Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

WEB solution for the management of data related to the School Bus Routing Problem

Matheus Teixeira Lemos

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Rafael Frederico Alexandre

Julho, 2019
João Monlevade–MG

Matheus Teixeira Lemos

WEB solution for the management of data related to the School Bus Routing Problem

Supervisor: Rafael Frederico Alexandre

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

Universidade Federal de Ouro Preto

João Monlevade

Julho de 2019

L557w Lemos, Matheus Teixeira.
WEB solution for the management of data related to the School Bus Routing Problem [manuscrito] / Matheus Teixeira Lemos. - 2019.

50f.:

Orientador: Prof. Dr. Rafael frederico Alexandre.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Aplicações web. 2. Levantamentos de rotas. 3. Transporte escolar - ônibus . 4. Sistemas de recuperação da informação - Transportes. I. Alexandre, Rafael frederico. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.9

Catalogação: ficha.sisbin@ufop.edu.br

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

WEB solution for the management of data related to the School Bus Routing Problem

Matheus Teixeira Lemos

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:

[illegible]

Rafael Frederico Alexandre
Doutor em Engenharia Elétrica
DECSI – UFOP

[illegible]

Fernando Bernardes de Oliveira
Doutor em Engenharia Elétrica
Examinador
DECSI – UFOP

[illegible]

George Henrique Godim da Fonseca
Doutor em Engenharia Elétrica
Examinador
DECSI – UFOP

[illegible]

Mateus Ferreira Satler
Doutor em Tecnologías Informáticas Avançadas
Examinador
DECSI – UFOP

João Monlevade, 12 de julho de 2019

Dedicated to my family, that supported me from behind, to my friends, who stood at my side, and to my teachers, who guided me from ahead.

Acknowledgements

Thank you to all the people who helped me get to this point. And specially to my parents, who provided me with the foundation I needed to start, to the friends and colleagues who were always supporting, and to my supervisor Rafael Frederico Alexandre, who went above and beyond the duties of any teacher. Without all of you I would not have made it halfway here.

"True wisdom comes to each of us when we realize how little we understand about life, ourselves, and the world around us" - Socrates

Abstract

The School Bus Routing Problem has been extensively studied in the recent years, due mainly to its real world applications. It is concerned with the pick-up and delivery of students to and from school subject to a set of constraints. Real-world applications of the School Bus Routing Problem need a considerable amount of data that changes with every year, such as which school each student is enrolled in. Such information could prove difficult to manage without the help of tools for the management of data. This work presents the development of one such WEB application to manage this data using React, AdminLTE, Java, and PostgreSQL. The final result is an application that allows for the management of students, schools, vehicles, and constraints related to the School Bus Routing Problem.

Key-words: School Bus Routing Problem. WEB development.

Resumo

O Problema de Roteamento de Ônibus Escolares tem sido extensivamente estudado nos anos recentes devido, principalmente, às suas aplicações no mundo real. O problema foca o transporte de estudantes para a escola e de volta, sujeito a um conjunto de restrições. Aplicações reais do Problema de Roteamento de Ônibus Escolares necessitam de uma quantidade considerável de informação que muda todos os anos, como, por exemplo, em qual escola cada estudante esta matriculado. Manter estes dados pode se mostrar impossível sem o auxilio de ferramentas para a gestão de dados. Este trabalho apresenta o desenvolvimento de uma aplicação WEB para gerir tal informação utilizando React, AdminLTE, Java e PostgreSQL. O resultado final é uma aplicação que permite a gestão de estudantes escolas, veículos, e restrições relacionadas ao Problema de Roteamento de Veículos Escolares.

Palavras chave: Problema de Roteamento de Ônibus Escolares. Desenvolvimento WEB.

List of figures

List of abbreviations and acronyms

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
CRUD	Create, Read, Update, and Delete
CSS	Cascading Style Sheets
DAO	Data Access Object
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HVRP	Heterogeneous Fleet Vehicle Routing Problem
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
JSON	JavaScript Object Notation
JSX	JavaScript XML
JVM	Java Virtual Machine
JWT	JSON Web Token
REST	Representational State Transfer
SBRP	School Bus Routing Problem
SOAP	Simple Object Access Protocol
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VGI	Volunteered Geographic Information
VRP	Vehicle Routing Problem
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language (sic)

Contents

1 Introduction

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem, and is concerned with finding optimal routes for allowing a fleet of vehicles to service a number of clients, subject to side constraints (GOLDEN; RAGHAVAN; WASIL, 2008). This is a NP-hard problem, which means that finding an optimal solution is computationally unviable leading most research on the topic to focus on heuristic approaches (PRINS, 2004).

Golden, Raghavan e Wasil (2008) describe a category of vehicle routing problems known as "rich" VRPs, which are closer to real-world problems. These VRPs are characterized by the presence of many constraints not present on the original problem, like multiple depots, multiple trips, a heterogenous fleet of vehicles, and time constraints. The school bus routing problem is one of those rich VRPs.

On the school bus routing problem, a fleet of vehicles has to visit a set of bus stops, collect students and then deliver them at their respective schools. The reverse process, picking the students at their schools and then delivering them at their bus stops is usually analogous with only a few differences (LI; FU, 2002). Lack of proper planning when utilizing school buses may lead not only to a waste of resources but also to great discomfort to the students, who may spend hours on the buses (KANG et al., 2015).

There are many variants of the school bus routing problem, which can be roughly categorized based on the characteristics of the problem. Such characteristics include the number of schools (one or many), the surroundings (urban, rural, or both), the fleet (all vehicles share the same characteristics or not), etc. The SBRP can also be categorized based on its constraints, like the schedule of the schools (earliest and latest time for the bus to arrive), the maximum time a student may spend on the vehicle, etc., and by its objectives, for instance reducing the number of vehicles needed or the distance traveled (PARK; KIM, 2010). For this work the following characteristics are considered: multiple schools, rural surroundings, heterogeneous fleet, trips during the morning, afternoon and night, students of different schools may be carried by the same bus, and special education students are considered. The constraints include the vehicle capacity, maximum time spent on the vehicle, maximum walking distance from the student's house to the bus stop, and the school schedule. Finally, the objective is the minimization of total costs, both by reducing the number of buses and by reducing the total travel distance.

In order to successfully generate viable solutions the school bus routing problem, a considerable amount of information is required, such as the geographical position of students and schools, the characteristics of the fleet of vehicles, the schedules of each school, as well as constraints placed on the problem, like the maximum distance between a student's house and their assigned bus stop. Such large amounts of data can quickly become unmanageable without the assistance of tools. This work sought to develop a web application that helps with the management of all the information necessary to solve the school bus routing problem in a real-world setting, focusing mainly on the students, the fleet, and the schools. So far as the author's knowledge, no similar application exists today on the brazilian market.

It is worth noting that this work was not concerned with finding solutions for the school bus routing problem, nor with helping decision making related to it, as such modules are being developed in parallel as part of a larger project, as shown in Figure 1. This work is concerned with the creation of the RESTful API[1] and the web page, meaning it focused on developing the part of the system that interacts directly eith the users. Other modules shown are the Bus Stop Selection, concerned with assigning students to bus stops in such a way as to minimize the total number of stops and the distance between them, the optimization algorithm, or School Bus Routing, that is concerned with generating routes for the colection of students on the bus stops and delivering them to school while minimizing the total distance traveled and the number of vehicles needed, and the tracking module, that allows for real time tracking of vehicles.

1.1 Objectives

This work aims to develop a tool for the management of information related to the school bus routing problem. In order to achieve that, the following steps were proposed. These steps were not sequential, and many were either revisited after completion or ongoing for as long as the project was not finished.

- Step 1: to study and analyze current technology and tools for the development of web applications;
- Step 2: to choose technologies and tools to be used during the development step of this work;
- Step 3: analysis of functional requirements for the project;
- Step 4: development of a web application using the chosen tools;
- Step 5: to evaluate and propose improvements for the application during and after development.

1 Application Programming Interface

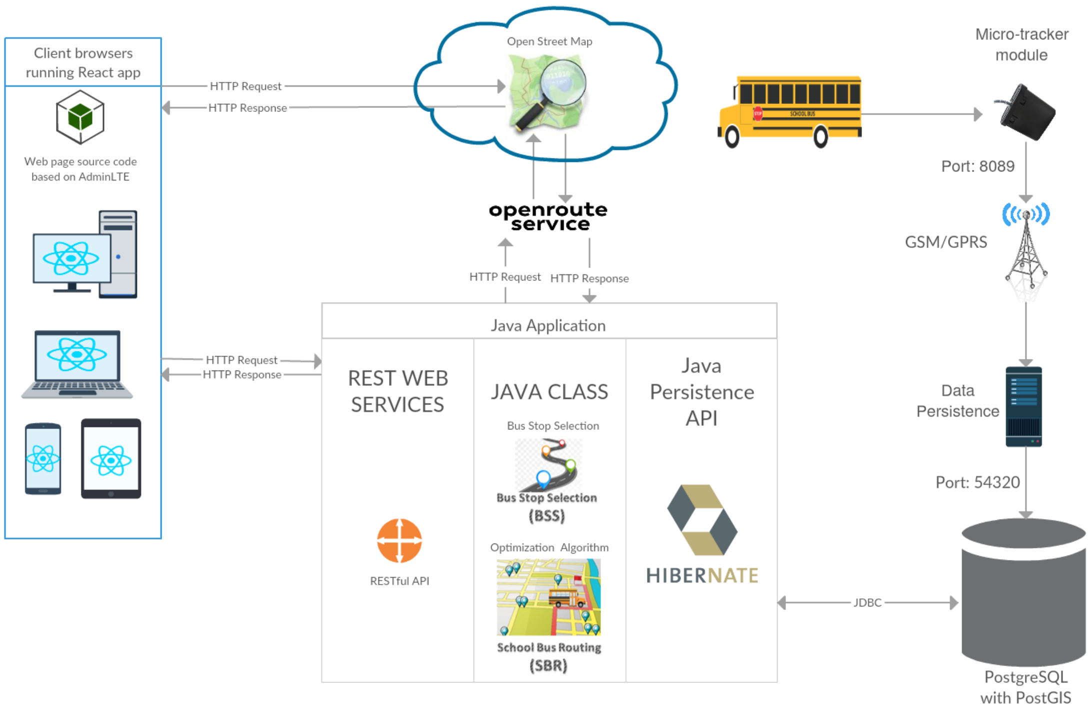


Figure 1 – Diagram depicting the modules of the project and their relationship to one another. This particular work is concerned with the creation of the RESTful API and the web page.

Source: Created by the author.

1.2 Structure of this work

This work is structured as follows: Chapter 2 discourses over the literature surrounding the school bus routing problem and the technologies currently in use for the development of web applications. Chapter 3 presents the development of the web app. And finally, Chapter 4 gives a summary of this entire paper as well as proposes some possible future works.

2 Literature review

This chapter will present a revision of the literature related to the project. The Vehicle Routing Problem will be presented in further detail in section 2.1, while the School Bus Routing Problem will be revisited in section 2.2 and a review of the technologies used in the project, as well as some that were considered but ultimately not used, in section 2.3.

2.1 The Vehicle Routing Problem

At its core, the Vehicle Routing Problem consists of finding optimal routes for a fleet of vehicles to visit a set of points, in such a way as to minimize cost (TOTH; VIGO, 2002). The problem was initially proposed in 1959 when Dantzig and Ramser released their paper entitled "The Truck Dispatching Problem", on which they describe and propose a solution for the problem of finding optimal routes for gasoline delivery trucks to service a set of gas stations (DANTZIG; RAMSER, 1959). The VRP is an NP-hard problem, and as such, finding the optimal solution for the problem is not viable, except for diminutive instances. Research on the VRP tends, therefore, to focus on heuristic and metaheuristic approaches (KUMAR; PANNEERSELVAM, 2012).

Many variants of the VRP have been considered in the literature, each defined by a different set of side constraints like fleet size and homogeneity, time constraints, and the number of depots (GOLDEN; RAGHAVAN; WASIL, 2008). One such variation of the VRP is the School Bus Routing Problem.

