INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS TERESINA CENTRAL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LUAN MACHADO PONTES

STARBUS - UMA PLATAFORMA DE DADOS SOBRE O TRANSPORTE PÚBLICO

TERESINA, PIAUÍ

2020

LUAN MACHADO PONTES

STARBUS - UMA PLATAFORMA DE DADOS SOBRE O TRANSPORTE PÚBLICO

Projeto apresentado à Banca Examinadora como requisito para aprovação na disciplina de Trabalho de Conclusão de Curso II do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí.

Orientador: Prof. Dr. Adalton Sena

TERESINA, PIAUÍ

2020

LUAN MACHADO PONTES

STARBUS - UMA PLATAFORMA DE DADOS SOBRE O TRANSPORTE PÚBLICO

Projeto apresentado à Banca Examinadora como requisito para aprovação na disciplina de Trabalho de Conclusão de Curso II do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí.

Aprovado pela banca examinadora em _____.

BANCA EXAMINADORA:

Prof. Dr. Adalton Sena
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

Prof. M^{e}. TBD
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

Prof. M^{e}. TBD
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

TERESINA, PIAUÍ

2020

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
WWW	World Wide Web
REST	Representational State Transfer
HTML5	Hypertext Markup Language, versão 5
JSON	JavaScript Object Notation
CSS	Cascading Style Sheets
MVC	Model View Controller
CDI	Contexts and Dependency Injection
ORM	Object/Relational Mapping
JDBC	Java DataBase Connection
SGBD	Sistema de Gerenciamento de Banco de Dados
UUID	Universally Unique Identifier

SUMÁRIO

1 INTRODUÇÃO

Após a Segunda Guerra Mundial, os processos de urbanização e industrialização do Brasil, levaram milhões de pessoas do campo à cidade. Segundo o IBGE, nos anos 60, mais da metade da população brasileira era considerada rural, pouco mais de 50 anos depois, 84,4% da população brasileira reside em áreas urbanas [**??**]. Os investimentos em infraestruturas não acompanharam o ritmo acelerado dessa mudança, o que levou ao agravamento dos problemas urbanos, incluindo os relacionados à mobilidade e mais diretamente o transporte público.

Diversos problemas afetam os passageiros em seu dia a dia, no entanto a superlotação provavelmente é o que mais interfere na qualidade de vida dos usuários. Algumas empresas de ônibus, a fim de solucionar tal problema, colocam diversos veículos na mesma linha em um curto intervalo de tempo durante os horários de pico, mas a ansiedade do usuários de chegar em casa, a instabilidade do horário dos ônibus e a incerteza se o ônibus extra já passou ou não, faz com que os passageiros normalmente peguem o primeiro veículo disponível, causando o mau aproveitamento da frota. Nessa caso a falta de informação sobre os veiculos disponíveis tem impacto direto sobre qualidade de vida dos usuário.

A espera de ônibus faz os usuário se exporem muitas vezes a uma situação de risco e desconforto. Cada minuto em um ponto de ônibus o passageiro fica exposto a intempéries do tempo, como chuva e sol e não raro, assaltos são registrados em tais locais. Muitas vezes o usuário obriga-se a estar na parada muito tempo antes do ônibus passar, por vezes, deixando de fazer suas atividades, perdendo sono ou deixando de passar mais tempo com a família, caracterizando outro impacto negativo que falta de informação sobre o transporte público pode ter sobre a qualidade de vida das pessoas. Aumentar a frota de veículos para diminuir o tempo de espera pode ser economicamente inviável, mas oferecer a informação da quantidade de veículo disponíveis, bem como suas respectivas localizações e horários, pode ajudar a diminuir a quantidade de tempo que o usuário precisará ficar na parada sem elevar consideravelmente os custos operacionais, possibilitando até mesmo a diminuição dos preços para o usuário final por tonar possível a diminuição da frota.

Do ponto de vista das autoridades que cuidam do sistema como um todo, existe a dificuldade de levantar problemas em pontos específicos do sistema. Por exemplo, como o órgão responsável pelos os pontos de ônibus da cidade pode saber se os tais atendem as expectativas dos usuários? Ou como saber a visão do usuário a respeito da frota de veículos de uma determinada linha ou empresa? Atualmente cada órgão possui seu canal de contato com o usuário, que poderiam colher tais informações, no entanto tais canais requerem um esforço considerável do usuário, que pode desestimular o engajamento e a perda do sentimento que levaria ele a apontar problemas. Por outro lado as atuais plataformas digitais permitem o aproveitamento do sentimento de engajamento de forma imediata e com um esforço cosideravelmente menor.

