tipo	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF003-Animação Veiculo	O sistema possui animação de um veiculo em movimento na tela de login	Tipo	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF004-Animação Cadastro	O sistema possui animação quando você clica em cadastrar	Tipo	(X) Desejável ( ) Obrigatório	(X) Permanente ( ) Transitório

RNF005-Tratamento de erros inputs	O sistema deverá tratar se o usuário colocou os dados corretamente nos inputs	Tipo	( ) Desejável (X) Obrigatório	( ) Permanente (X) Transitório

RNF006- Tratamento de erros gerais	O sistema deverá mostrar um toast de 5 segundos se ocorrer um erro	Tipo	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF007- Ilustrações	O sistema possui imagens como ilustrações	Tipo	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF008- Foto do usuário	O sistema mostrará a foto do perfil em várias ocasiões para facilitar a interação do usuário	Tipo	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório

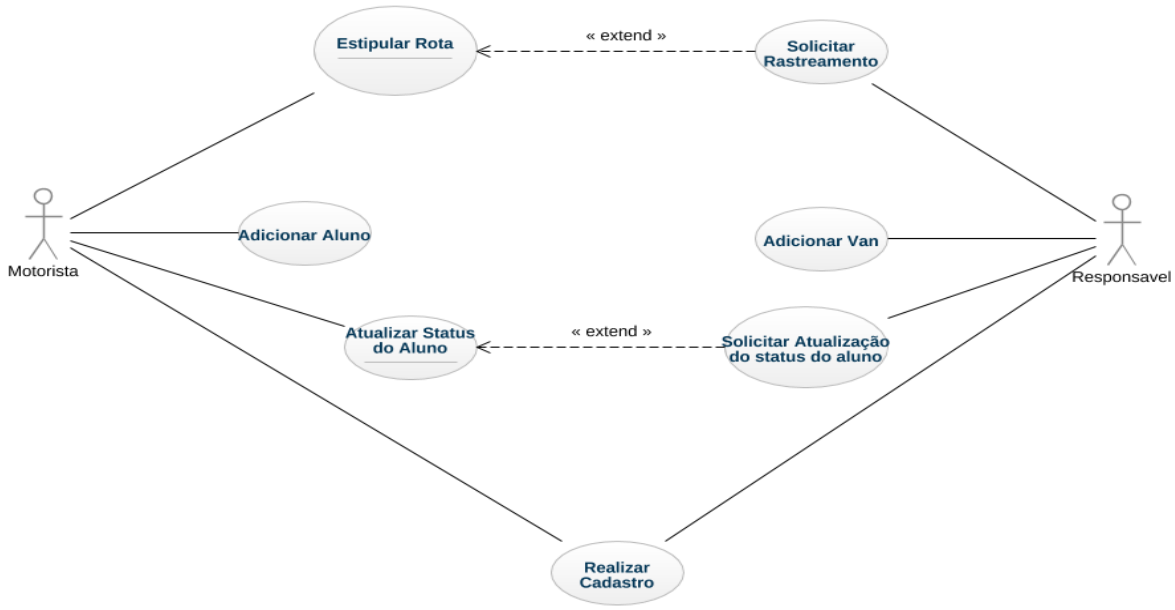
2.5 Regras de Negócio

Quadro 3 – Regras de Negócio do sistema.

RN001 – Cadastro Motorista
Descrição: Só serão permitidos cadastros que estiverem com todos os dados preenchidos
RN002 – Cadastro Responsável
Descrição: Só serão permitidos cadastros que estiverem com todos os dados preenchidos
RN003 – Adicionando Van
Descrição: Para o aluno ter acesso ao veículo a ser utilizado, o responsável terá que cadastrar o veículo dentro do sistema
RN004 – Vínculo de Passageiro
Descrição: Um passageiro somente poderá estar vinculado a um motorista.
RN005 – Vínculo de Motorista
Descrição: Um motorista poderá estar vinculado a vários responsáveis.

Um caso de uso representa uma **unidade discreta da interação entre um usuário (humano ou máquina) e o sistema**. Um caso de uso é uma unidade de um trabalho significante. Por exemplo: o " login para o sistema", "registrar no sistema" e "criar pedidos" são todos casos de uso.

Figura 2 – Caso de Uso



Quadro 4 – Use Case Cadastrar Usuários

Caso de Uso – Realizar Cadastro	
ID	UC 001
Descrição	Este caso de uso tem por objetivo cadastrar tanto o responsável do aluno quanto o motorista da van
Ator Primário	Responsável do Aluno e Motorista
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção cadastro de alunos, ou cadastro de motorista 2. O sistema carrega o formulário de cadastro 3. O usuário informa seus dados 4. O sistema verifica se esses dados já existem para assim requisitar ao back-end 5. O usuário confirma seus dados 6. O sistema gera um código token para ficar armazenado no contexto e facilitar na hora de realizar o login 7. O sistema redireciona de volta para tela de login quando o cadastro for bem sucedido.
Pós-condição	Nenhuma
Cenário Alternativo	

Caso de Uso – Vincular motorista	
ID	UC 002
Descrição	Este caso de uso tem por objetivo vincular o aluno no motorista que irá levar em sua rota
Ator Primário	Motorista
Pré-condição	O responsável ter se cadastrado no sistema
Cenário Principal	1. O use case inicia quando o responsável seleciona a opção vincular motorista

	2. O sistema carrega uma listagem de motoristas 3. O responsável escolhe seu motorista 4. O sistema vincula o aluno ao motorista
Cenário Alternativo	1. O responsável pode fazer a busca pelo seu motorista

Caso de Uso – Estipular Rota	
ID	UC 003
Descrição	Este caso de uso tem por objetivo o motorista estipular no mapa qual rota ele está realizando
Ator Primário	Motorista
Pré-condição	O responsável ter se cadastrado no sistema
Cenário Principal	1. O use case se inicia quando o motorista clica na opção visualizar percurso no aplicativo 2. O motorista seleciona o aluno que irá fazer a rota 3. Um percurso é traçado para o motorista
Pós-condição	Ter realizado login
Cenário Alternativo	

Caso de Uso – Atualizar Status do Aluno	
ID	UC 004
Descrição	Este caso de uso tem por objetivo o motorista atualizar em sua lista a situação atual do aluno
Ator Primário	Motorista
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o motorista escolhe a opção visualizar percurso 2. O motorista terá acesso a todos seus alunos 3. O motorista deverá clicar no aluno 4. O motorista deverá atualizar o status do aluno
Pós-condição	Ter realizado login
Cenário Alternativo	

Caso de Uso – Solicitar Rastreamento	
ID	UC 006
Descrição	Este caso de uso tem por objetivo a solicitação do responsável ao aplicativo, para ter o rastreamento da localização da van escolar
Ator Primário	Responsável do Aluno
Pré-condição	Nenhuma
Cenário Principal	1. O use case se inicia quando o responsável clica na tela de visualizar percurso 2. O sistema carrega um mapa com a localização exata de seu filho
Pós-condição	Nenhuma