2.2 The School Bus Routing Problem

Initially presented in 1969 by Newton e Thomas (1969), the School Bus Routing Problem (SBRP) consists of a subset of the VRPs concerned with the transportation of students between a set of bus stops and their schools, and vice versa. While the school bus routing problem consists of one large problem, it can be divided into five smaller problems (PARK; KIM, 2010):

- Data preparation: the web of roads and relevant nodes (student homes, schools, etc.) are prepared and converted to an origin-destination matrix, which contains the lowest costs for travel between any two nodes;
- Bus stop selection: the bus stops are selected and students are assigned to them;

- Bus route generation: similar to the basic VRP, routes are created so that the fleet of vehicles can service every bus stop and school;
- School bell time adjustment: necessary to make sure all students get to school on time for their classes. While in most of the literature the school bell time adjustment is considered a restriction, some works try to find the optimal times that would allow for a smaller fleet to service all schools;
- Route scheduling, which is the scheduling of buses for multiple successive trips.

Ellegood et al. (2019) do not take the data preparation into consideration, instead focusing on strategic transportation policy, the study of how a policy may help or hinder the problem. For instance, Ellegood, Campbell e North (2015) discourse over how a mixed load policy may lead to shorter total distances under certain conditions.

As the considerable complexity of the main problem makes tackling it as one single entity considerably more difficult, the sub-problems of the SBRP are usually considered separately, allowing for a divide and conquer strategy. This is despite the fact that all of the parts of the problem are intrinsically related. Moreover, the literature on the subject tends to forego one or more of the sub-problems, focusing more on the routing and scheduling of the vehicles and assuming the remaining information will be provided by the interested parties (PARK; KIM, 2010).

According to Ellegood et al. (2019), there has been a large increase in the number of publications about the SBRP in recent years. According to the authors, there have been more papers released in the last decade than in the thirty years prior. Not only that, but recent research has been trending towards more emphasis on real-world issues and meta-heuristic solutions.

2.2.1 Characteristics of the problem

Park e Kim (2010) manage to isolate eight characteristics of the SBRP that impact heavily on possible solutions. Those are:

- **The number of schools**: the SBRP usually considers multiple schools, however, in the literature about the subject, there is a large number of papers focused on solving the single school variant. Should a single school be considered for the problem, then the solution for the bus route generation sub-problem can be achieved in a similar manner as with the Open VRP, presented in Li, Golden e Wasil (2007) and differing from the traditional VRP by the vehicle not returning to the depot after servicing the last customer on a route.

As for multiple school solutions Spada, Bierlaire e Liebling (2005) describe two possible ways to tackle the problem, either by splitting it into multiple single school variants and solving them individually or by allowing a single route to service multiple schools. The second variant is more flexible, however, whenever a student from a new school is assigned to a route, the school must be inserted into it at the best insertion point, a process which is both costly and complicated.

- **Urban versus rural surroundings**: Chen et al. (1990) describe the characteristics of rural environments as, amongst other things, lower population density and fewer alternative routes.

  On an urban environment, it is likely that there will be multiple students on any given residential area, allowing for the definition of bus stops they can walk to. On a rural environment, however, due to the low population density, there is a real possibility that there will be no points to which two or more students can reasonably be expected to walk. Furthermore, urban areas provide relatively safe walking paths, with sidewalks and lower speed limits, that cannot always be expected of rural surroundings (ELLEGOOD et al., 2019). As such, the bus stop selection is oftentimes simpler on rural areas, with the stops being set as the residence of the students; Park e Kim (2010) even goes as far as to claim the bus stop selection as "not necessary for rural surroundings".

  According to Bowerman, Hall e Calamai (1995), on an urban environment, due to the greater population density, oftentimes the bus will reach full capacity before the time students spend traveling becomes a problem. The paper even goes as far as to drop the maximum travel time constraint entirely. The opposite is true for rural areas, on which a bus may be unable to reach capacity without breaking time constraints (ELLEGOOD et al., 2019).

  Finally, Ripplinger (2005) discusses the relatively small size of the rural problem in relation to the urban version. The author claims that in certain cases, the instance of the problem may even be as small as to allow for a manual selection of an optimal solution.

- **Morning problem versus afternoon problem**: in some countries, students are taken to school during the morning and return home during the afternoon. In such countries, solutions for the SBRP can take into consideration the difference between the two time periods, however, since the afternoon problem is usually just a more lax version of the morning problem done backward, most of the literature focuses on the morning problem and simply assumes that any solution found for it can be applied to the afternoon problem as well (PARK; KIM, 2010).

There are, however, some like Bodin e Berman (1979) who research the afternoon problem. Amongst other things, the authors discuss the difference between reversing the route for the morning problem, a more intuitive approach, or replicating it, a "fair" approach, which would ensure a more even distribution of travel time between students.

- **Allowance of mixed loading**: as discussed by Braca et al. (1997), allowing students from different schools to ride on the same vehicle can lead to a more flexible and efficient solution, however Spada, Bierlaire e Liebling (2005) talks about how it is significantly more complicated to deal with the routing of vehicles if multiple schools are taken into consideration. Chen, Kallsen e Snider (1988) mentions how forbidding mixed loading is overly restrictive and can lead to a waste of resources.

  According to Ellegood et al. (2019) both allowing and forbidding mixed loading show advantages and disadvantages. Mixed solutions allowing for every bus stop to only be visited once at the cost of buses traveling under capacity after visiting a school, and possibly higher wait times. Meanwhile, non-mixed approaches allow for the buses to travel at full capacity for a relatively longer time, at the cost of the possibility of multiple buses needing to visit the same bus stop. As such different instances will get better results with different approaches. Ellegood, Campbell e North (2015), for instance, claim that mixed loading provides the best results in larger districts where students of different schools are more likely to be allocated to the same bus stop, and in districts where the schools are closer together.

  Miranda et al. (2018) extend the definition of mixed loading to allow for a bus to simultaneously deliver and collect students, regardless of turn or whether they are going or returning from school, a process the authors named multi-loading. The hope is that multi-loading will allow for more versatile solutions and a possible reduction of costs.

- **Routing special-education students**: special-education students differ from general-education students on a fundamental level. Firstly, while general-education students may be expected to walk to a bus stop, special-education students are usually picked up at their own homes, meaning that, for each special ed. student added to the problem, a new and impossible to relocate point is added to the set of bus stops. Secondly, special ed. students oftentimes have to travel long distances to reach specialized schools with the proper infrastructure. Meanwhile, most general ed. students will be enrolled at a nearby school. Finally, some special ed. students (such as those who need wheelchairs), will necessitate special vehicles, making these Site-Dependent VRPs (BRACA et al., 1997). Site-Dependent VRPs are those on which there are compatibility issues between clients and vehicles, meaning only specific vehicles can supply the clients demand (GOLDEN; RAGHAVAN; WASIL, 2008). Caceres, Batta e He (2019) also mention the fact that special ed. students have a need for constant supervision by someone other than the driver.

Interestingly, both the first and second problems outlined above are somewhat mitigated on rural environments, since both long travel distances and the collection of students at their homes are considered a given on that scenario.

- **Homogeneous or heterogeneous fleet**: the Heterogeneous Fleet Vehicle Routing Problem (HVRP) is a variation of the VRP on which the difference in capacity and cost of different vehicles is taken into consideration (GENDREAU et al., 1999). A similar restriction can be considered for the SBRP (PARK; KIM, 2010). Moreover, Newton e Thomas (1974) considers that the same vehicle can have different capacities for different schools, since some schools will be more restrictive with how crowded a vehicle can be during transportation.

  According to Miranda et al. (2018), most publications consider a homogeneous fleet, and many of the ones that do not, still fail to consider the costs of having different vehicles. Exceptions include Ripplinger (2005), Li e Fu (2002), and Miranda et al. (2018) themselves.

- **Objectives**: according to Savas (1978) public services can be evaluated using one of three measures: effectiveness, efficiency, and equity. Effectiveness being the capacity of a service to satisfy a demand, efficiency being the cost-effectiveness of said service, and equity being its fairness. Of those three, most of the literature on the SBRP focuses on efficiency and effectiveness, with only ever so few papers focusing on equity (PARK; KIM, 2010). The majority of publications on the subject tends to try and minimize costs by either reducing fixed costs, via reducing the total number of vehicles needed, or by reducing variable costs, via minimizing the total distance traveled (ELLEGOOD et al., 2019). Of course, there is no reason for only one objective to be pursued, Corberán et al. (2002) for instance seek to both reduce the number of buses and the total travel time of the students.

- **Constraints**: various different sets of constraints can be applied to the SBRP, many of which only make sense on a specific iteration of the problem. Braca et al. (1997) and Spada, Bierlaire e Liebling (2005) list the following as "general" constraints:

  - Vehicle capacity; how many students a vehicle can transport at once;
  - Maximum riding time: how long a student may spend traveling;
  - Maximum walking distance: maximum distance between a student's home and bus-stop;
  - School time window: the time window during which a vehicle may arrive at the school;

– Upper bounds on the number of students at stops;

– Earliest pickup time for children;

– The minimum number of children to create a route.

Besides the characteristics outlined above, that are present in most, if not all publications about the subject, there are also specific characteristics that are tackled only by a few papers. Some examples of such esoteric papers are Bögl, Doerner e Parragh (2015), that consider the transfer of students between multiple buses, a process that may lead to better routing but also to a perceived loss of quality of the service, Caceres, Batta e He (2017) that deal with overbooking of buses, claiming that the chance of a student riding the bus varies between 22% to 77%, allowing for buses to have a number of pupils allocated to them larger than their maximum capacity, and Miranda et al. (2018) that take into consideration the necessity of the presence of a monitor in some buses, such as when the students are too young. In such cases, one seat will be occupied by the monitor, and the fixed costs of the solution will be impacted by their pay.

2.3 Technologies for the creation of web Apps

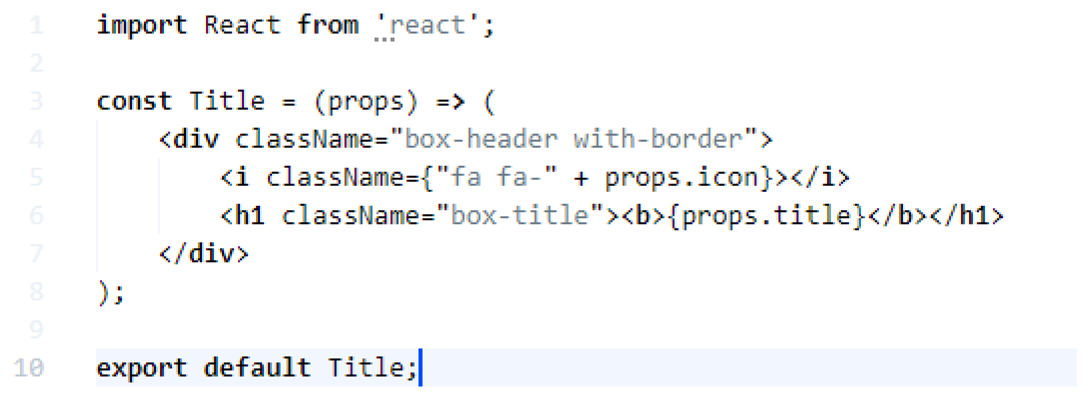
This section aims to present the technologies and tools used in this work, as well as the reasoning that led to these specific choices. For the sake of comparison, it will also briefly mention some technologies that were considered but not used, either because of technical issues, licensing costs, or simply user preference.

The first step in selecting the technologies to build the application was the user interface (UI), also known as the front-end. Special care has to be put on the user interface. As it is the part of the system that directly interacts with the client, the UI must not only look pleasing, but also contain all of the information necessary for the user to deduce which functions are available and how to access them, concepts known as *discoverability* and *understanding* (NORMAN, 2013). On that front, two technologies were necessary, React to control behavior, and AdminLTE to control appearance.

After the technologies for the front-end were defined it was time to look at the available tools for the back-end, the part of the software responsible for the control and processing of requests and data. The back-end can be further divided into two parts, the Application Programming Interface (API) and the database. The API receives requests from the front-end, validates them, and, either fetches or inserts information into the database. The database is responsible for the storage of data on a semi-permanent state, a term that here means information can never be erased but may be overwritten. Since the back-end does not interact directly with users, function was sought over form leading to the decision of displaying a simple report of oncoming requests on a command line interface instead of a full-blown UI for the API. Java was used for the construction of the API and PostgreSQL for the database.