O presente trabalho descreve o desenvolvimento de uma plataforma que ofereça informações sobre o transporte público a desenvolvedores de aplicativos móveis, além oferecer um repositório centralizado de avaliações realizadas por usuário sobre diversos aspectos do transporte público. Com tal ferramenta os usuários poderão ter acesso fácil a informações sobre os horários, intinerarios, localizão de veiculos e linhas por paradas, que podem ajudar a resvolver os problemas citados anteriormente. Por outro lado, os mesmos poderão conceber uma visão sobre sistema transporte público local, por avaliarem pontos de ônibus ou estado de conservação dos veículos, informações essas que podem ser uteis as entidades que cuidam do transporte público da cidade.

1.1 OBJETIVOS

Abaixo são listados os objetivos gerais e específicos do presente trabalho.

1.1.1 Objetivo Geral

O objectivo geral deste trabalho é descrever o desenvolvimento de uma plataforma que centralize os dados do transporte público de Teresina e que facilite o desenvolvimento de novas aplicações voltadas para a melhoria da mobilidade urbana.

1.1.2 Objetivos Específicos

- Descrever os processo de desenvolvimento de uma REST API web, incluindo as decisões tecnicas tomadas para a mesma.
- Demostrar o funcionamento da plataforma criada.
- Expor a documentação da API exposta.

2 TECNOLOGIAS ENVOLVIDAS

As tecnologias utilizadas na construação da plataforma Starbus são divididos em três grupos de ferramentas. Abaixo é específicado cada grupo, bem como as ferramentas pertencentes a cada um deles.

O primeiro grupo abrange as ferramentas utilizadas para a construação da plataforma em si, sendo a linguagem de programação, o framework e banco de dados utilizado na aplicação. A liguagem Ruby na versao 2.2 é uma linguagem de curta curva de aprendizagem, dispondo de um conjunto de ferramentas maduras e com abordagem pragmática para desenvolvimento. Faz parte de seu ferramental o framework Grape, que tem se mostrado uma solução prática para a construção de uma API REST, focando em oferecer uma interface simples para manipulação de dados oriundos de requições web. Por isso, Grape foi utilizado na concepção da API REST da plataforma. O dados são armazenados em bando de dados Postgres, na versão 12.1, que é de fácil configuração e possui uma comunidade ativa.

O segundo grupo é composto por ferramentas de suporte ao desenvolvimento. A primeira escolha foi o editor Vim que possui uma excelente suporte ao Ruby, além de permitir uma fácil alternacia entre o terminal e o editor, tornando pratica a execução de tarefas no terminal, que é tão comum no desenvolvimento Ruby. Para escrita dos testes da aplicação foi escolida a biblioteca Rspec na versao 3.9, que tem influenciado toda a comunidade de desenvolvimento de software com sua expressividade e clareza. Para acesso, debug e manipulação de dados no Postgres, foi utilizada a IDE DBeaver, na versao 7.0, um software open source com funcionalidades excelentes.

Por fim, para executar a plataforma Starbus em ambiente produtivo e assim atingir os objetivos específicos do trabalho, procurou-se uma ferramenta que pudesse trazer a menor carga de trabalho relacionada a infra estrutura de aplicação, visto que o foco do trabalho deveria ser no desenvolvimento da solução. Assim sendo, a plataforma Heroku foi a solução mais viável, visto que oferece um ambiente extremamente flexível e escalável com apenas poucas linha de configuração. Por exemplo, para se colocar no ar uma nova versão da aplicação basta realizar um git push para um determinado repositório da ferramenta.

3 MODELAGEM DO PROJETO

3.1 LEVANTAMENTO DE REQUISITOS

3.1.1 Requisitos Funcionais

Abaixo são listados os requisitos funcionais do sistema:

1. **Fornecimento de localização de veículos**: Fornecer informações de localização dos veículos do sistema no momento solicitado.
2. **Catálogo de linhas disponíveis no Sistema**: Listar as linhas disponíveis no sistema.
3. **Catálogo das paradas de ônibus**: Listar as paradas disponíveis no sistema e as linhas que estão disponíveis em cada uma delas.
4. **Avaliação de entidades do sistema**: Registar avaliações dos usuários sobre paradas e veículos do sistema de transporte público.
5. **Gerenciamento de usuários e autenticação**: Listar histórico de operações do usuário e ranking de usuários cadastrados.