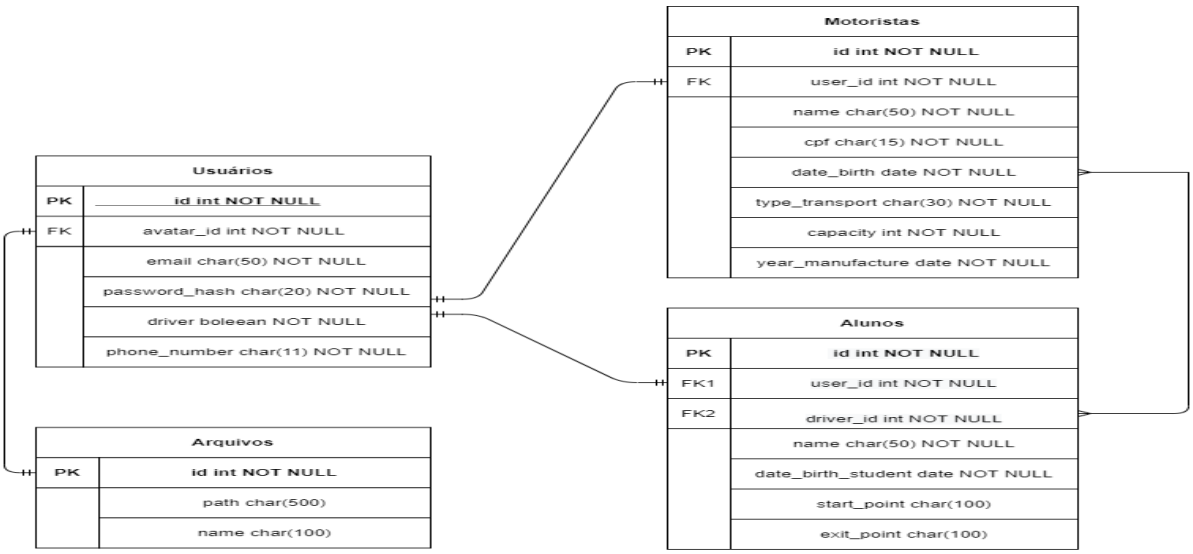
Cenário Alternativo	

Caso de Uso – Verificação de Status	
ID	UC 007
Descrição	Este caso de uso tem por objetivo mostrar o status do aluno
Ator Primário	Responsável
Pré-condição	Nenhuma
Cenário Principal	1. O use case se inicia quando motorista atualiza o status do passageiro, para assim o responsável receber se seu filho já subiu no veículo ou já foi entregue no seu destino 2. O responsável poderá visualizar o endereço do destino do seu filho
Pós-condição	Nenhuma
Cenário Alternativo	

2.6 Diagrama Entidade-Relacionamento

Em engenharia de software, um modelo entidade relacionamento é um modelo de dados para descrever os dado ou aspectos de informação de um domínio de negócio ou seus requisitos de processo, de uma maneira abstrata que em última análise se presta a ser implementada em um banco de dados, como um banco de dados relacional.

Figura 3 – Diagrama Entidade-Relacionamento



3 Ferramentas e Métodos ou Desenvolvimento

3.1 Ferramentas

Nesta seção são apresentadas as ferramentas utilizadas para desenvolver o projeto de aplicativo para segurança do transporte escolar. Dentre estas ferramentas destacam-se as linguagens de programação de frameworks utilizados.

3.1.1 React Native

Segundo Frias (2019) o React [1] possui um poderoso algoritmo de manipulação do DOM, o que se torna um grande diferencial. Com o VDOM, o React, atualiza apenas os elementos necessário na tela, sem que tenha um recarregamento da tela toda, melhorando assim tempo de 107 carregamento e usabilidade dos usuários. Pereira (2018) define VDOM como “um conceito de programação onde existe uma representação da UI cacheada em memória, sincronizada com o DOM real do browser, com a utilização do ReactDOM”

3.1.2 NodeJS

Node.js [2] é um interpretador, com código aberto, de código JavaScript de modo assíncrono e orientado a eventos, focado em migrar a programação do JavaScript do lado do cliente para os servidores, criando assim aplicações de alta escalabilidade (como um servidor web), capazes de manipular milhares de conexões/requisições simultâneas em tempo real, numa única máquina física [NODE.JS 2019]. O Node.js é baseado no interpretador V8 JavaScript Engine (interpretador de JavaScript open source implementado pelo Google em C++ e utilizado pelo Chrome). Foi criado por Ryan Dahl em 2009, e seu desenvolvimento é mantido pela fundação Node.js em parceria com a Linux Foundation [NODE.JS 2019]. Segundo uma pesquisa realizada por um popular fórum de desenvolvedores, pelo sétimo ano consecutivo, o JavaScript é a linguagem de programação mais usada [Stack Overflow 2019].

---

[1] https://reactnative.dev/
[2] https://nodejs.org/en/

3.1.3 Sequelize

Sequelize[3] é um Node.js ORM baseado em promessa para PostgreSQL, MySQL, MariaDB, SQLite e Microsoft SQL Server. Ele oferece suporte a transações sólidas, relações, carregamento rápido e lento, replicação de leitura e muito mais.

3.1.4 PostgreSQL

O PostgreSQL[4] é um dos resultados de uma ampla evolução que se iniciou com o projeto Ingres, desenvolvido na Universidade de Berkeley, Califórnia. Após seu retorno a Berkeley, em 1985, Stonebraker começou um projeto pós-Ingres com o objetivo de resolver problemas com o modelo de banco de dados relacional.

3.1.5 Heroku

Heroku[5] estreou a versão comercial de sua solução de hosting de Rails na última semana, depois de uma etapa gratuita que durou por um ano. Eles descrevem seu serviço como um "provisionless deployment" porque opera e escala automaticamente, sem nenhuma administração do sistema. Enquanto ele é mais caro que suas alternativas e (por enquanto) é baseado na cloud da Amazom EC2 com as limitações de SLA que o acompanham, nós pensamos que eles têm uma sólida oferta que compensa uma análise mais detalhada.

---

[3] https://sequelize.org/
[4] https://www.postgresql.org/
[5] https://www.heroku.com/

3.1.6 AWS

Em 2006, a Amazon Web Services (AWS[6]) começou a oferecer serviços de infraestrutura de TI para empresas por meio de serviços web – hoje conhecidos como computação em nuvem. Um dos principais benefícios da computação em nuvem é a oportunidade de substituir diretamente gastos com a infraestrutura principal por preços variáveis baixos, que se ajustam de acordo com sua empresa. Com a Nuvem, as empresas não precisam mais planejar ou adquirir servidores, assim como outras infraestruturas de TI, com semanas ou meses de antecedência. Em vez disso, podem instantaneamente rodar centenas de milhares de servidores em minutos e oferecer resultados mais rapidamente.