2.3.1 React

In order to control the behavior of the front-end of the application React was utilized. React is a JavaScript library for the development of front-end applications. Its main selling point is the concept of components (Figure 2), which work similarly to JavaScript functions receiving a "prop" value and returning a piece of code describing what should appear onscreen (REACT, 2018a). Like functions, components are reusable, allowing for better manutenibility and less reworking. React components are written in a mix between plain JavaScript and JSX (JavaScript XML[1]). JSX is a syntax extension to JavaScript, that intends to allow the developer to use the power of JavaScript with a simpler syntax, resembling that of HTML (REACT, 2018b). While the usage of JSX is by no means mandatory, it greatly simplifies the process of component creation.



```
import React from 'react';

const Title = (props) => (
    <div className="box-header with-border">
        <i className={"fa fa-" + props.icon}></i>
        <h1 className="box-title"><b>{props.title}</b></h1>
    </div>
);

export default Title;
```

Figure 2 – Example of a React component that receives the name of an icon and a title as props and renders the icon followed by the title in bold letters.

Source: Created by the author.

React is also known for the use the virtual DOM (Document Object Model), a technique which replicates the DOM, operates upon the copy and then compares the virtual and real DOMs in order to only update the parts which are different. This technique updates what is shown in the browser in a very fast and efficient way, since there is no need to reload the entire page every time something changes (KALUŽA; TROSKOT; VUKELIĆ, 2018). This, in turn, allows for the creation of single-page applications, web applications that do not reload the entire page whenever the need for a different functionality arrises. Single-page applications allow for a more fluid, interactive, and overall more user-friendly experience when compared to classical multi-page applications (MESBAH; DEURSEN, 2007).

[1] Extensible Markup Language

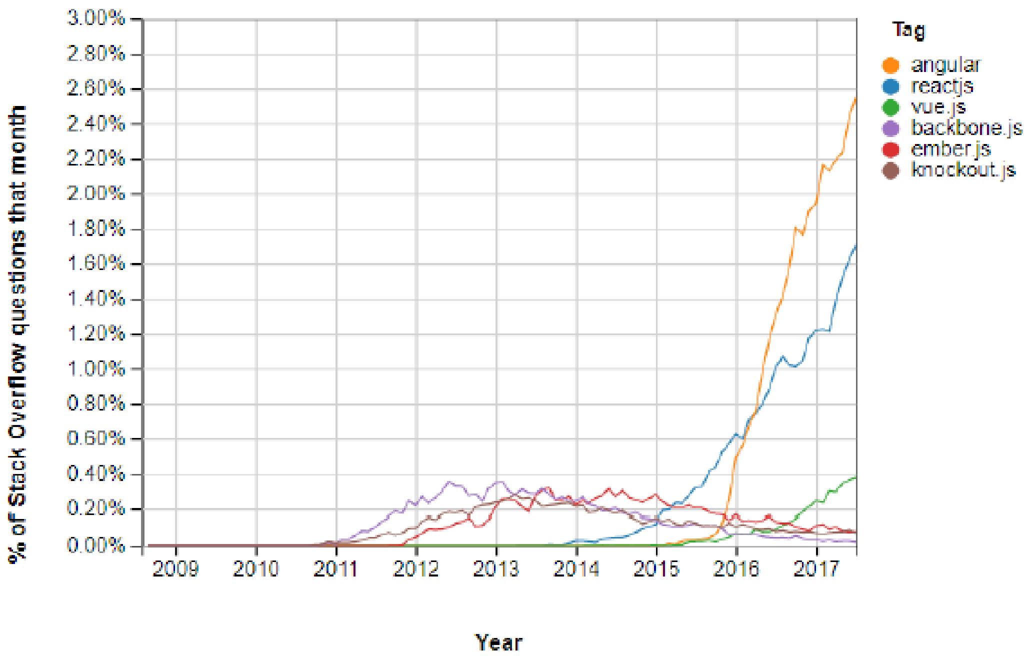


Figure 3 – The popularity scale of the 6 most well-known Javascript frameworks. React is shown in blue.

Source: Kaluža, Troskot e Vukelić (2018).

React was released in 2013 by Facebook (FEDOSEJEV, 2015), and it is still mantained and updated regularly, with a major update, the introduction of Hooks, being released mere months before this work was written.

React was chosen for this project due to the attractiveness of its component based structure, as well as its active community; it is amongst the most popular JavaScript libraries currently, with over 131.000 stars on GitHub at the time of writing. Figure 3 shows the popularity of six different front-end JavaScript frameworks for the development of web applications. It is worth noting that both of the described reasons for choosing to use react are shared by Vue and in the end the choice between the two tools was mostly due to preference. Angular was a close third choice, however, due to a steeper learning curve it was not quite as desirable as the other two.

Since this work aimed to create a single-page-application, the page is never reloaded, however updates are still triggered via changes in the URL[2]. This is possible via the use of a Router, that renders different components depending on the current URL (React Training, 2019). Multiple routers are used throughout the application, dealing with different parts of the URL. For example, if the current URL is http://hostname/users/new, the main Router will read the /users part of the address and render the Users Router that will, in turn, read the /new part of the address and render the component used to create new

[2] Uniform Resource Locator

users.

2.3.2 AdminLTE

The choice between creating a new template or modifying an existing one to suit the project's needs was the main driving force behind the choice of AdminLTE. Although it would be simple enough to build a new template from scratch via the use of CSS (Cascading Style Sheet), it would be a time-consuming task, and the end result would most likely be inferior to the templates freely available on the web, so one such template, AdminLTE, was used.

AdminLTE is an open source web template, developed by Abdullah Almsaeed and based on Bootstrap 3 (ALMSAEED, 2014), a framework for the styling of web pages, which is in turn based upon CSS. AdminLTE was chosen not only for its beautiful, responsive design, which means that applications still looks attractive on different and even mobile devices (OLIVEIRA-CIABATI et al., 2017), but also as a natural consequence of choosing to use React, since its modular design goes well together with React's component based style of presenting user interfaces.

Due to the nature of the project as a data management tool focused on rural areas, there was a concern that some of the users may have little experience with computers and especially with web tools, and as such, an intuitive design that draws the eye to the important parts of the system was considered essential. As shown in Figure 4 AdminLTE presents a large lateral menu in a dark color which contrasts with the white background of the main page, as well as several smaller components in bright colors, that could be used to draw attention to important info, guiding the user without the need of heavy tutorials or training.

2.3.3 Java

Java is an object-oriented programing language. It is rather well-known and has a huge community surrounding it. It is known mainly for its Java Virtual Machine (JVM) which was intended to allow developers to "write once, run anywhere". What that means is that, in theory at least, developers are able to make code that could run in any platform as long as it has a JVM installed. In practice, differences between architectures and the JVMs themselves mean some degree of adaptation is still required. Java also automated the task of memory management via the use of the Garbage Collector (ALOMARI et al., 2015).

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan in 1991. Initially called Oak, the original impetus behind Java was to create a platform-independent language that could be used not only by computers but

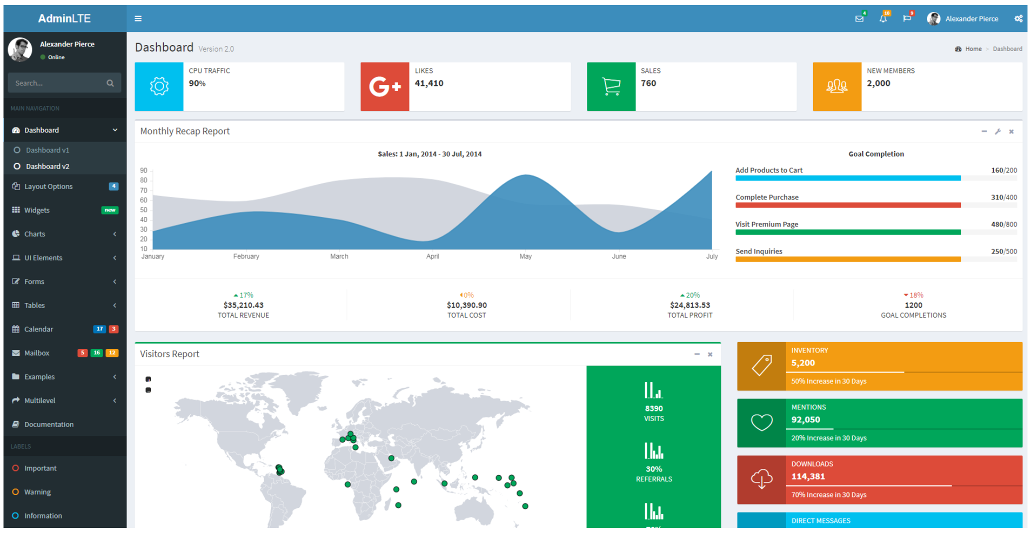


Figure 4 – Demonstration page for AdminLTE Dashboard V2.

Source: http://adminlte.io/.

also by consumer electronic devices. During Java's development, it became clear that the portable programs java sought to enable would prove even more useful on the World Wide Web, and as such the focus changed to the creation of a programming language for the internet (SCHILDT, 2003).

Nanz e Furia (2015) compare eight programming languages, ranking them according to average lines of code on a solution, size of generated executables, running time performance, memory usage, and runtime failure proneness, and while Java does not top any of the lists, it is also rarely amongst the worse. This jack of all trades, master of none approach seem to be popular with the community; Java has consistently been near the top in rankings of programming languages for years (O'GRADY, 2017; O'GRADY, 2018; O'GRADY, 2019). Figure 5 shows a diagram created by Stephen O'Grady about the popularity of Java in comparisson to other programing languages on Stack Overflow and GitHub at the start of 2019.

Overall Java is powerful, simple to use, and has an active community, which, added to the familiarity the authors have with it, made it the natural choice as the API language. It is also worth noticing that other modules of the project, like the module responsible for the estimation of costs, are also being developed in Java, which impacted on the decision.

2.3.4 REST

While it is an architecture and not a technology in and of itself, the Representational State Transfer (REST) architecture is interesting enough to warrant it's own subsection.

Before explaining REST, first we need to explain web services. Halili e Ramadani (2018) define web services as a way to expose business logic as a service over the Internet.

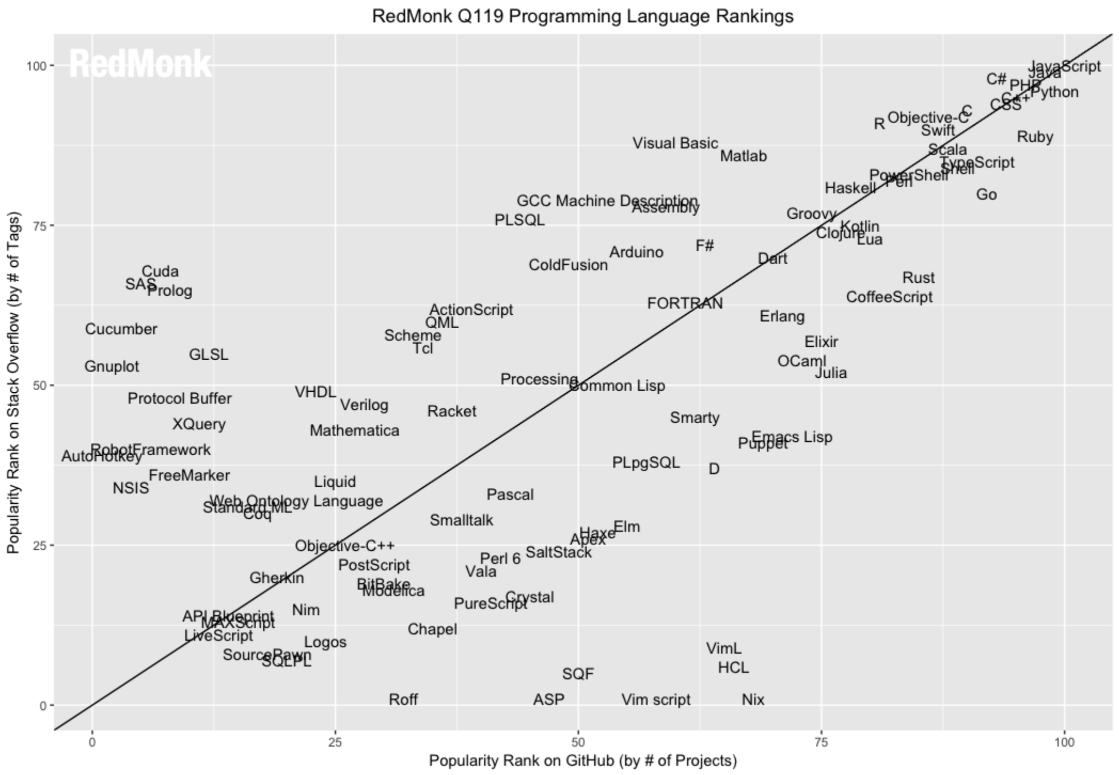


Figure 5 – Diagram showing the popularity of programing languages at the start of 2019 on GitHub and Stack Overflow. Java can be found on the top right, bellow only JavaScript.