3.1.2 Requisitos Não Funcionais

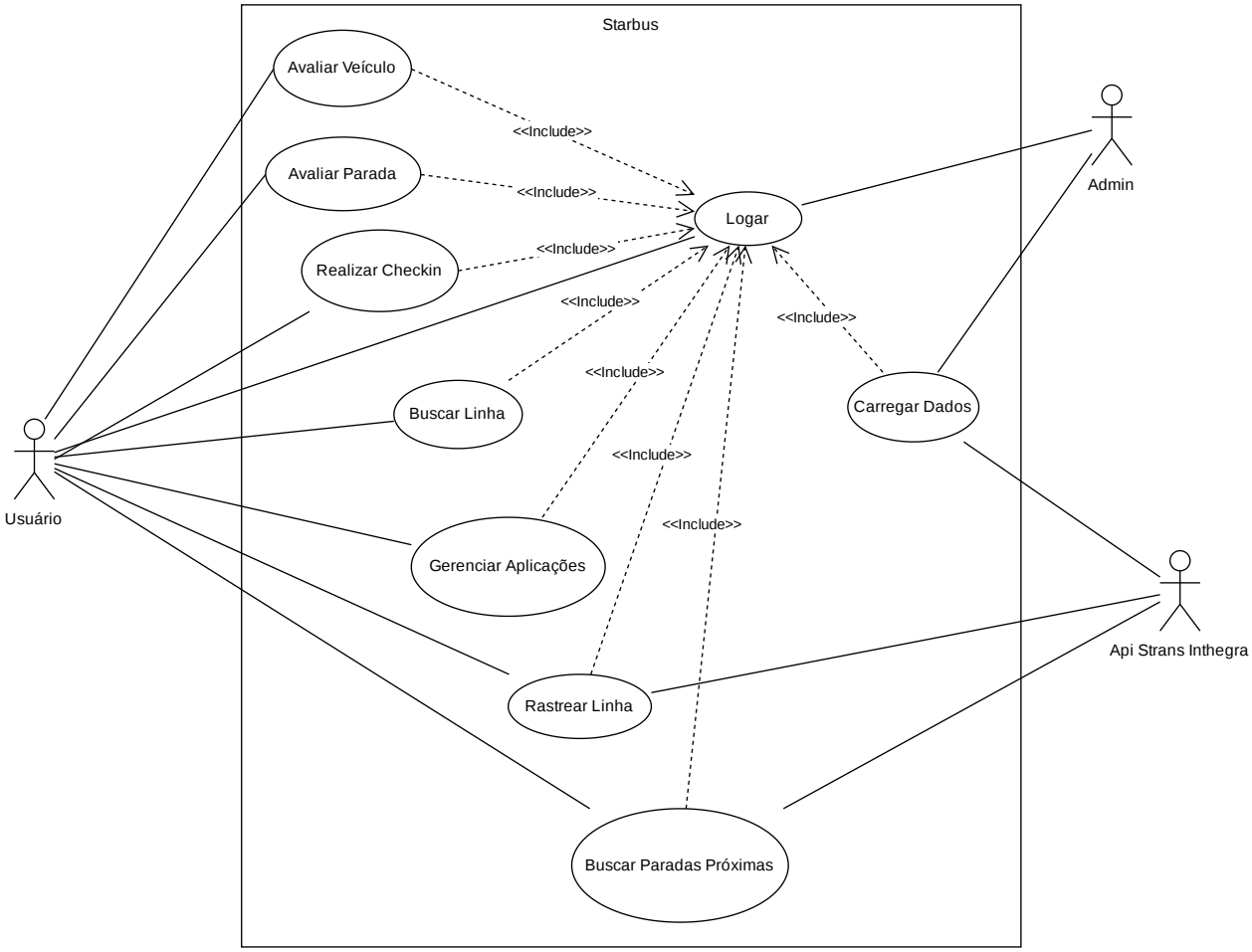
Abaixo são listados os requisitos não funcionais do sistema:

1. **Escalável e de alta disponibilidade**: Deve ser facilmente escalável para atender o aumento repetino de demanda ao mesmo tempo que mantem-se sempre disponível.
2. **Baixa carga de trabalho de Infra**: A baixa carga carga baixa de trabalho de infra pode ajudar na evolução da plataforma.
3. **API deve se simples e flexível**: Para ser atrativa ao uso de desenvolvidores a api deve ser bem documentada e flexível.