Atualmente, a Amazon Web Services oferece uma plataforma de infraestrutura altamente confiável, escalável e de baixo custo na nuvem que potencializa centenas de milhares de empresas em 190 países ao redor do mundo. Com datacenters localizados nos EUA, Europa, Brasil, Cingapura, Japão e Austrália, clientes de todos os setores estão tendo vantagens com os seguintes benefícios.

3.2 Métodos ou Desenvolvimento

O que é Back-End? Back-End, como o próprio nome sugere, vem da ideia do que tem por trás de uma aplicação. Pode ficar meio abstrato em um primeiro momento, mas pense que para conseguir usar o Facebook no dia a dia, os dados do seu perfil, amigos e publicações precisam estar salvos em algum lugar, sendo esse lugar um banco de dados.

O que é Front-End? Front-End é basicamente a interface do usuário, sendo tudo o que o usuário pode ver e interagir com toda a “frente” do software. Há então as camadas de interações do usuário, ou seja, efeitos, animações. Toda interatividade que existe em uma página, ao clicar em algo e ter um alerta, por exemplo, todo fluxo da aplicação e toda a camada visual é o Front-End.

[6] https://aws.amazon.com/pt/?nc2=h_lg

3.2.1 Back-End

O desenvolvimento do back-end foi feito com a ferramenta Node.js, com auxílio de outras aplicações.

No arquivo principal, “app.js” a classe App inicia com um construtor que chama o servidor Express, junto com ele também está uma função de middlewares, para tratar as requisições, e outra função de routes, que recebe como parâmetro todas as rotas da aplicação, conforme pode ser visualizado na Figura 4.

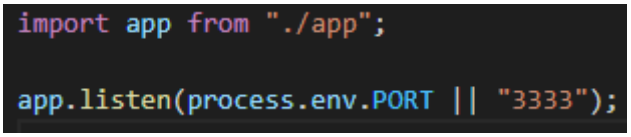
Figura 4 – Arquivo app.js

```
class App {
  constructor() {
    this.server = express();
      ViniciusGMorgan, 3 months ago • Add files via upload
    Sentry.init(sentryConfig);
    this.middlewares();
    this.routes();
    this.exceptionHandler();
  }

  middlewares() {
    this.server.use(Sentry.Handlers.requestHandler());
    this.server.use(cors());
    this.server.use(express.json());
    this.server.use(
      "/files",
      express.static(path.resolve(__dirname, "..", "tmp", "uploads"))
    );
  }

  routes() {
    this.server.use(routes);
    this.server.use(Sentry.Handlers.errorHandler());
  }
```

Fonte: Os Autores

No arquivo “server.js” importamos a classe app para iniciar o servidor na porta desejada, conforme mostrado abaixo.

Figura 5 – Arquivo server.js

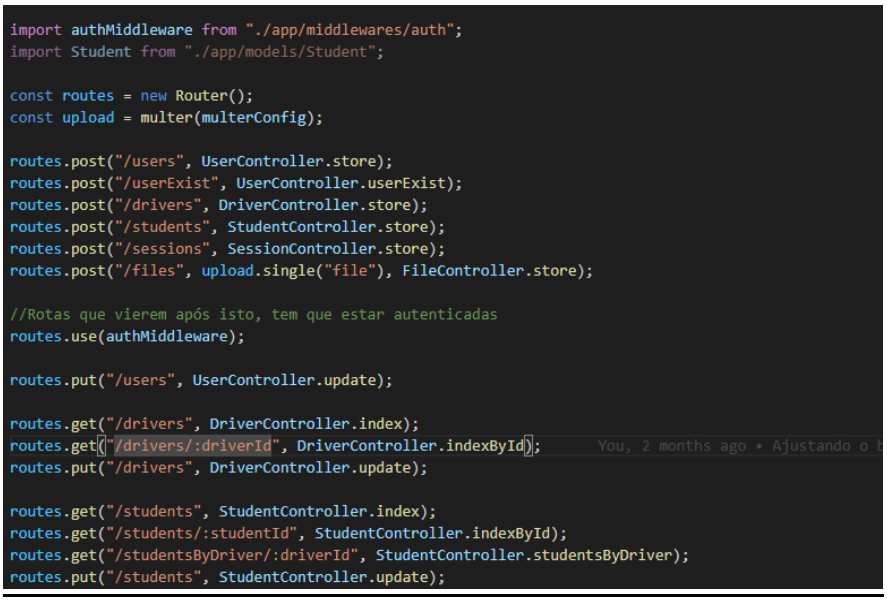


```
import app from "./app";

app.listen(process.env.PORT || "3333");
```

Fonte: Os Autores

Veja como as rotas são executadas no arquivo “routes.js”. É importante ressaltar que a partir do momento que a função “authMiddlewares” é usada todas as rotas que vierem após ela tem que estar autenticada com o token.

Figura 6 – Arquivo route.js



```
import authMiddleware from "./app/middlewares/auth";
import Student from "./app/models/Student";

const routes = new Router();
const upload = multer(multerConfig);

routes.post("/users", UserController.store);
routes.post("/userExist", UserController.userExist);
routes.post("/drivers", DriverController.store);
routes.post("/students", StudentController.store);
routes.post("/sessions", SessionController.store);
routes.post("/files", upload.single("file"), FileController.store);

//Rotas que vierem após isto, tem que estar autenticadas
routes.use(authMiddleware);

routes.put("/users", UserController.update);

routes.get("/drivers", DriverController.index);
routes.get("/drivers/:driverId", DriverController.indexById);      You, 2 months ago • Ajustando o i
routes.put("/drivers", DriverController.update);

routes.get("/students", StudentController.index);
routes.get("/students/:studentId", StudentController.indexById);
routes.get("/studentsByDriver/:driverId", StudentController.studentsByDriver);
routes.put("/students", StudentController.update);
```

Fonte: Os Autores

Como visto na Figura 6, as rotas chamam uma Controller, cujo as requisições são tratadas.

Atualmente o sistema contém 5 controllers, são elas: SessionController, UserController, StudentController, DriverController e FileController conforme mostrado na Figura 7.

Figura 7 – Arquivo SessionController.js

```
class SessionController {
  async store(req, res) {
    const schema = Yup.object().shape({
      email: Yup.string().email(),
      password: Yup.string().required(),
    });        ViniciusGMorgan, 3 months ago • Add files via upload
    if (!(await schema.isValid(req.body))) {
      return res.status(400).json({ error: "Verifique os campos enviados" });
    }
    if (!user) {
      return res.status(401).json({ error: "Usuário não encontrado!" });
    }
    if (!(await user.checkPassword(password))) {
      return res.status(401).json({ error: "A senha não corresponde" });
    }
    return res.json({
      user: {
        id,
        name,
        email,
        phone_number,
        driver,
        avatar,
        provider: provider,
      },
      token: jwt.sign({ id }, authConfig.secret, {
        expiresIn: authConfig.expiresIn,
      }),
    });
  }
}
```

Fonte: Os Autores

No arquivo database/index.js, fazemos a conexão com o Sequelize, e junto com ele um map de todas as models.