Source: O'Grady (2019).

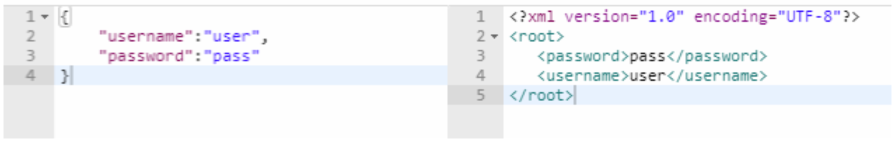
Most web services are created following on of two architectures: REST or SOAP (Simple Object Access Protocol).

When deciding between SOAP and REST for this work, the defining factor was simplicity. While SOAP is more secure and less error prone, as well as having plenty of other advantages over REST (MUEHLEN; NICKERSON; SWENSON, 2005; TIHOMIROVS; GRABIS, 2016), it is significantly more complicated to learn and implement, has worse performance, is harder to scale, and is restricted to using XML in its requests while REST allows for the use of JSON[3] and YAML[4], both much more human-friendly alternatives (HALILI; RAMADANI, 2018). A comparison between the JSON and XML formats can be seen in figure 6.

A web service is called RESTful if it follows the REST architecture (HALILI; RAMADANI, 2018). According to Rodriguez (2008), RESTful services follow four basic principles:

- the explicit use of HTTP[5] methods, which are used as a way to communicate the

3 JavaScript Object Notation
4 YAML Ain't Markup Language (sic)
5 Hypertext Transfer Protocol



```json
{
    "username":"user",
    "password":"pass"
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
   <password>pass</password>
   <username>user</username>
</root>
```

Figure 6 – Representations of the same resource in JSON (left) and in XML (right).

Source: Created by the author.

intent of the request (GET to recover data, POST to save data, PUT to update, and DELETE to delete);

- the lack of server-side state, meaning that all data required to fulfill the request is present within its headers and body;
- intuitive URIs, so that the address necessary to perform an action can be easily inferred;
- simple and readable resource representations, for this project the JSON format was chosen, but XML and YAML are also viable options.

2.3.5 PostgreSQL

Once the front and back-end technologies were decided, all that was left was to choose the tool for data storage. There are more than a few database management systems available for free, each with their own advantages, however, in the end, PostgreSQL was chosen.

With over 30 years of development, PostgreSQL is an open source object-relational database system. It runs on all major operating systems and has a reputation for reliability, robust features, and extensibility (POSTGRESQL, 2018). Perhaps more important to this project, it has a well-documented set of geometric data types, which can be used to represent geographical coordinates (POSTGRESQL, 1999).

2.3.6 JSON Web Tokens

In order to protect sensitive information, it was necessary to limit user access based on their system permissions. The solution was to use JSON Web Tokens (JWT), encrypted with the SHA-256, to validate requests. The workings of the SHA-256 algorithm are complicated and not really relevant to this work, suffice to explain that the SHA-2 family of algorithms are resistant to many different types of attacks, providing better security than it's predecessors (GILBERT; HANDSCHUH, 2003).

JWTs themselves are defined in RFC-7519 and are formed by a header that includes information about the token and the encryption algorithm, the payload containing information about the entity (normally the user), and the signature which is the concatenation of the hashes created using the header and payload, used to validate the integrity of the token and whether it was modified (JONES; BRADLEY; SAKIMURA, 2015).

As shown in figure 7, the application requests a token by sending the username and password of the user to the server. Afterward, any requests to the API must include the valid token in their header, otherwise, they will be ignored. JWTs also include an expiration date, after which they become unusable. The JWT can, therefore, be compared to a temporary key needed to access the API.

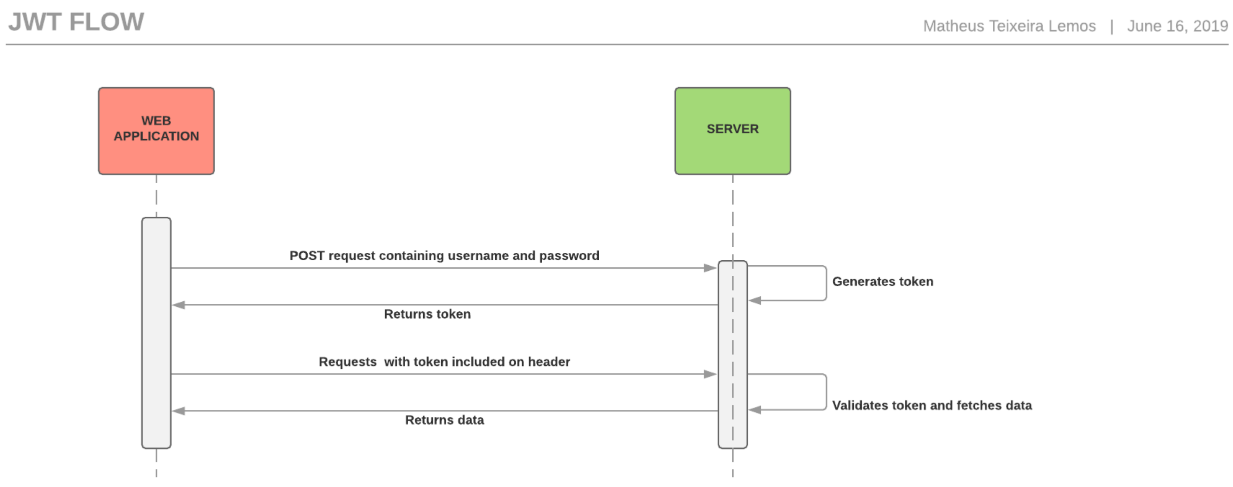

JWT FLOW
Matheus Teixeira Lemos | June 16, 2019
WEB APPLICATION
SERVER
POST request containing username and password
Generates token
Returns token
Requests with token included on header
Validates token and fetches data
Returns data

Figure 7 – Diagram of the partial lifecycle of a JWT, not including expiration.

Source: Created by the author using Lucidchart.

2.3.7 Open Street Map and Nominatim

While the technologies described above cover the entire structure of the project, one problem arose from the school bus routing problem that could not be solved by any of them. That was the need to have geographical coordinates for the relevant points, such as schools and student's houses. Most people do not have this kind of information on hand, so leaving it to the user was also not possible, therefore one more tool was procured, a database with a search tool that would allow users to input an address and then convert said address into coordinates to be shown on a map onscreen. The user could then manually adjust the location. Google Maps was originally considered but in the end, OpenStreetMap and its search tool, Nominatim, were chosen.

OpenStreetMap is an open database of maps. It is free to use and works via the use of Volunteered Geographic Information (VGI), also described as "crowdsourcing geospatial data", which means that the users themselves gather and upload data to the central database. OpenStreetMap surged in popularity in 2012, when changes to the licensing of Google Maps increased potential costs (NEIS; ZIPF, 2012).

Nominatim is a search engine for OpenStreetMap's data which allows both inputting an address in order to find out its geographical coordinates and inputting coordinates in order to discover an address. OpenStreetMap and Nominatim were chosen mostly for their free to use nature, though their quality compared to both paid and free options is worthy of note.

3 Development

This section will present the development of the web tool, from its initial proposal to the ultimate result. The development of the application followed a framework derived from Scrum and adapted for the university environment. The first and largest change was in relation to the roles of the team. Scrum defines three roles members of the team can assume: the Product Owner, responsible for maintaining the backlog of the product, the Scrum Master, responsible for making sure the team follows Scrum directives, and the Development team. Due to the nature of the environment, only two people were available for these roles with the Supervisor assuming the role of Product Owner and Scrum Master, while the author assumed the role of the development team. The second change was related to the meetings. The SCRUM guide calls or daily meetings, however, due to time restrictions and incompatible schedules, those were replaced by weekly meetings instead (SCHWABER; SUTHERLAND, 2011).

The functional requirements of the application were managed by the Product Owner, who would pass them on to the developer. One of the principles outlined in the Agile Manifesto is "Working software over comprehensive documentation" (BECK et al., 2001), so no effort was put into maintaining a comprehensive list of the implemented requirements. In fact until late into the development when Trello, a tool for the management of projects was introduced, no list was kept at all. Finally, the product owner was also in charge of testing and validating the work of the developer. No part of the project was considered done until such validation was made.

Before starting to define a proposal for the system, it is necessary to define the restrictions placed on the problem. Section 2.2 discourses about the set of characteristics that most influence the solution for the SBRP, and in this project the following were considered: multiple schools, morning problem, mixed loads are allowed, special-education students are considered, the fleets are heterogeneous, the objective is to minimize total cost, and the focus is on rural surroundings. That last one is of particular importance as it was the central factor in many of the design decisions related to the solution.

The first proposal for the system, created by analysing the functional requirements raised, was fairly simple. A tool that would allow the users to input data related to the school bus routing problem and then, after a certain amount of time passed, present the results. The application's intended users are public workers, be they from the schools or the city were the solution is being implemented. This initial proposal was then refined with time, as with each new functionality developed, new problems and opportunities presented themselves.

With the initial idea in mind, the next step was to choose the technologies and tools that would be used in the project. After gathering information on the available tools, the decision was made to use PostgreSQL for the database, Java for the back-end of the application, and Angular for the front-end. The latter was quickly substituted by React when it became clear the learning curve for it was far more steep. When designing the looks of the webpage became an issue, AdminLTE was included in this list. All of these tools were described in more detail in section 2.3.

The data related to the SBRP was divided into five main tables on the database: vehicles, schools, students, parameters used for the calculation of costs, and simulations, that contains all of the requests for solutions. There were also two more tables related less to the problem and more to the users: cities and the users themselves. The idea was to have the CRUD (create, read, update, and delete) operations for all of the main tables available for all of the users, while the users table was to be restricted to access by administrators, and the cities table was static and therefore would not be subject to any of the CRUD operations. The website itself would be organized in much the same way, with sections for each of the main data tables, as well as a section for the users table that would only appear if the user logged into the system had administrator privileges.

Three levels of user access were defined, pertaining to restrictions on the data they could access. First came the administrators, who had access to all of the information available on the database including all of the data relating to other users. Of special notes is the fact that administrators were the only ones with permission to perform CRUD operations on the users table, therefore being responsible for allowing and revoking access to the system. Then came state-level users, who could access information on every student, school and vehicle on their state, and city level users who were limited to data on their own municipality. While both administrators and state-level users have permission to access information pertaining to multiple cities, there was a concern that the amount of available data could prove overwhelming. The state of Minas Gerais alone has 853 cities, and according to a survey by INEP, had over three million students enrolled in 2018 (INEP, 2019). Assuming even 1% of said students were registered on the system, there would be an overwhelming 30.000 registers on the students table for that year alone. As such, a deliberate choice was made to not allow information on multiple cities to be requisitioned at once, with the user needing to change context whenever data from a different location was needed. The problem would most likely still persist for large cities, and especially capitals, however, we must remember the focus of the application is on rural environments, and it is therefore optimized for performing in such areas.

Another fairly troublesome difficulty encountered during the development was the need to have the geographic position of students and schools. While most users would be able to provide the address of such places, this information by itself is of no use to the solution of the SBRP. What was needed were the coordinates of points, that could be used to calculate distances, however, it was unlikely that the average user would have such information on hand. therefore the need for a "translator" was felt. Nominatim (described in subsection 2.3.7) could fulfill this role to a certain extent, however, while it was somewhat accurate on large cities, it was significantly less effective on more rural regions, which just so happen to be the main target of this project. Thus it was decided to use a mix of both strategies, using Nominatim to transform an address into a set of approximate coordinates, then presenting said coordinates to the user on a map. The user could then click on the map to fine-tune the position to an acceptable degree of error.

After the basic CRUD[1] operations for the input data were complete, it came time to focus on allowing the user to request solutions. For this step, two main concerns were identified: the user must be able to request customized solutions and the cost of said solutions had to be calculated based on a set of variable parameters. As such, two more sections were to be added to the system: simulations and parameters. The simulations section would allow for the creation of requests for solutions based on a set of parameters related to the students and the fleet, while the parameters section would allow for the input of data related to the variable costs of the problem, such as gas prices and taxes.