3.2 DIAGRAMAS DE CASOS DE USO

O diagrama de caso de uso desenvolvido mostrado na figura 1 descreve as possíveis interações do usuário através da ferramenta.

Figura 1 – Diagrama UML de Casos de Uso



Caso de uso Starbus

O diagrama acima expressa os seguintes casos de uso:

1. **Logar**: Ação de identificar-se como usuário para permitir a utilização dos recursos da plataforma representados pelos casos de uso seguintes. Para manter a credibiliade das informações fornecidas pelo usuários é essencial a identificação de cada fornecedor.

2. **Carregar Dados**: Usuário adminnistrador pode carregar as informações de linhas e paradas a partir da API Strans Inthegra. Essa ação será executada sempre que houver necessidade de atualizar a base do Starbus com novas informações da API Strans Inthegra.

3. **Avaliar Veículo**: Usuário devidamente identificado publica sua opinião sobre o veículo que utilizado em aspectos como acessibilidade, estado de conservação, lotação e conforto;

4. **Avaliar Parada**: Usuário devidamente identificado publica sua opinião sobre a parada de ônibus em aspectos como acessibilidade, estado de conservação, movimentação e conforto;

5. **Realizar Checkin**: Usuário devidamente identificado indica que no momento econtra-se em uma devida parada de ônibus ou veículo; Essa informação pode ser útil para levantar a lotação dos veículos e das paradas de ônibus.

6. **Buscar Linha**: Usuário fornece um código ou um termo para pesquisa que é usado para encontrar uma linha que ele deseja detalhes, como itinerário e localização dos veículos;

7. **Gerenciar Aplicações**: Um usuário pode ter várias aplicações, aqui ele pode gerencia-las;

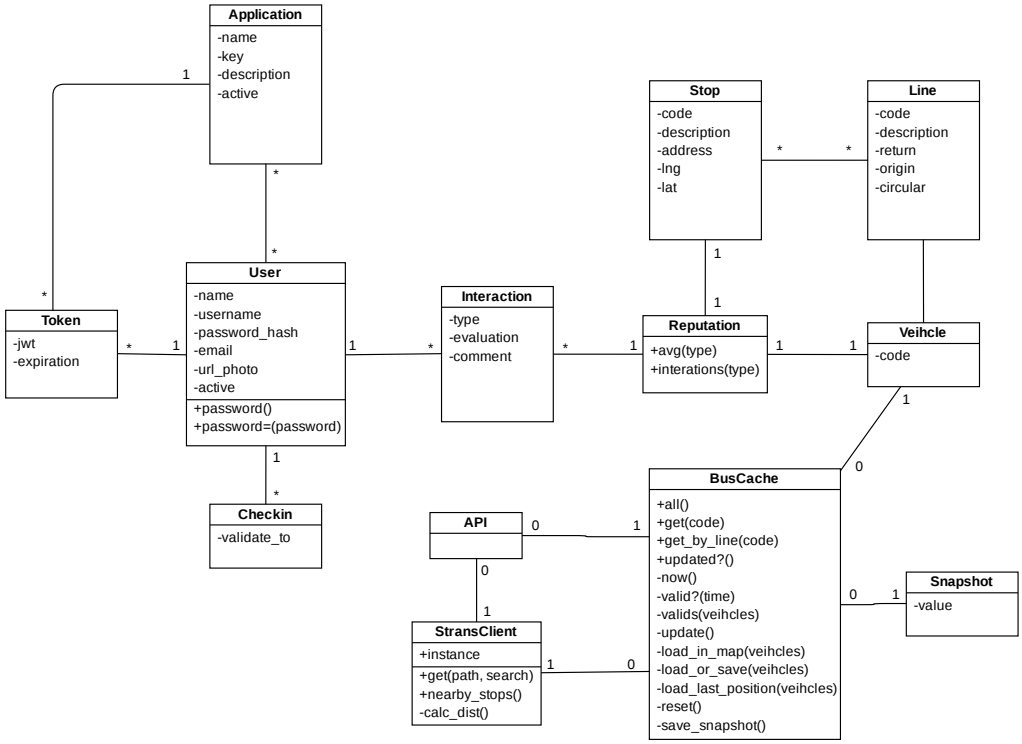
8. **Rastrear Linha**: Tendo um identificador de uma linha, que pode ser recuperado no caso de uso anterior, consulta a localização dos veículos dessa linha em tempo real com dados fornecidos pela Api Strans Inthegra;

9. **Buscar Paradas Próximas**:Usuário fornece uma localização, que pode ser a sua atual ou de um ponto específico que pode quer chegar, a plataforma retorna todas as paradas próximas aquele ponto, com os dados da parada inclusive a avaliação dos usuários sobre a parada;

3.3 DIAGRAMAS DE CLASSE

O modelo de dados da aplicação apresentado na Figura 2, onde é apresentado um diagrama UML de classes do projeto model.