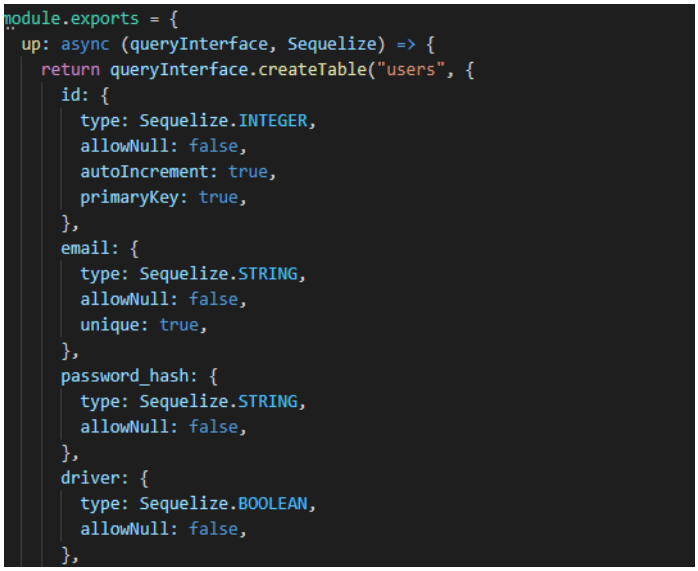
Figura 8 – Arquivo Database.js

```
const models = [User, File, Driver, Student];

ViniciusGMorgan, 3 months ago | 1 author (ViniciusGMorgan)
class Database {        ViniciusGMorgan, 3 months ago • Add files via upload
  constructor() {
    this.init();
  }
  init() {
    this.connection = new Sequelize(databaseConfig);

    models
      .map((model) => model.init(this.connection))
      .map(
        (model) => model.associate && model.associate(this.connection.models)
      );
  }
}

export default new Database();
```

Fonte: Os Autores

Ainda dentro de /database, temos uma pasta migrations. Migrations é uma forma de versionar o schema de sua aplicação, ela trabalha na manipulação da base de dados: criando, alterando ou removendo. Uma forma de controlar as alterações do seu banco juntamente com o versionamento de sua aplicação e compartilha-la, conforme mostrado na Figura 9.

Figura 9 – Arquivo migrations.js



```
module.exports = {
  up: async (queryInterface, Sequelize) => {
    return queryInterface.createTable("users", {
      id: {
        type: Sequelize.INTEGER,
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
      },
      email: {
        type: Sequelize.STRING,
        allowNull: false,
        unique: true,
      },
      password_hash: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      driver: {
        type: Sequelize.BOOLEAN,
        allowNull: false,
      },
```

Fonte: Os Autores

Com as migrations criadas, temos que ter as models para salvar os atributos, na figura 10, fazemos isto.

Figura 10 – Arquivo User.js

```
class User extends Model {
  static init(sequelize) {
    super.init(
      {
        email: Sequelize.STRING,
        password: Sequelize.VIRTUAL,
        password_hash: Sequelize.STRING,
        driver: Sequelize.BOOLEAN,
        phone_number: Sequelize.STRING,
      },
      {
        sequelize,
      }
    );

    this.addHook("beforeSave", async (user) => {
      if (user.password) {
        user.password_hash = await bcrypt.hash(user.password, 8);
      }
    });

    return this;
  }

  static associate(models) {
    this.belongsTo(models.File, { foreignKey: "avatar_id", as: "avatar" });
  }

  checkPassword(password) {
    return bcrypt.compare(password, this.password_hash);
  }
}
```

Fonte: Os Autores

4.2.2 Front-End

Para o desenvolvimento do front-end utilizamos a ferramenta React Native.

No arquivo principal “App.js”, pedimos a permissão de localização. Logo depois rende rizamos a Status Bar e todas as rotas do Aplicativo.

Figura 11 – Arquivo app.js

```
async requestLocationPermission() {
  try {
    const granted = await PermissionsAndroid.request(
      PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION,
      {
        title: 'Example App',
        message: 'Example App access to your location ',
      },
    );
    if (granted === PermissionsAndroid.RESULTS.GRANTED) {
      console.log('You can use the location');
      alert('You can use the location');
    } else {
      console.log('location permission denied');
      alert('Location permission denied');
    }
  } catch (err) {
    console.warn(err);
  }
}

async componentDidMount() {
  await requestLocationPermission();
}

render() {
  return (
    <View style={{flex: 1}}>
      <StatusBar backgroundColor={'#7159c1'} barStyle="light-content" />
      <Routes />
    </View>
  );
```

Fonte: Os Autores

Existem 7 rotas em todo o aplicativo, são elas: Login, Home, Cadastro de motorista, Cadastro de aluno, Vincular motorista, Visualizar Percurso, Visualizar alunos.

Em Login, podemos ver como é feita a autenticação, com o frontend se conectando na API conforme mostrado na Figura 12

Figura 12 – Arquivo login.js

```
async login() {
  const {user, password} = this.state;
  const params = {
    email: user,
    password: password,
  };

  let config = {
    method: 'post',
    data: params,
    url: '/sessions',
  };

  this.setState({loading: true});

  if (!user || !password) {
    this.setState({loading: false});
    return Toast.show({
      type: 'error',
      position: 'top',
      text1: 'Campos vazios ou inválidos',
      visibilityTime: 3000,
    });
  }

  api(config)
    .then((response) => {
      AsyncStorage.multiSet([
        ['motorista', JSON.stringify({motorista: response.data.user.driver})],
        ['user', JSON.stringify({user: response.data.user})],
        ['token', JSON.stringify({token: response.data.token})],
      ]).then(() => Actions.home());
    })
    .catch((error) => {
      return Toast.show({
        type: 'error',
        position: 'top',
        text1: handleError(error),
        visibilityTime: 3000,
      });
```

Fonte: Os Autores

Os cadastros de motoristas e alunos são feitos praticamente iguais. Eles passam por uma validação tanto no frontend como no back-end, para que o usuário não entre com dados irregulares.

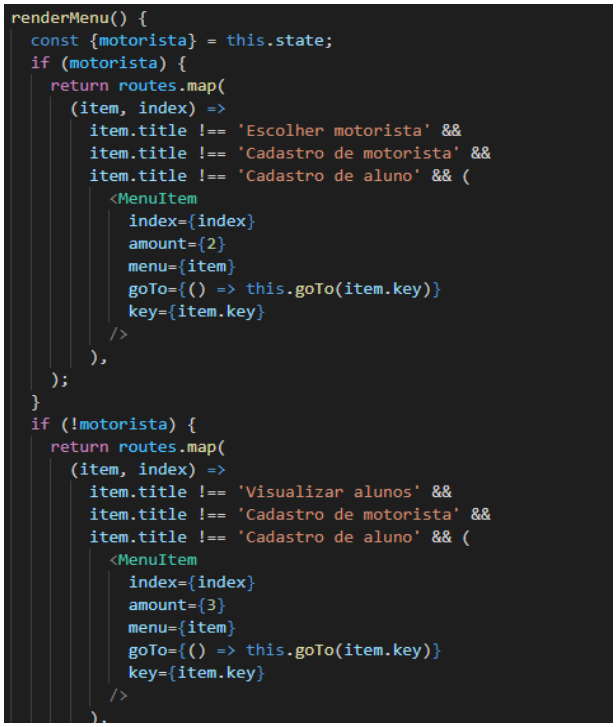
Na página Home é renderizado os menus correspondentes do usuário, diferenciando se ele é motorista ou responsável do aluno. Além disso é renderizado o “Drawer”, que é um Menu lateral onde o usuário pode visualizar e editar sua conta conforme mostrado na Figura 13.