When it came to the development of the application itself, it was structured as such: first, the tables were created in the database, then the back-end, and finally, the front-end was built. Keeping in mind that a significant portion of the system was added after the initial version was up and running, those later additions followed the same structure.

The creation and structure of the back-end will be described in Subsection 3.1, and the front-end in Subsection 3.2. Seeing as the creation of the database is rather simple, if extremely important, it will not be given a subsection. Suffice to say, the necessary data was modeled following the same structure described above, resulting in the Entity Relationship Diagram shown in Figure 8.

3.1 Back-end

The back-end was built with Java following the Representational State Transfer (REST) architecture. It is structured as follows, the service package exposes a set of HTTP routes which listen for user requests, the jaxbean package provides resource representations for the user requests, the model package provides resource representations for the server responses, and the dao package provides an interface for the database. Each package contains one class for each of the following: simulations, cities, users, students, schools, vehicles, and parameters. There are also classes on the service and dao packages to handle login attempts. This structure is shown in figure 9.

[1] Create, Read, Update, and Delete

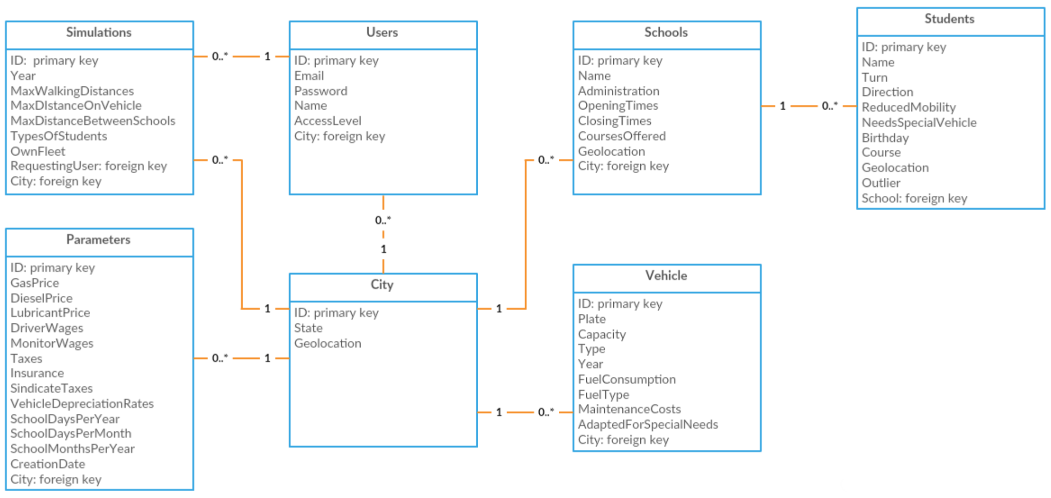


Figure 8 – Entity Relationship Diagram of the database.

Source: Created by the author using Creately.

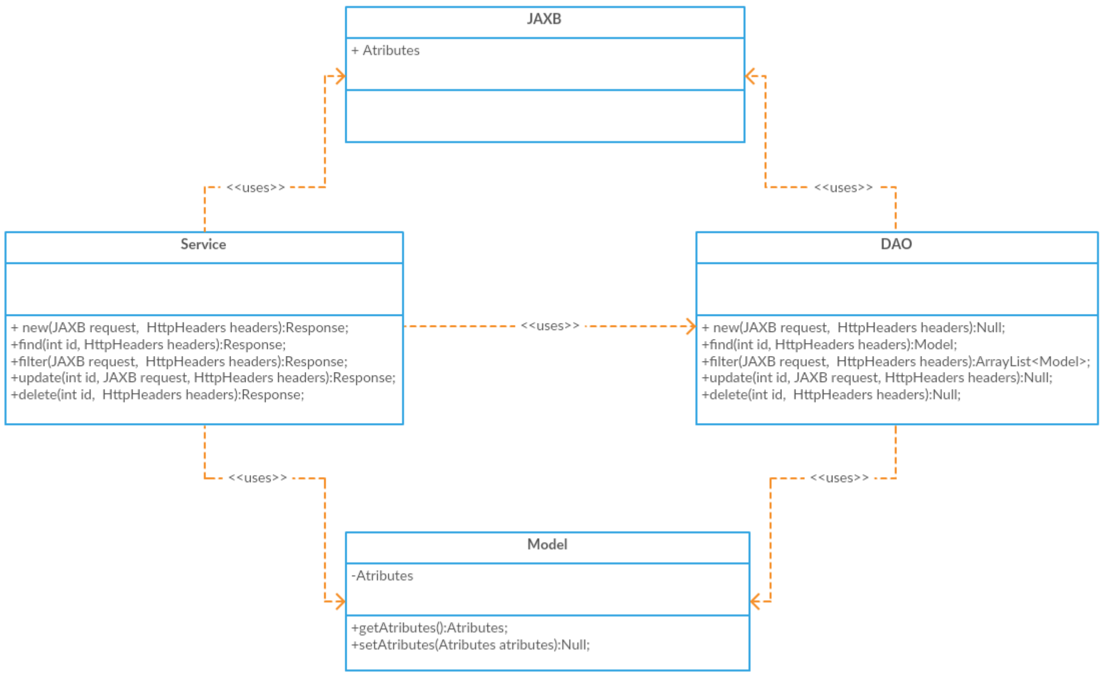

JAXB
+ Atributes
<<uses>>
Service
+ new(JAXB request, HttpHeaders headers):Response;
+find(int id, HttpHeaders headers):Response;
+filter(JAXB request, HttpHeaders headers):Response;
+update(int id, JAXB request, HttpHeaders headers):Response;
+delete(int id, HttpHeaders headers):Response;
DAO
+ new(JAXB request, HttpHeaders headers):Null;
+find(int id, HttpHeaders headers):Model;
+filter(JAXB request, HttpHeaders headers):ArrayList<Model>;
+update(int id, JAXB request, HttpHeaders headers):Null;
+delete(int id, HttpHeaders headers):Null;
Model
-Atributes
+getAtributes():Atributes;
+setAtributes(Atributes atributes):Null;

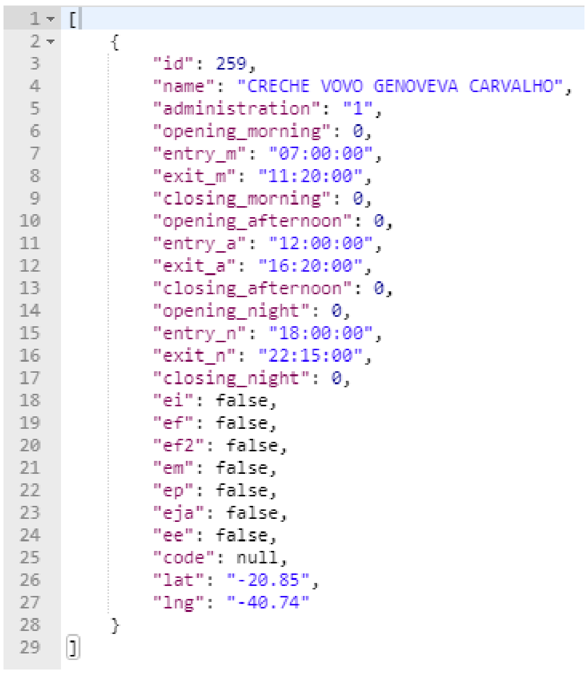
Figure 9 – UML class diagram describing the structure of the back-end. One instance of each of the shown classes was created for simulations, cities, users, students, schools, vehicles, and parameters.

Source: Created by the author using Creately.

Classes on the services package provide an interface for the front-end to send user requests to via HTTP. This interface is presented in the form of routes, which are exposed URIs that when called will validate the request and then call the appropriate method. HTTP methods are used to identify the type of operation to be performed, so a GET operation, for example, will return data from the server, while a PUT operation will insert new data. The URIs follow a specific pattern: the name of the resource, followed by the operation to be performed, followed by optional parameters, so as an example, the route for the exclusion of a user would be called by sending a DELETE HTTP request to the URL http://hostname/users/delete/{id}, with the section between curly braces being the parameter that lets the service know which user to exclude. It is important to note that a traditional REST implementation would not contain the operation on the URI, instead letting it be inferred from the HTTP method used for the call, however, in an effort to make the service more intuitive to people not accustomed to REST, a deliberate choice was made to include them.

Should a valid request be made to a route exposed by a class on the services package, the appropriate jaxbean will be prepared. Jaxbean classes are simple representations of the resources in XML terms, that map the fields on the request's body to a Java object that can be easily interpreted by the service.

Jaxbean objects are then passed to the appropriate method on the dao package. Data Access Objects (DAO) classes provide the appropriate interfaces for performing CRUD opperations on the database. Based on the type of request, the DAO object will either insert the received data or, if it was a read operation, create an object containing the requested information and return it to the service class, that sends it back to the user as a JSON (figure 10), completing the cycle.



```json
[
    {
        "id": 259,
        "name": "CRECHE VOVO GENOVEVA CARVALHO",
        "administration": "1",
        "opening_morning": 0,
        "entry_m": "07:00:00",
        "exit_m": "11:20:00",
        "closing_morning": 0,
        "opening_afternoon": 0,
        "entry_a": "12:00:00",
        "exit_a": "16:20:00",
        "closing_afternoon": 0,
        "opening_night": 0,
        "entry_n": "18:00:00",
        "exit_n": "22:15:00",
        "closing_night": 0,
        "ei": false,
        "ef": false,
        "ef2": false,
        "em": false,
        "ep": false,
        "eja": false,
        "ee": false,
        "code": null,
        "lat": "-20.85",
        "lng": "-40.74"
    }
]
```

Figure 10 – JSON returned after a request to the API.

Source: Created by the author.

There are also classes responsible for generating and validating JWTs, however due to the sensitive and complicated nature of the subject, all those classes do is delegate the job to a trustworthy library, Prime JWT, which can be found in https://github.com/bigdata06/prime-jwt.

3.2 Front-end

The development of the front-end of the system was composed of two stages. First, the AdminLTE template had to be adapted to work with React. This was made considerably easier due to the fact that AdminLTE is already somewhat modular, with its source code being divided into well-defined "parts". Some of these parts became React components, while others were deemed unnecessary and removed completely. The sidebar menu, the header, and the footer of the dashboard were kept, with minimal alterations made to adapt them to the system's needs. Other components, like form fields, some icons, and buttons were simple enough that no changes were deemed necessary.

After the conversion of the template, came the development of the components related to the CRUD operations. Each of these components renders a variety of other smaller ones, that may, in turn, render even more. The front-end followed a similar structure as the back-end, with a section dedicated for each of the following: simulations, users, students, schools, vehicles, and parameters. Each of these section is composed of a Router, a component that allows searching for resources, a component that allows for the creation of resources, and a component that allows the user to update a resource. This structure is shown in figure 11.

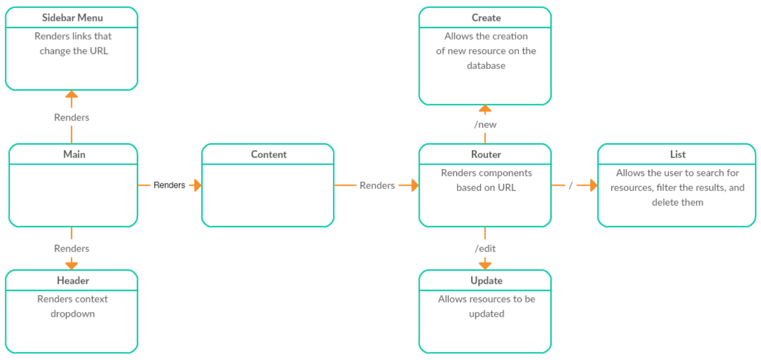

Sidebar Menu
Renders links that change the URL
Renders
Main
Renders
Header
Renders context dropdown
Renders
Content
Renders
Create
Allows the creation of new resource on the database
/new
Router
Renders components based on URL
/edit
Update
Allows resources to be updated
/
List
Allows the user to search for resources, filter the results, and delete them

Figure 11 – Generic structure of the components of the front-end. A more detailed view is presented in figure 17.

Source: Created by the author using Creately.