Figura 2 – Diagrama UML de Casos de Uso



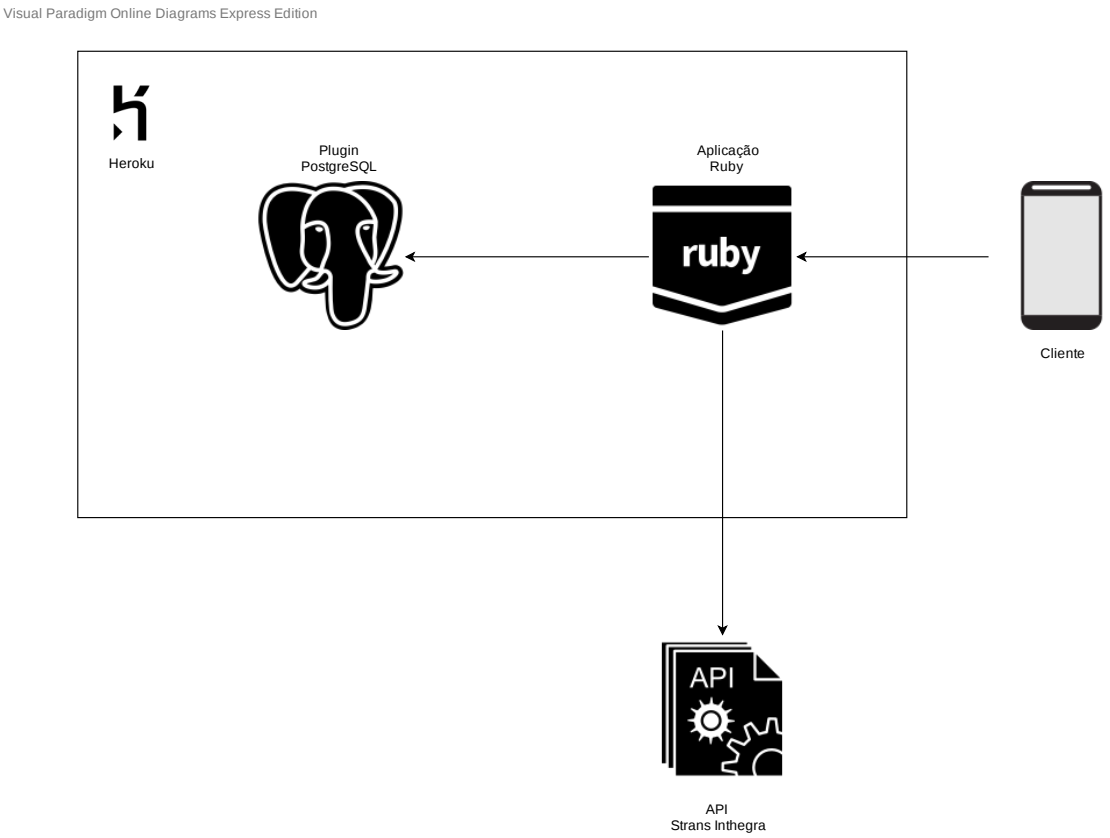
- **Application**: Representação de uma entidade que gera ou consome dados do starbus, sendo que esta pertence a um usuário e pode ter diversos usuários associados a ela.
- **User**: A identidade de um usuário do sistema que acessa a plataforma por meio de uma aplicação. Todo usuário possui username e password que são utilizados para a autenticação no sistema, somente após a autenticação é possível que um usuário possa consumir ou produzir informações para a plataforma.
- **Token**: Identidade Temporaria para que um usuário logado possa interajir com a plataforma. O token é gerado no momento do login e armazenado para ser validado posteriormente a cada iteração com a plataforma. Um Token tem um ciclo de vida bem definido e curto.
- **Checkin**: Um usuário pode notificar a sua localização em um determinado instante, pode ser uma parada de ônibus ou em um veículo específico, tal informação poderá ser utilizado pela plataforma para verificar as condições de lotação das paradas ou dos veículos em determinado momento. O Checkin representa essa informação e está diretamente relacionado ao caso de uso cinco, Realizar Chekin.