Veja como é feito a diferenciação dos clientes, conforme mostrado na Figura 14.

Figura 13 – Arquivo home.js

Fonte: Os Autores

Figura 14 – Arquivo home.js



Fonte: Os Autores

No arquivo “Map.js” é possível ver como o Mapa é mostrado.

Figura 15 – Arquivo map.js

Fonte: Os Autores

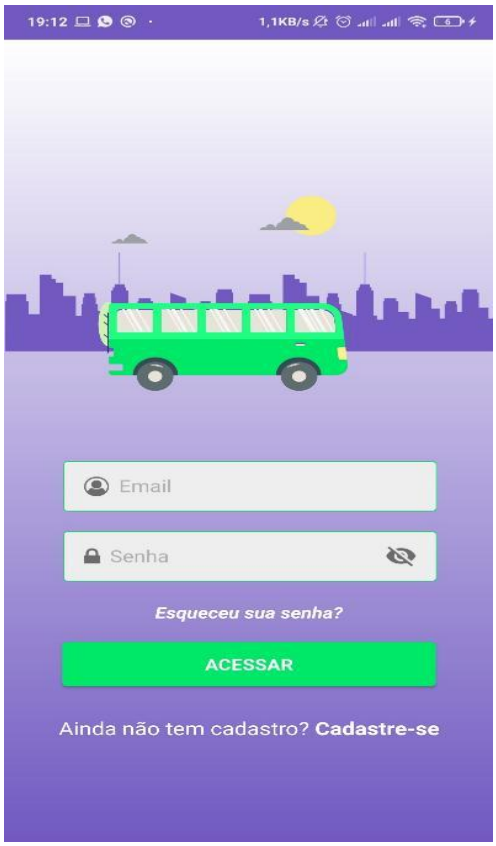
5 Resultados e Discussão

Neste capítulo é apresentado o aplicativo safe-school-bus conforme descrito o seu desenvolvimento nos capítulos anteriores.

Quando o usuário abre o sistema logo ele se depara com a tela de login. Se o usuário já tiver cadastrado ele pode logar, caso não, ele apertará em “Cadastre-se”, conforme mostrada a Figura 16.

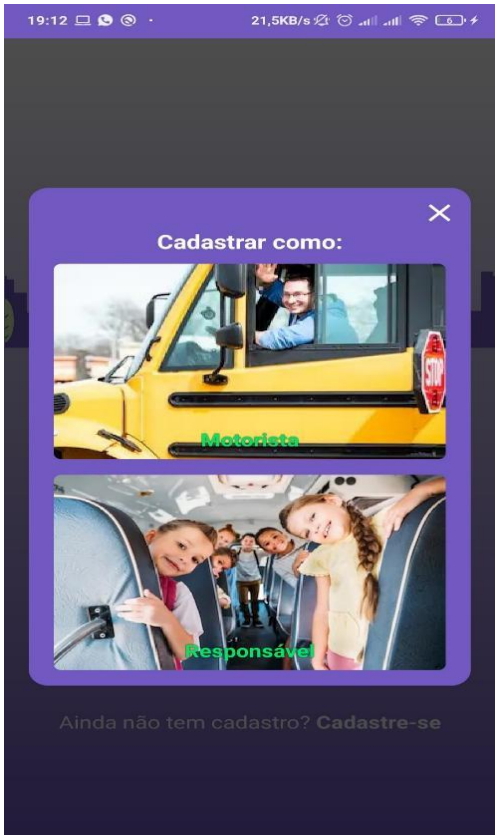
Quando o usuário clica em cadastro, ele deve escolher se deseja cadastrar um motorista ou um responsável/aluno, conforme mostrado na Figura 17.

Figura 16 – Login



Fonte: Os Autores

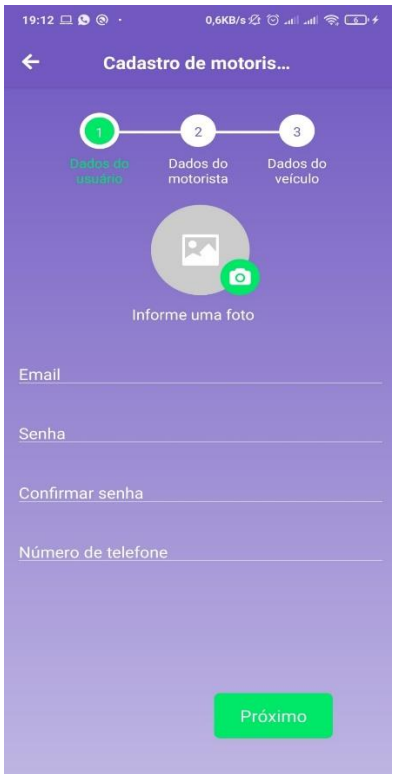
Figura 17 – Cadastrar como



Fonte: Os Autores

O cadastro de motorista é dividido em 3 etapas, dados do usuário, dados do motorista e dados do veículo conforme mostrado nas Figuras 18,19 e 20

Figura 18- Dados do usuário

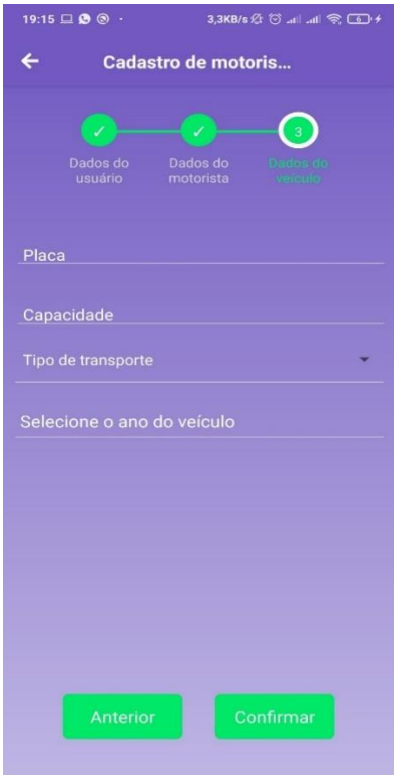


Fonte: Os Autores

Figura 19 – Dados do motorista

Fonte: Os Autores

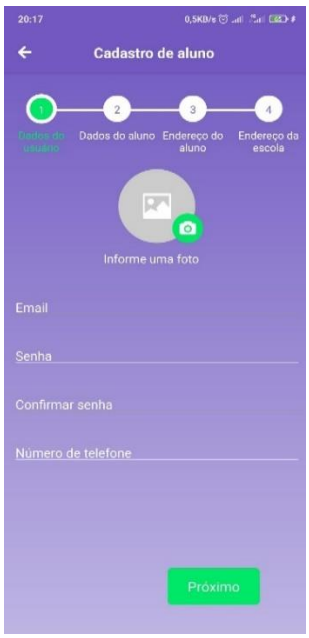
Figura 20 – Dados do veículo



Fonte: Os Autores

Já o cadastro de aluno é dividido em 4 etapas, dados do usuário, dados do aluno, endereço da escola, endereço do aluno, conforme mostrado nas Figuras 21,22, 23 e 24.