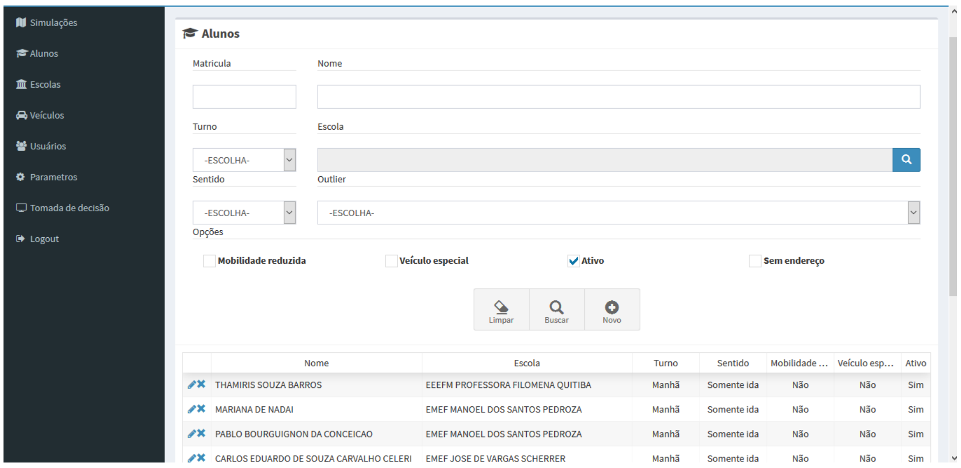
3.2.1 Main structure

When a user connects to the application they are presented with a simple login form. This form requests a JWT from the back-end that is stored on the session storage of the browser and is used on all of the following requests. The login form also presents a

"Forgot my password" option. Should this option be checked, the form will change to one requesting the email of the user. if a valid email address is inserted in this section, the back-end sends a link via email to the provided address that can be used to change the password. All of the requests from the front-end are performed via AJAX (Asynchronous Javascript and XML).

Should the user logged into the system have sufficient clearance, a button on the header opens a dropdown menu that can be used to change the municipality, allowing the user to view info about different contexts.

Links that lead to the sections described above are shown in a lateral menu that can be collapsed to provide extra space. When a user clicks on any of said links, the "List" component (figure 12) is loaded.



Matricula	
Nome	
Turno	-ESCOLHA-
Escola	
Sentido	-ESCOLHA-
Outlier	-ESCOLHA-
Opções	☐ Mobilidade reduzida ☐ Veículo especial [✓ Ativo] ☐ Sem endereço

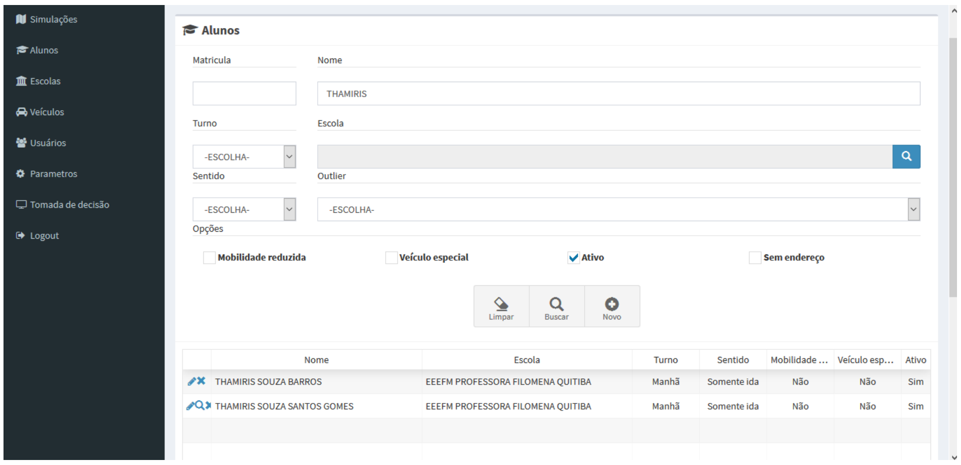
	Nome	Escola	Turno	Sentido	Mobilidade ...	Veículo esp...	Ativo
✎ ✕	THAMIRIS SOUZA BARROS	EEEFM PROFESSORA FILOMENA QUITIBA	Manhã	Somente ida	Não	Não	Sim
✎ ✕	MARIANA DE NADAI	EMEF MANOEL DOS SANTOS PEDROZA	Manhã	Somente ida	Não	Não	Sim
✎ ✕	PABLO BOURGUIGNON DA CONCEICAO	EMEF MANOEL DOS SANTOS PEDROZA	Manhã	Somente ida	Não	Não	Sim
✎ ✕	CARLOS EDUARDO DE SOUZA CARVALHO CELERI	EMEF JOSE DE VARGAS SCHERRER	Manhã	Somente ida	Não	Não	Sim

Figure 12 – List of students, displaying the students on the town of Piúma. The form component may be used to narrow down search results.

The List component is comprised of a table on which data referring to the SBRP is presented and a form with fields that can be used to narrow the search (figure 13).

Buttons on the list lead to the "Update" component, while a larger button on the form loads the "New" component (figure 14). Both components look and behave similarly, presenting a form that can be filled with information about the entity being inserted. The main difference between the two is that the "Update" version will come pre-filled with the data of the entity being updated. Once the form is filled the "save" button will trigger a local validation of the data, and, provided all of the information is valid, a request will be made to the back-end and the appropriate response message will be presented to the user based on whether or not the requisition was successful.

Whenever geographical data of a location is necessary, a button will be provided that, when clicked, will open a map with a search form (figure 15). Addresses may be searched on Nominatim and the results are presented on a side menu. When a result is clicked, coordinates that approximate that address will be presented on the map, and the



Alunos

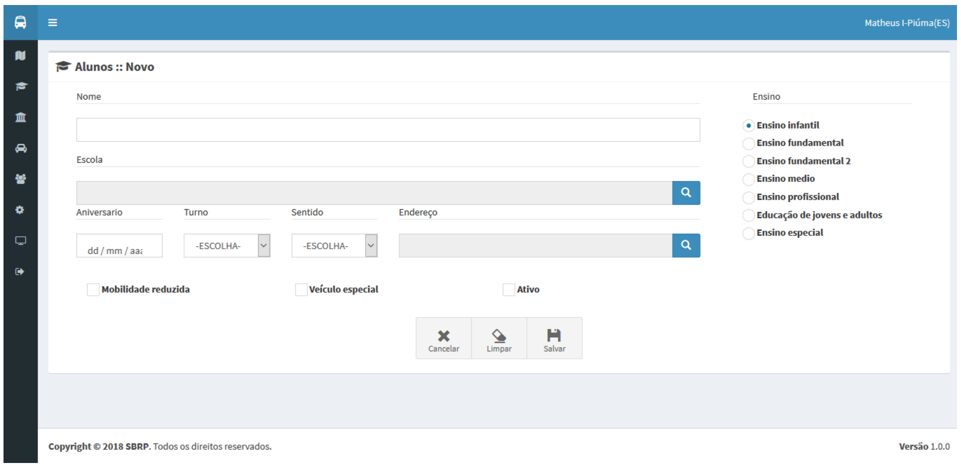
Matricula	
Nome	THAMIRIS
Turno	-ESCOLHA-
Escola	
Sentido	-ESCOLHA-
Outlier	-ESCOLHA-

Opções: Mobilidade reduzida [ ] Veículo especial [ ] Ativo [✓] Sem endereço [ ]

	Nome	Escola	Turno	Sentido	Mobilidade ...	Veículo esp...	Ativo
	THAMIRIS SOUZA BARROS	EEEFM PROFESSORA FILOMENA QUITIBA	Manhã	Somente ida	Não	Não	Sim
	THAMIRIS SOUZA SANTOS GOMES	EEEFM PROFESSORA FILOMENA QUITIBA	Manhã	Somente ida	Não	Não	Sim

Figure 13 – List of students on the town of Piúma with results filtered by name.

Source: Created by the author.



Alunos :: Novo

Nome	
Escola	
Aniversario	dd / mm / aa
Turno	-ESCOLHA-
Sentido	-ESCOLHA-
Endereço	

Ensino: [✓ Ensino infantil] Ensino fundamental Ensino fundamental 2 Ensino medio Ensino profissional Educação de jovens e adultos Ensino especial

Mobilidade reduzida [ ] Veículo especial [ ] Ativo [ ]

Copyright © 2018 SBRP. Todos os direitos reservados. Versão 1.0.0

Figure 14 – Registration form for students.

user may then adjust the location by clicking on the map.

Finally, a logout option on the side menu will delete the JWT from the session storage and return the user to the login screen.

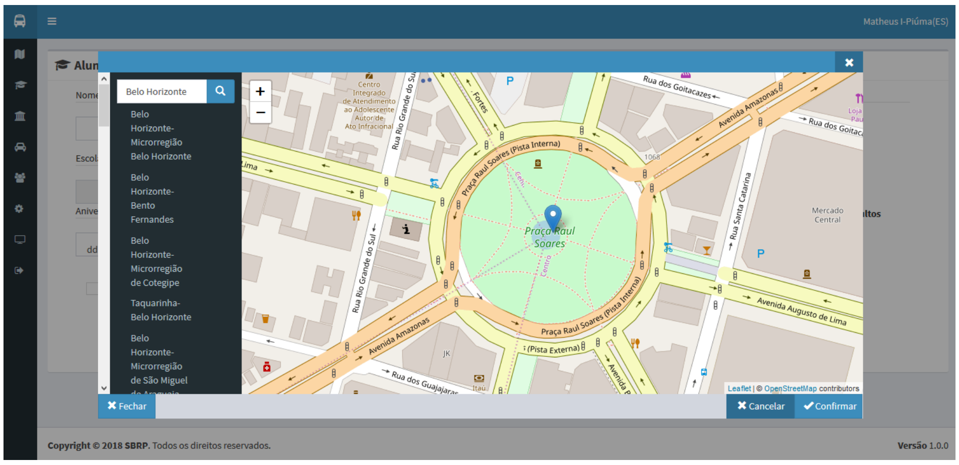

Matheus I-Piúma(ES)
Belo Horizonte
Belo Horizonte-Microrregião Belo Horizonte
Belo Horizonte-Bento Fernandes
Belo Horizonte-Microrregião de Cotegipe
Taquarinha-Belo Horizonte
Belo Horizonte-Microrregião de São Miguel
Praça Raul Soares
Leaflet | © OpenStreetMap contributors
Fechar
Cancelar
Confirmar
Copyright © 2018 SBRP. Todos os direitos reservados.
Versão 1.0.0

Figure 15 – Component for the selection of addresses. Users can search for an address on the sidebar or click on the map to set a location.

3.2.2 Simulations

The simulation section is one of the simplest ones. It allows the requests to be filtered by ID, stage of completion, the user responsible, and types of students considered.

When requesting a simulation, the user is requested to provide a set of parameters. The requested parameters are:

- The maximum walking distance allowed for each class of students;
- The maximum distance a student may travel on the vehicle;
- The maximum amount of time the student is allowed to stay in the vehicle;
- The maximum distance allowed between schools serviced by the same bus;
- The types of students to be included;
- The average speed the vehicles are expected to run at;
- The maximum allowed age for the vehicles;
- The maximum capacity of vehicles.

The age and maximum capacity of vehicles are needed in case the provided fleet is too small to cover the totality of the demand and new vehicles have to be requisitioned. These are parameters that may impact the solution and that usually vary due to the decisions of local authorities. They are inserted at the time of the request instead of in the parameters section to facilitate the creation of multiple simulations with tweaked parameters that can be compared to find the best solution.

3.2.3 Students

The students may be filtered by ID, name, school, turn (morning, afternoon, integral education, or night), direction (whether they take the bus to go to school, to return from school, or both), whether they have a need for special vehicles, whether they have reduced mobility, and whether they are outliers. A student is considered an outlier if they either live so far from the school that the restrictions for maximum distance or time traveling are broken regardless of which route they are assigned to, or in the opposite case, where students live so close to the school that it is more reasonable to walk than to take the bus. Besides showing the expected textual data, the table used to present the results of a search also presents a looking glass shaped icon that, when pressed, will open a map showing the location of the student's home.