- **Interaction**: Dados e comportamento dos casos de uso de Avaliar Veículo e Avaliar Parada, que são sumarizados, na classe seguinte Reputation;
- **Reputation**: Sumarização de todas as avaliações realizadas por usuários a uma entidade do sistema de transporte público, representando a reputação de cada veículo ou parada do sistema a ela associada.
- **Stop**: Parada de ônibus, com seu indentificador, endereço e localização.
- **Line**: Linha de ônibus, com seu identificador e intinerário.
- **Veihcle**: Cada veículo de transporte de passageiros.
- **BusCache**: Otimização que permite diminuir o número de consultas a API strans, armazenamento em memória todas as consultas relizadas a api referente a localização de veículos. Guarda as regras para a definição de dados desatualizados, que permitem uma nova consulta a API.
- **Snapshot**: Cada vez que uma nova consulta de localização de veículos é realizada a API um JSON é armazenado para conservação do histórico da localização de veículos em diferentes horários do dia, futuramente esse dados podem ser usados para estudar o comportamento do sistema e sugerir melhorias.
- **StransClient**: Interface simplificada de consulta a API Stran Inthegra.

3.4 ARQUITETURA DO SISTEMA

A arquitetura do sistema é apresentado na Figura 3 onde são aprecsentados o componentes do sistema e sua respectivas descrições abaixo.