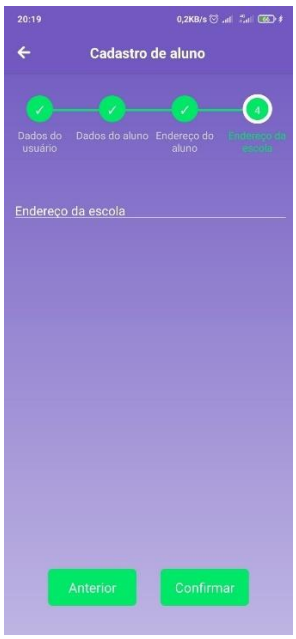
Figura 21 **Figura 22** **Figura 23** **Figura 24**



Fonte: Os Autores

Fonte: Os Autores

Fonte: Os Autores

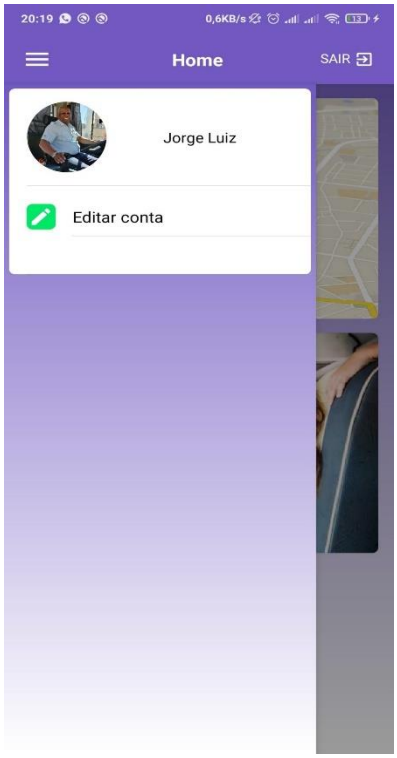


Fonte: Os Autores

Após o usuário fazer login ou se cadastrar ele é redirecionado a página “Home”. Nela tem os menus disponíveis do usuário, também existe um menu lateral, onde ele pode clicar para editar sua conta, conforme mostrado na figura 25

Se o usuário for motorista os menus disponíveis são Visualizar Percurso e Visualizar Alunos, conforme mostrando na figura 26. Já se o usuário for o responsável os menus disponíveis são Visualizar Percurso e Escolher Motorista. conforme mostrado na figura 27

Figura 25 – Menu lateral **Figura 26** – Home Responsável **Figura 27** – Home Motorista



Fonte: Os Autores

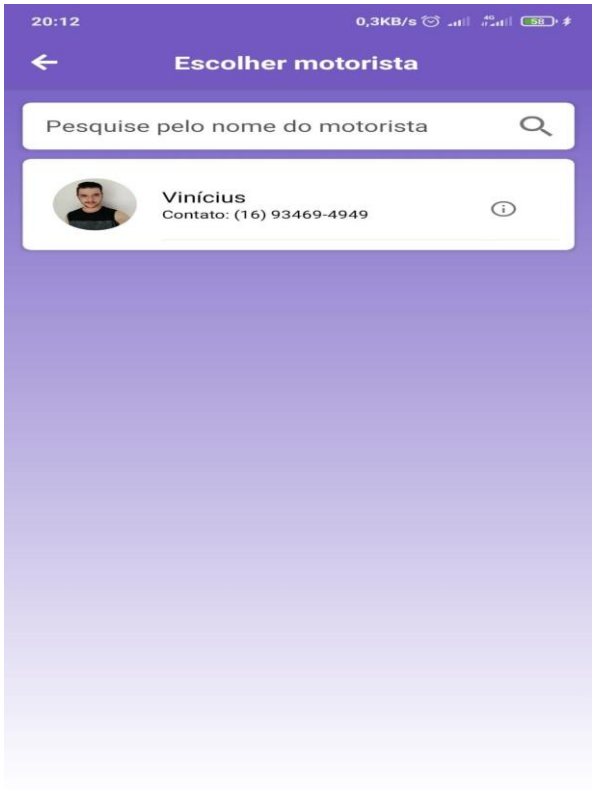
Fonte: Os Autores

Fonte: Os Autores

Em “Escolher Motorista” o usuário terá uma listagem dos motoristas disponíveis, e também poderá procurar pelo motorista conforme mostrado na Figura 28.

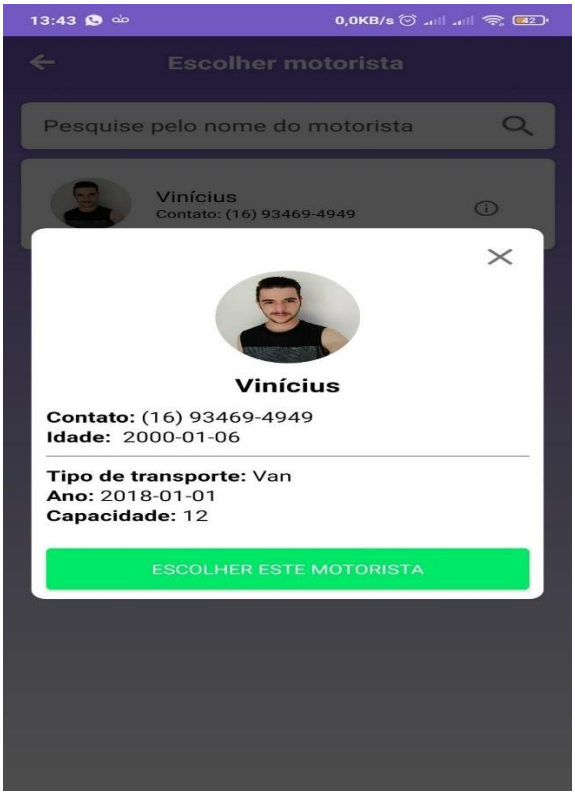
Ao escolher o motorista um modal abrirá, conforme mostrado na Figura 29, mostrando mais informações e um botão “Escolher motorista”, se o usuário confirmar o aluno será vinculado ao motorista.