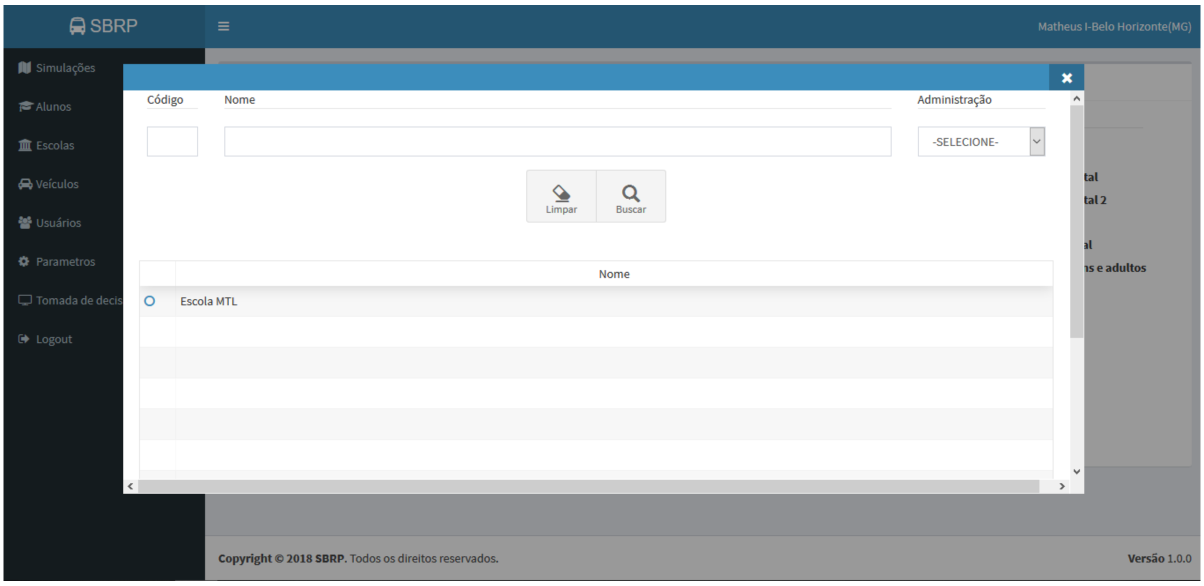


Figure 16 – Popup used for the selection of schools.

When registering students, their name, the school they are enrolled in, their date of birth, the turn of their classes, the direction, whether or not they have reduced mobility and/or need special vehicles, their category, and their address. Two of these deserve special attention. First, when choosing a school, initially a dropdown containing all schools in the municipality was presented, however, it was often a long list that needed to be scrolled through and therefore not very user-friendly. The solution was to add a button that opens a popup containing a simplified version of the search tool for schools that could be used to find the appropriate one (figure 16). Second, the input of the address is one of the two cases that required geographical information that users could not be expected to have on hand, as such the popup described in 3.2.1 and shown in figure (figure 16) was included in this section.

Students can also be "deactivated" either during creation or by the update form. Deactivated students will not be taken into account by any simulations created after they were deactivated. Deactivated students can be reactivated in the update form to be included in future simulations.

3.2.4 Schools

When searching for a specific school, a user may filter by ID, name, or administration (municipal, state, or federal). Like with the students, a looking glass shaped icon is presented together with the results of the search, that opens a map showing the position of the school.

When registering schools, users are asked to fill in the name, the administration responsible for the school, the courses offered (fundamental school, middle school, special education, etc.), the schedule of arrivals and departures, including the time the gates are opened and closed as well as how early or late the bus may arrive, and the address. Once again similar to the students, the map for the selection of addresses (figure 15) is present.

3.2.5 Vehicles

Vehicles can be searched by plate number and number of seats. When inserting a vehicle, the plate number, the number of seats, the year of manufacture, the average consumption of fuel, the type of fuel consumed, the average consumption of lubricant, the average wear of the tires, the number of tires, the market value of the vehicle, the total value of taxes, the average cost of maintenance, and whether the vehicle is adapted for the transport of special education students must be provided. Most of these values are used for the calculation of the total cost of the solution.

Like the students, a vehicle can be "deactivated" if there is a need to exclude it from a simulation, and later reactivated.

3.2.6 Users

Users who possess administrator privileges are the only ones capable of accessing the "Users" section of the application. The link will not even be present to those who do not have permission and should they try to access it via the URL, the system automatically redirects to the main page.

When searching for users, the results can be filtered by name, email and access level. When registering a user, an email address must be provided to serve as a username to the new user. A name, password at least eight characters long, a access level and a city must be provided. Before it is possible to select a city, a state must be selected in order to filter the results. Although Administrators and state level users are not restricted to a single city, one can still be provided to serve as a "default" value when logging into the system.

3.2.7 Parameters

The parameters section is unique in that it does not have a "New" component. That is because only one set of parameters may be active at a time, therefore, any new set must necessarily overwrite the old one where the user is concerned. On the database, however, the opposite is true. Since the calculation of costs is always based on the set of parameters at the time of the request, historical values must be kept. Therefore, from the database's viewpoint, sets of parameters cannot ever be updated, with new ones being created every time a change is deemed necessary.

When accessing the parameters section, the user is presented with the current set of parameters, including but not limited to, the average prices of fuel, lubricant, and tires, the average wages of drivers and monitors, the number of school days on the year, taxes, the depreciation rate of vehicles.

At the bottom of the page, there are two buttons, one leading to a table showing past values, and one that leads to the form for the update of current parameters.

4 Final considerations

This paper described the development of a web tool for the management of data related to the School Bus Routing Problem. In order to do so, three steps were followed: a revision of the literature surrounding the problem itself, a revision of tools and technologies available to tackle the problem, and the development itself.

The first step was necessary to ensure the proper execution of the final objective. The idea of trying to create a solution without first understanding the problem is, at best, dubious, so an analysis of the SBRP and of it's predecessor, the VRP, was essential. In order to ensure such a strong foundation to work on, books and papers related to the problem were used.

The second step, though not as fundamental as the first one, was every bit as important. Many tools were considered, with the pros and cons of each being weighted until the set of React, AdminLTE, Java, and PostgreSQL was decided on. The tools selected can make or break an application, and the final result of this project could have been significantly different if even a single one of the final set had been changed.

Finally, the development of the web tool itself was the main focus of this work. The final result was an application with all of the necessary functions to be put into production. The back-end API followed a REST architecture, and the front-end based on React's component architecture. While excellency is a state, and not a form, and therefore something that must be constantly sought even after being achieved, else it is lost, the author is satisfied with the end result of the development.

Overall, the author considers the project a success, both from the viewpoint of being an application for the management of data related to the SBRP, as well as from the viewpoint of being a learning experience.

4.1 Future works

Future works to be done on the project could be related to a few points not addressed by this particular work, or to the evolution of it. On the first front, many modules are being developed in parallel with this one to tackle the SBRP as a whole, including but not limited to a module for the definition of bus stops and one for the real-time tracking of vehicles. As such, the suggestions made in this section will focus on the second front presented, the improvement upon this work.

This work is focused on a tool intended to be used by workers at the school or towns where the system would be inplemented, and this singular focus has its drawbacks. Mainly, the need for accurate geographical data could be easily sated with a sister application meant to be used on mobile devices by the students or their parents. Such a system could take advantage of the location services already provided by mobile devices to acquire more accurate data, with less effort from the users.

Yet another point that may prove troublesome is the lack of an option to insert students into the system in bulk. Manually inserting students one by one may prove a tedious and error prone process, so a modification that could allow many students to be inputed at once may be desirable.

One point of great importance that was not developed for this application was that of automated testing. Manual testing of software is a labor intensive and human error prone activity, so the addition of automated testing would significantly increase the level of confidence to the project. It is important to note, however, that automated testing is meant to be applied in conjunction with manual testing, not replace it (BERNER; WEBER; KELLER, 2005). User experience tests are also recomended.

Another point that could be focused on is the recent (at the time of writing) addition of React Hooks. Hooks are designed to increase performance and reusability, as well as decrease the size of components on React applications. Hooks were added to React fairly late in the development of this application, and major refactoring to include them was simply impossible. A future project might be able to refactor the application to use the new technique.

Future projects may also want to develop internationalization for the webpage. Currently, the system is only available in brazilian portuguese, so should a future oportunity appear to launch the application on an international scale, translations need to be provided.

References

ALMSAEED, A. *AdminLTE Control Panel Template*. 2014. Disponível em: <https://adminlte.io/>. Citation on page 23.

ALOMARI, Z. et al. Comparative studies of six programming languages. *arXiv preprint arXiv:1504.00693*, 2015. Citation on page 23.

BECK, K. et al. Manifesto for agile software development. 2001. Citation on page 29.

BERNER, S.; WEBER, R.; KELLER, R. K. Observations and lessons learned from automated testing. In: ACM. *Proceedings of the 27th international conference on Software engineering*. [S.l.], 2005. p. 571–579. Citation on page 43.

BODIN, L. D.; BERMAN, L. Routing and scheduling of school buses by computer. *Transportation Science*, INFORMS, v. 13, n. 2, p. 113–129, 1979. Citation on page 18.

BÖGL, M.; DOERNER, K. F.; PARRAGH, S. N. The school bus routing and scheduling problem with transfers. *Networks*, Wiley Online Library, v. 65, n. 2, p. 180–203, 2015. Citation on page 20.

BOWERMAN, R.; HALL, B.; CALAMAI, P. A multi-objective optimization approach to urban school bus routing: Formulation and solution method. *Transportation Research Part A: Policy and Practice*, Elsevier, v. 29, n. 2, p. 107–123, 1995. Citation on page 17.

BRACA, J. et al. A computerized approach to the new york cityschool bus routing problem. *IIE transactions*, Springer, v. 29, n. 8, p. 693–702, 1997. Cited 2 times on page(s) 18 and 19.

CACERES, H.; BATTA, R.; HE, Q. School bus routing with stochastic demand and duration constraints. *Transportation science*, INFORMS, v. 51, n. 4, p. 1349–1364, 2017. Citation on page 20.

CACERES, H.; BATTA, R.; HE, Q. Special need students school bus routing: Consideration for mixed load and heterogeneous fleet. *Socio-Economic Planning Sciences*, Elsevier, v. 65, p. 10–19, 2019. Citation on page 19.

CHEN, D.-S. et al. A bus routing system for rural school districts. *Computers & Industrial Engineering*, Elsevier, v. 19, n. 1-4, p. 322–325, 1990. Citation on page 17.

CHEN, D.-S.; KALLSEN, H. A.; SNIDER, R. C. School bus routing and scheduling: an expert system approach. *Computers & Industrial Engineering*, Elsevier, v. 15, n. 1-4, p. 179–183, 1988. Citation on page 18.

CORBERÁN, A. et al. Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the operational research society*, Springer, v. 53, n. 4, p. 427–435, 2002. Citation on page 19.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. *Management science*, Informs, v. 6, n. 1, p. 80–91, 1959. Citation on page 15.

ELLEGOOD, W. A.; CAMPBELL, J. F.; NORTH, J. Continuous approximation models for mixed load school bus routing. *Transportation research part B: Methodological*, Elsevier, v. 77, p. 182–198, 2015. Cited 2 times on page(s) 16 and 18.

ELLEGOOD, W. A. et al. School bus routing problem: Contemporary trends and research directions. *Omega*, Elsevier, 2019. Cited 4 times on page(s) 16, 17, 18, and 19.

FEDOSEJEV, A. *React. js Essentials*. [S.l.]: Packt Publishing Ltd, 2015. Citation on page 22.

GENDREAU, M. et al. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, Elsevier, v. 26, n. 12, p. 1153–1173, 1999. Citation on page 19.

GILBERT, H.; HANDSCHUH, H. Security analysis of sha-256 and sisters. In: SPRINGER. *International workshop on selected areas in cryptography*. [S.l.], 2003. p. 175–193. Citation on page 26.

GOLDEN, B. L.; RAGHAVAN, S.; WASIL, E. A. *The vehicle routing problem: latest advances and new challenges*. [S.l.]: Springer Science & Business Media, 2008. v. 43. Cited 3 times on page(s) 12, 15, and 19.

HALILI, F.; RAMADANI, E. Web services: a comparison of soap and rest services. *Modern Applied Science*, v. 12, n. 3, p. 175, 2018. Cited 2 times on page(s) 24 and 25.

INEP. *Sinopses Estatísticas da Educação Básica*. 2019. Disponível em: <http://portal.inep.gov.br/sinopses-estatisticas-da-educacao-basica>. Citation on page 30.