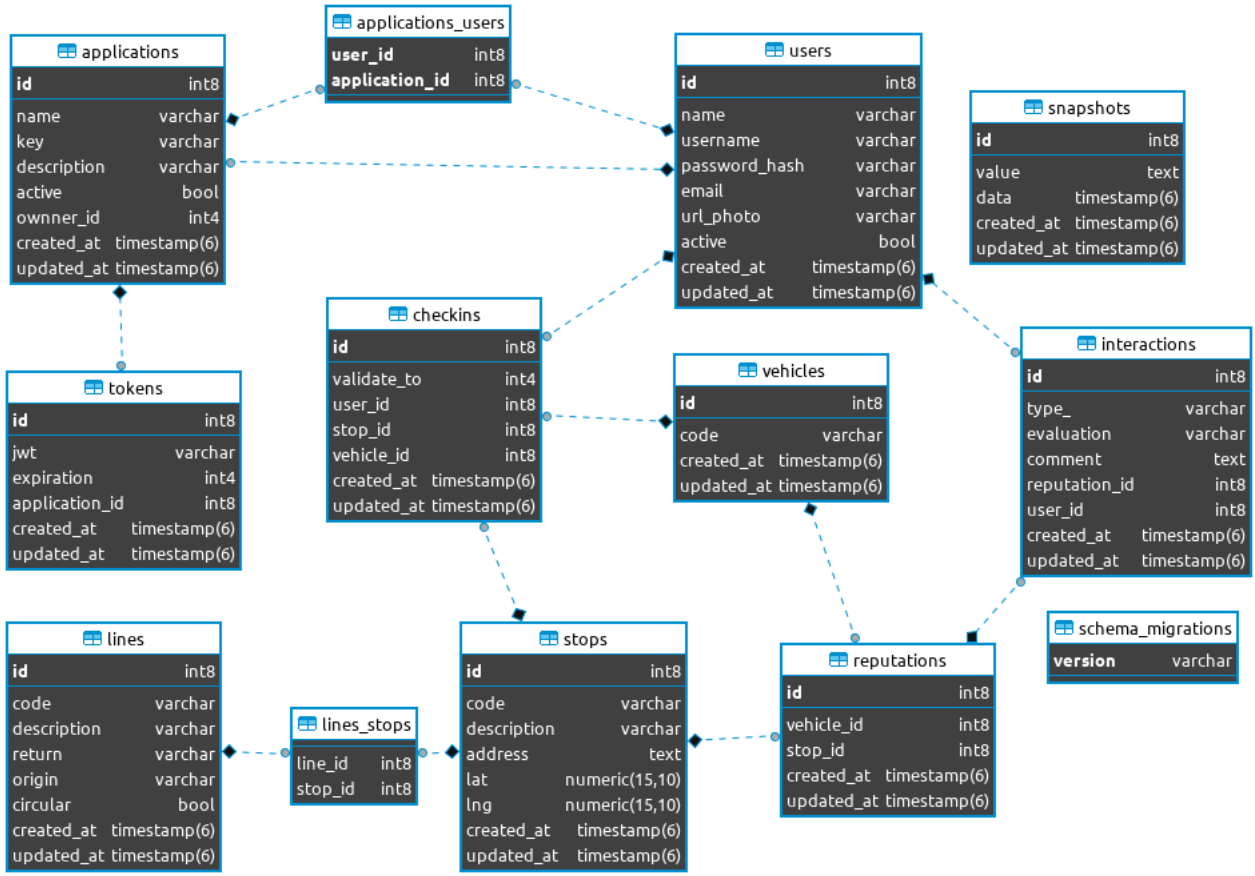
Figura 3 – Arquitetura do Sistema



3.5 DIAGRAMA DE ENTIDADES-RELACIONAMENTOS

Na figura 4 podemos ver a estrutura lógica usada como base para o banco de dados. A maioria representam as entidades detalhadas anteriormente no Diagrama de Classe, com exceção da entidade schema_migrations, que é utilizada pelo framework Active Record para armazenar o histórico de migrações utilizadas por ele.

Figura 4 – Diagrama entidade-relacionamento



3.6 API REST

Como já descrito, essa plataforma que utiliza o conceito de API REST e toda a comunição é feita por meio de métodos http, que são organizados sematicamente com base em resources, abaixo serão descritos os resources bem como os metodos que são disponíveis na API da plataforma. Toda a transmissão de dados será feita utilizand o padrão JSON, assim sendo o corpo das requisições bem como das resposta da api serão texto obedecendo as regras desse padrão amplamente difundido na WEB.

Todos as requisições a api deve ser feitas utilizando o cabeçalho de "Authorization"com o token de autenticação recuperado de "auth/login", intuitivamente essa chamada é a única que não é obrigatório a inclusão do cabeçalho de autenticação.

Por padrão os resources abaixo retornam os seguintes codigos http, para cada situação descrita.

- **200**: Solicitação executada com sucesso.
- **403**: Token inválido ou não enviado.
- **404**: O path não foi encontrado ou não existe entidade com id/código especificado.

- **415**: Solicitação fora do formato esperado pela API.
- **500**: Erro interno do servidor.

A url base da API é https://starbus-v2.herokuapp.com/v2/ que é seguida pelo resources detalhados a seguir.

3.6.1 /auth

Responsável pelos dados de autenticação na platafroma. Basicamente retorna o token de acesso que deve ser utilizando nas próximas requisições, e returna dados sobre o usuário autenticado com o com o token utilizado.

Método	Caminho	Descrição
POST	/login	Autentica o usuário e retorna o token que será utilizado posteriomente
GET	/sesion	Retorna os dados o usuário logado com token passado

Tabela 1 – Resource Auth

3.6.2 /applications

Gerenciamento de aplicações da plataforma que só pode ser feito pelo usuário que possui as aplicações. Assim cada usuário gerencia suas próprias aplicações.

Método	Caminho	Descrição
POST	/	Cria uma nova Application com os dados passados
GET	/	Retorna todas as Applications
PUT	/:id	Atualiza a Application com id espeficicado, como os dados passados
GET	/:id	Retorna a Application com o id espeficicado
DELETE	/:id	Desativa a Application com o id espeficicada

Tabela 2 – Resource Applications

3.6.3 /users

Gerenciamento de usuários.

Método	Caminho	Descrição
POST	/	Cria uma nova User com os dados passados
GET	/	Retorna todas as Users
PUT	/:id	Atualiza a User com id espeficicado, como os dados passados
GET	/:id	Retorna a User com o id espeficicado
DELETE	/:id	Desativa a User com o id espeficicado

Tabela 3 – Resource Users

3.6.4 /lines

As Linhas disponíveis na aplicação são fornecidas e gerenciadas pela API Strans, assim o Starbus não faz o gerenciamento dessas linhas, ele apenas carrega tais linhas de sua fonte. O resource abaixo carrega tais linhas e oferece algumas consultas as linhas na base.

Método	Caminho	Descrição
POST	/load	Carregas as linhas a partir da API Strans
GET	/	Retorna todas as Lines
GET	/?codigos[]=:codigo	Retorna as Lines com os códigos especificados
GET	/:codigo/vehicles/	Retorna os Vehicles com na rota da linha com o código passado

Tabela 4 – Resource Lines

3.6.5 /vehicles

Assim como as linhas a Lines, Veihcles não são gerenciados pelo Starbus, a cada consulta na Strans a plataforma se o Veihcle já está cadastrado em sua base, se não ela cadastra. Sendo assim, esse resource retorna simplesmente os Veihcles rodando no momento da requisição.