Figura 28 – Escolher Motor



Fonte: Os Autores

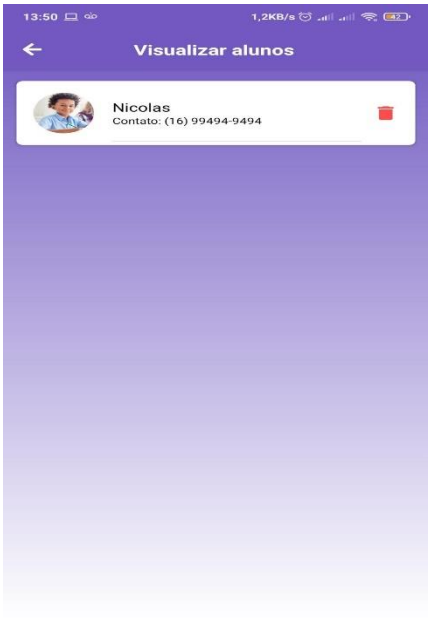
Figura 29 – Detalhes Motorista



Fonte: Os Autores

Já com o usuário motorista o menu “Visualizar Alunos” mostrará uma listagem de todos os alunos vinculado, conforme mostrado na Figura 30

Figura 30 – Visualizar Alunos

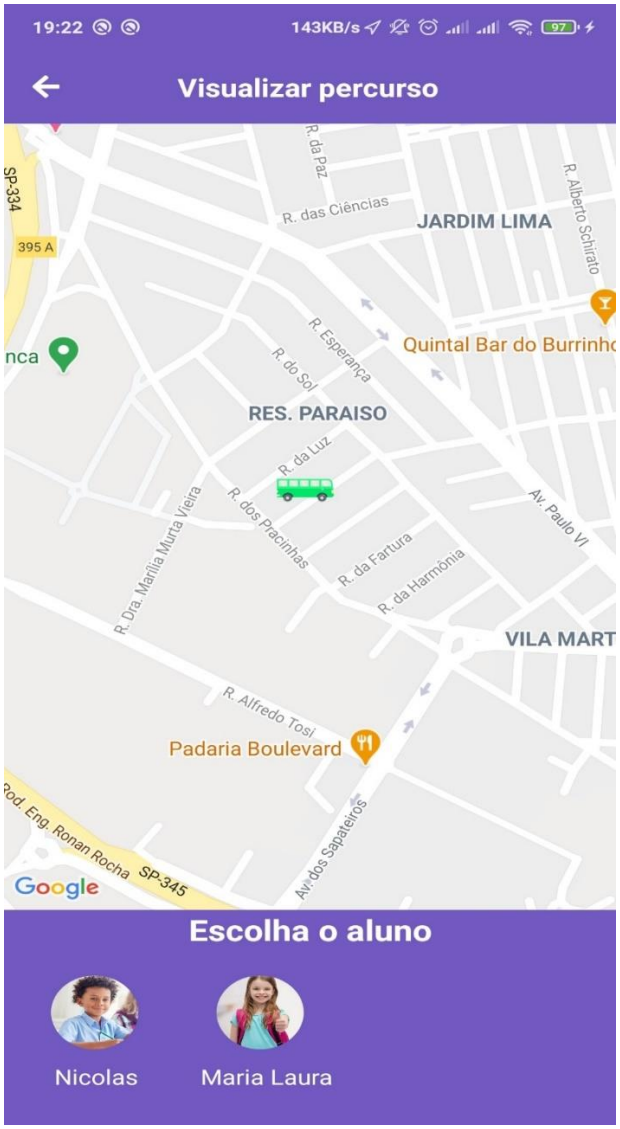


Fonte: Os Autores

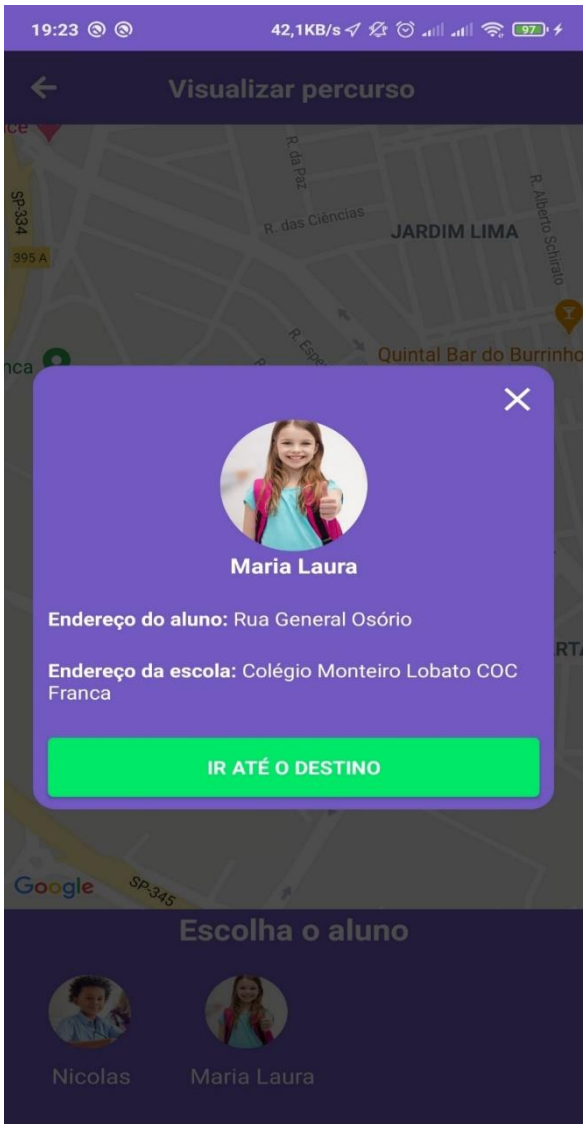
Na home do motorista terá uma opção visualizar percurso, que assim que o usuário clicar mostrará em um mapa a localização do transporte escolar, e também uma lista com os alunos vinculados ao motorista, ao clicar no aluno abrirá um modal que mostra as informações do aluno e um botão caso o motorista queira iniciar este percurso, conforme mostrado nas figuras 31 e 32.

Figura 31 – Visualizar Percurso Motorista

Figura 32 – Modal com informações do aluno



Fonte: Os Autores



Fonte: Os Autores

Se o motorista clicar no botão “Ir até o destino”, uma rota será traçada, indicando aonde o motorista tem que ir, conforme mostrado na figura 33.

Assim que o motorista concluir o percurso e embarcar o aluno ele deverá clicar no botão “Aluno já está no veículo”, feito isso, o sistema irá ilustrar com uma borda

verde e uma van ao lado do aluno, indicando que ele está no veículo, conforme mostrado na figura 34.