JONES, M.; BRADLEY, J.; SAKIMURA, N. *JSON Web Token (JWT)*. [S.l.], 2015. Disponível em: <https://tools.ietf.org/html/rfc7519>. Citation on page 27.

KALUŽA, M.; TROSKOT, K.; VUKELIĆ, B. Comparison of front-end frameworks for web applications development. *Zbornik Veleučilišta u Rijeci*, Veleučilište u Rijeci, v. 6, n. 1, p. 261–282, 2018. Cited 2 times on page(s) 21 and 22.

KANG, M. et al. Development of a genetic algorithm for the school bus routing problem. *International Journal of Software Engineering and Its Applications*, v. 9, n. 5, p. 107–126, 2015. Citation on page 12.

KUMAR, S. N.; PANNEERSELVAM, R. A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, Scientific Research Publishing, v. 4, n. 03, p. 66, 2012. Citation on page 15.

LI, F.; GOLDEN, B.; WASIL, E. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & operations research*, Elsevier, v. 34, n. 10, p. 2918–2930, 2007. Citation on page 16.

LI, L.; FU, Z. The school bus routing problem: a case study. *Journal of the Operational Research Society*, Springer, v. 53, n. 5, p. 552–558, 2002. Cited 2 times on page(s) 12 and 19.

MESBAH, A.; DEURSEN, A. V. Migrating multi-page web applications to single-page ajax interfaces. In: IEEE. *11th European Conference on Software Maintenance and Reengineering (CSMR'07)*. [S.l.], 2007. p. 181–190. Citation on page 21.

MIRANDA, D. M. et al. A multi-loading school bus routing problem. *Expert Systems with Applications*, Elsevier, v. 101, p. 228–242, 2018. Cited 3 times on page(s) 18, 19, and 20.

MUEHLEN, M. Z.; NICKERSON, J. V.; SWENSON, K. D. Developing web services choreography standards—the case of rest vs. soap. *Decision Support Systems*, Elsevier, v. 40, n. 1, p. 9–29, 2005. Citation on page 25.

NANZ, S.; FURIA, C. A. A comparative study of programming languages in rosetta code. In: IEEE. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. [S.l.], 2015. v. 1, p. 778–788. Citation on page 24.

NEIS, P.; ZIPF, A. Analyzing the contributor activity of a volunteered geographic information project—the case of openstreetmap. *ISPRS International Journal of Geo-Information*, Molecular Diversity Preservation International, v. 1, n. 2, p. 146–165, 2012. Citation on page 27.

NEWTON, R. M.; THOMAS, W. H. Design of school bus routes by computer. *Socio-Economic Planning Sciences*, Elsevier, v. 3, n. 1, p. 75–85, 1969. Citation on page 15.

NEWTON, R. M.; THOMAS, W. H. Bus routing in a multi-school system. *Computers & Operations Research*, Elsevier, v. 1, n. 2, p. 213–222, 1974. Citation on page 19.

NORMAN, D. *The design of everyday things: Revised and expanded edition*. [S.l.]: Constellation, 2013. Citation on page 20.

O'GRADY, S. *The RedMonk Programming Language Rankings: January 2017*. 2017. Disponível em: <https://redmonk.com/sogrady/2017/03/17/language-rankings-1-17/>. Citation on page 24.

O'GRADY, S. *The RedMonk Programming Language Rankings: January 2018*. 2018. Disponível em: <https://redmonk.com/sogrady/2018/03/07/language-rankings-1-18/>. Citation on page 24.

O'GRADY, S. *The RedMonk Programming Language Rankings: January 2019*. 2019. Disponível em: <https://redmonk.com/sogrady/2019/03/20/language-rankings-1-19/>. Cited 2 times on page(s) 24 and 25.

OLIVEIRA-CIABATI, L. et al. Sisprenacel–mhealth tool to empower prenacel strategy. *Procedia Computer Science*, Elsevier, v. 121, p. 748–755, 2017. Citation on page 23.

PARK, J.; KIM, B.-I. The school bus routing problem: A review. *European Journal of operational research*, Elsevier, v. 202, n. 2, p. 311–319, 2010. Cited 5 times on page(s) 12, 15, 16, 17, and 19.

POSTGRESQL. *PostgreSQL 9.4.20 Documentation*. 1999. Disponível em: <https://www.postgresql.org/docs/9.4/datatype-geometric.html>. Citation on page 26.

POSTGRESQL. *PostgreSQL homepage*. 2018. Disponível em: <https://www.postgresql.org/>. Citation on page 26.

PRINS, C. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, Elsevier, v. 31, n. 12, p. 1985–2002, 2004. Citation on page 12.

REACT. *React Documentation*. 2018. Disponível em: <https://reactjs.org/docs/components-and-props.html>. Citation on page 21.

REACT. *React Documentation*. 2018. Disponível em: <https://reactjs.org/docs/introducing-jsx.html>. Citation on page 21.

React Training. *React Router*. 2019. Disponível em: <https://reacttraining.com/react-router/>. Citation on page 22.

RIPPLINGER, D. Rural school vehicle routing problem. *Transportation Research Record*, SAGE Publications Sage CA: Los Angeles, CA, v. 1922, n. 1, p. 105–110, 2005. Cited 2 times on page(s) 17 and 19.

RODRIGUEZ, A. Restful web services: The basics. *IBM developerWorks*, v. 33, 2008. Citation on page 25.

SAVAS, E. S. On equity in providing public services. *Management Science*, INFORMS, v. 24, n. 8, p. 800–808, 1978. Citation on page 19.

SCHILDT, H. *Java 2 a beginer's guide*. [S.l.]: The McGraw-Hill Companies, Inc, 2003. Citation on page 24.

SCHWABER, K.; SUTHERLAND, J. The scrum guide. *Scrum Alliance*, v. 21, 2011. Citation on page 29.

SPADA, M.; BIERLAIRE, M.; LIEBLING, T. M. Decision-aiding methodology for the school bus routing and scheduling problem. *Transportation Science*, INFORMS, v. 39, n. 4, p. 477–490, 2005. Cited 3 times on page(s) 17, 18, and 19.

TIHOMIROVS, J.; GRABIS, J. Comparison of soap and rest based web services using software evaluation metrics. *Information Technology and Management Science*, De Gruyter Open, v. 19, n. 1, p. 92–97, 2016. Citation on page 25.

TOTH, P.; VIGO, D. *The vehicle routing problem*. [S.l.]: SIAM, 2002. Citation on page 15.

Appendix

APPENDIX A – Materials elaborated by the author

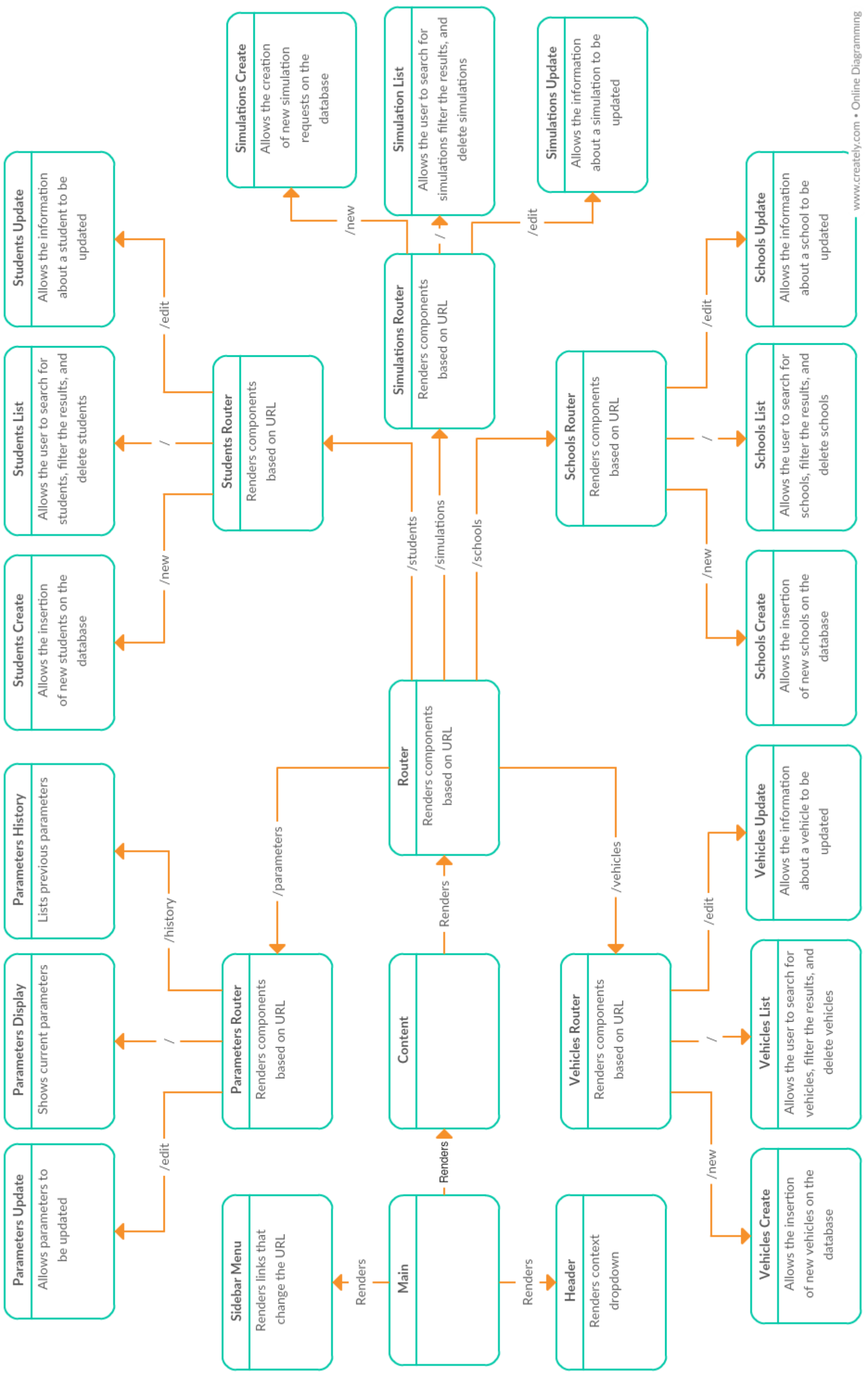

Main
Renders
Sidebar Menu
Renders links that change the URL
Renders
Header
Renders context dropdown
Renders
Content
Renders
Router
Renders components based on URL
/parameters
Parameters Router
Renders components based on URL
/edit
Parameters Update
Allows parameters to be updated
/
Parameters Display
Shows current parameters
/history
Parameters History
Lists previous parameters
/vehicles
Vehicles Router
Renders components based on URL
/new
Vehicles Create
Allows the insertion of new vehicles on the database
/
Vehicles List
Allows the user to search for vehicles, filter the results, and delete vehicles
/edit
Vehicles Update
Allows the information about a vehicle to be updated
/students
Students Router
Renders components based on URL
/new
Students Create
Allows the insertion of new students on the database
/
Students List
Allows the user to search for students, filter the results, and delete students
/edit
Students Update
Allows the information about a student to be updated
/simulations
Simulations Router
Renders components based on URL
/new
Simulations Create
Allows the creation of new simulation requests on the database
/
Simulation List
Allows the user to search for simulations filter the results, and delete simulations
/edit
Simulations Update
Allows the information about a simulation to be updated
/schools
Schools Router
Renders components based on URL
/new
Schools Create
Allows the insertion of new schools on the database
/
Schools List
Allows the user to search for schools, filter the results, and delete schools
/edit
Schools Update
Allows the information about a school to be updated
www.creately.com • Online Diagramming

Figure 17 – Detailed structure of the components of the front-end.

TERMO DE RESPONSABILIDADE

Eu, **Matheus Teixeira Lemos** declaro que o texto do trabalho de conclusão de curso intitulado "***WEB solution for the management of data related to the School Bus Routing Problem***" é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 12 de julho de 2019

matheus teixeira lemos

Matheus Teixeira Lemos

DECLARAÇÃO DE CONFORMIDADE

Certifico que o(a) aluno(a) **Matheus Teixeira Lemos**, autor do trabalho de conclusão de curso intitulado "***WEB solution for the management of data related to the School Bus Routing Problem***" efetuou as correções sugeridas pela banca examinadora e que estou de acordo com a versão final do trabalho.

João Monlevade, 12 de julho de 2019.

[illegible]

Rafael Frederico Alexandre