Método	Caminho	Descrição
GET	/	Retorna todas as Vehicles rodando no sistema nesse momento
GET	/:codigo	Retorna o Vehicle rodando no sistema com sua posição no momento
POST	/:codigo/checkin	Realiza o Checkin na Stop com o código passado

Tabela 5 – Resource Vehicles

3.6.6 /stops

Stops também não são gerenciados pela plataforma e são carregados por quando as Lines são carregadas, a cada Line carrega são automaticamente salvos as paradas associadas a ela.

Método	Caminho	Descrição
GET	/	Retorna todas as Stops do sistema
GET	/:codigo/lines	Retorna todas as Lines de uma Stop com o código especificado
GET	/closes/?lat=:lat&long=:long	Retorna todas as Stops proximas a localização passadas por meio de lat e long
POST	/:code/checkin	Realiza o Checkin na Stop com o código passado.

Tabela 6 – Resource Lines

3.6.7 /interations

Esse resource o é possível registrar as avaliações dos usuários sobre dois componentes do sistem de transpote píblico, paradas de ônibus e veículos.

Método	Caminho	Descrição
GET	/:criterio/vehicle/:codigo	Retorna a User com o id especificado
GET	/:criterio/stop/:codigo	Retorna todas as Users
POST	/:criterio/vehicle/:codigo	Cria uma nova User com os dados passados
POST	/:criterio/stop/:codigo	Cria uma nova User com os dados passados

Tabela 7 – Resource Interactions

Abaixo são listados os critérios que são utilizados para avaliar essas entidades, que nos caminhos são representados pelo parametro ":criterio".

- **accessibility**: Acessibilidade de acesso para pessoas dificuldade de locomoção.
- **comfort**: Conforto para o usuário.

- **safety**: Segurança, tanto no aspecto de uso como de segurança pública.
- **state**: Estado de conservação.

Par cada um dos critérios a acima o usuário pode associar uma avaliação listada a abaixo para a entidade que está avaliando que é especificada pelo ":codigo"da entidade.

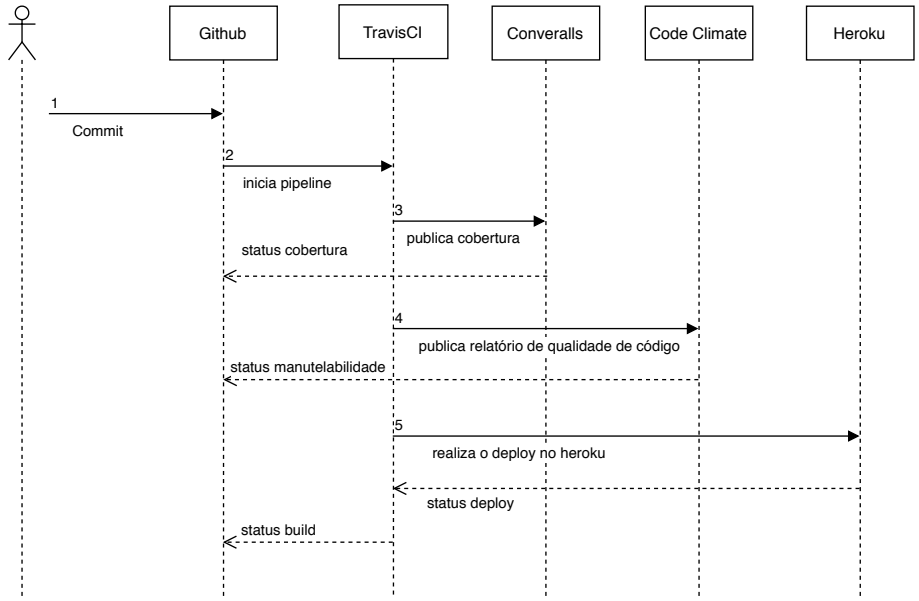
- **bad**: Ruim, longe de ser o ideal.
- **regular**: Regular, atende as necessidades do usuário.
- **good**: Muito bom, não deixando a desejar em nenhum aspecto.

4 SOFTWARE

4.1 IMPLANTAÇÃO

A platafroma utiliza-se do conceito de Continous Delivery e Continous Integration para garantir a entraga de software com agilidade e qualidade. O desenhho abaixo mostra como funciona esse conceito em prática e todas as ferramentas envolvidas para a implantação do software.

Figura 5 – Implantação



Implantação

4.2 TESTES

5 CONSIDERAÇÕES FINAIS

5.1 TRABALHOS FUTUROS