Figura 33 – Rota

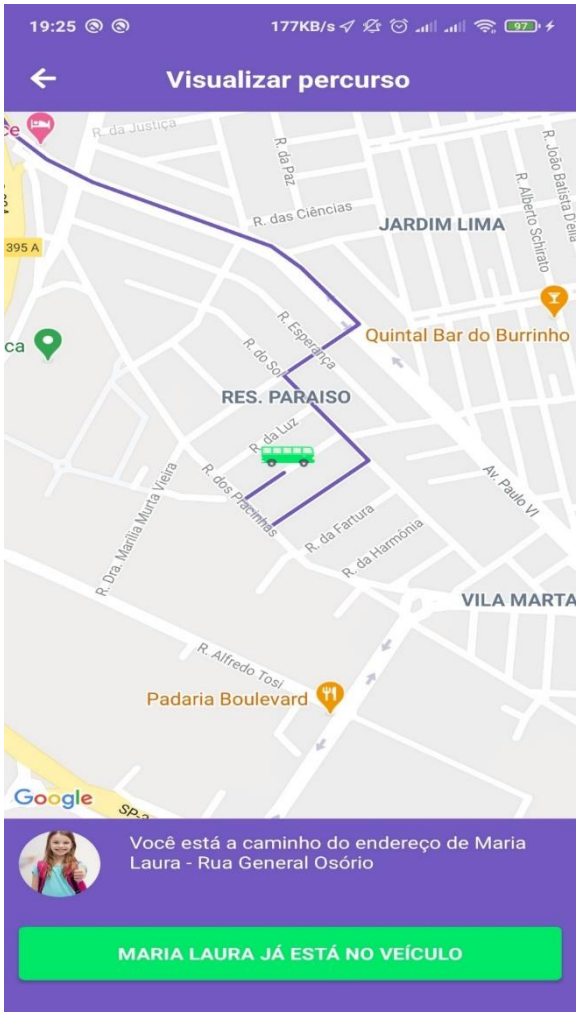
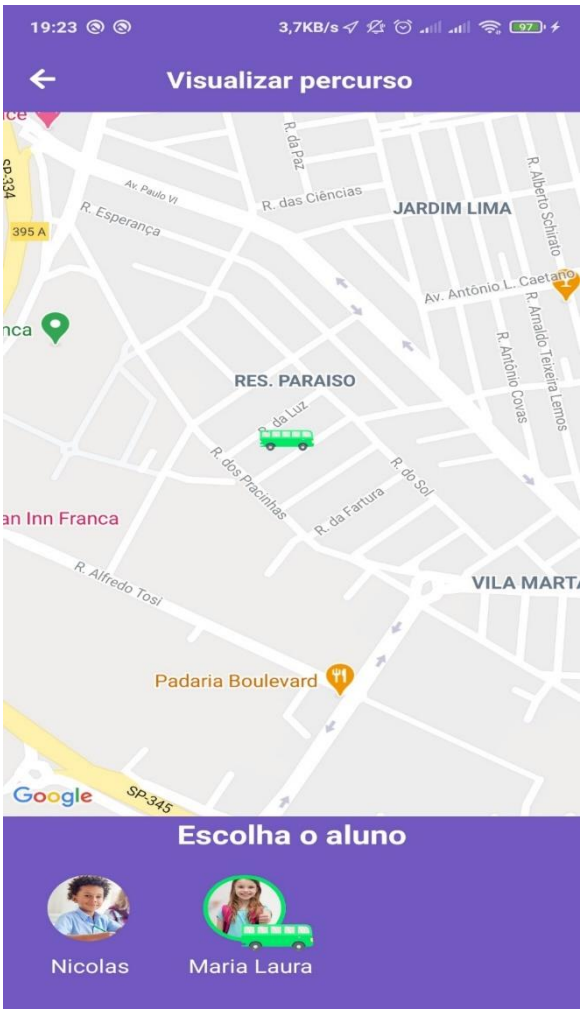


Figura 34– Aluno já no veiculo



Os mesmos procedimentos podem ser feitos na volta do percurso. Todas as mudanças de status do motorista serão mostradas em tempo real ao responsável, trazendo assim, a maior segurança ao seu filho.

Considerações finais

Concluímos que este projeto vem a contribuir muito a sociedade, principalmente no meio de transportes escolares. As hipóteses levantadas foram confirmadas e por isso acreditamos que conseguimos resolver a questão problema, que era resolver a segurança do transporte escolar, isso tudo por meio de um aplicativo, capaz de facilitar a vida do motorista e do responsável pelo aluno.

Vimos que com aplicativo em mãos o motorista poderá oferecer aos seus clientes um diferencial, alavancando assim o seu negócio, além disso o motorista terá um controle maior de seus alunos, cujo as informações vão estar toda no sistema. Já pelo lado do responsável, acompanhar por onde o motorista anda e saber se seu filho está a caminho da escola ou se já foi entregue, é crucial. Além disso, o responsável terá a sua disposição uma lista de motoristas para escolher a melhor opção, facilitando assim a busca de um motorista.

Em relação ao desenvolvimento do projeto, tivemos algumas considerações, o objetivo inicial era desenvolver algo em relação a transporte, tivemos como base a Uber que inovou o meio de transportes urbanos, a partir daí surgiu a ideia de desenvolver um aplicativo de transporte escolar, para inovar este meio também. Nosso maior desafio em relação ao projeto foi a experiência, e ao passar por estes desafios evoluímos muito, agregamos muito conhecimento que com certeza irá ajudar na nossa carreira profissional.

O desenvolvimento do presente estudo possibilitou uma análise de como este projeto pode ser melhorado, a partir do projeto foi constatado a descoberta de novos problemas, então decidimos que iremos aperfeiçoar esse projeto futuramente, reestruturando ele totalmente para uma estrutura melhor e mais leve, para disponibilizarmos para todos os clientes interessados e a todos os pais ou alunos que queiram ter mais segurança em seu transporte.

Referências

DE CAMARGOS, João Gabriel Colares et al. Uma Análise Comparativa entre os Frameworks Javascript Angular e React. **Computação & Sociedade**, v. 1, n. 1, 2019.

DANIELSSON, William. React Native application development. **Linköpings universitet, Swedia**, v. 10, p. 4, 2016.

GOULART, Angelito M.; RODRIGUES, Ricardo N. Análise Comparativa Entre Linguagens de Back End.

JACKSON, Keith R. et al. Performance and cost analysis of the Supernova factory on the Amazon AWS cloud. **Scientific Programming**, v. 19, n. 2-3, p. 107-119, 2011.

LEE, Bih-Hwang; DEWI, Ervin Kusuma; WAJDI, Muhammad Farid. Data security in cloud computing using AES under HEROKU cloud. In: **2018 27th Wireless and Optical Communication Conference (WOCC)**. IEEE, 2018. p. 1-5.

PEREIRA, Caio Ribeiro. Working with SQL Databases. In: **Building APIs with Node. js**. Apress, Berkeley, CA, 2016. p. 27-36.

STONEBRAKER, Michael; ROWE, Lawrence A.; HIROHAMA, Michael. The implementation of POSTGRES. **IEEE transactions on knowledge and data engineering**, v. 2, n. 1, p. 125-142, 1990.

TEIXEIRA, Pedro. **Professional Node. js: Building Javascript based scalable software**. John Wiley & Sons, 2012.