VanApp: Um sistema de gerenciamento de transporte escolar para a região de Hortolândia

Lucas V. Piza, Alexandre Grotta (Orientador)

[1]Análise e Desenvolvimento de Sistemas
Instituto Federal de São Paulo - campus Hortolândia (IFSP) – Hortolândia – SP – Brasil

lucas.piza@aluno.ifsp.edu.br, grotta@ifsp.edu.br

Abstract. *School transportation in the city of Hortolândia-SP is characterized by a diversity of service providers, including self-employed drivers who generally organize and advertise their activities independently. This decentralization makes it difficult for students to identify the routes and drivers that are most compatible with their location. In this context, this research proposes VanApp, a web system that centralizes the supply of school transportation vacancies, facilitating the search according to the students' preferred routes. The system was developed with an architecture based on the Model-View-Controller (MVC) pattern, using the framework W3.CSS in the front end, REST API for communication and Java in the back end, recent technologies that allow people, even with simpler smartphones, to access this solution. This project aims to contribute to the optimization of school transportation, promoting greater accessibility and efficiency in the connection between students and drivers.*

Resumo. *O transporte escolar na cidade de Hortolândia-SP é caracterizado por uma diversidade de prestadores de serviço, incluindo condutores autônomos que, geralmente, organizam e divulgam suas atividades de forma independente. Essa descentralização dificulta a identificação, por parte dos estudantes, daqueles trajetos e motorista mais compatíveis com a localização. Neste contexto, esta pesquisa propõe o VanApp, um sistema web que centraliza a oferta de vagas no transporte escolar, facilitando a busca conforme as rotas de preferência dos alunos. O sistema foi desenvolvido com uma arquitetura baseada no padrão Modelo-Visão-Controlador (MVC), utilizando o framework W3.CSS no front-end, API REST para comunicação e Java no back end, tecnologias recentes e que permitem que as pessoas, inclusive com smartphones mais simples, acessarem esta solução. Este projeto visa contribuir para a otimização do transporte escolar, promovendo maior acessibilidade e eficiência na conexão entre estudantes e condutores.*

1. Introdução

Partindo da vivência do pesquisador no mercado de transporte escolar, este projeto se inspira em melhorar o processo de localização de vans escolares por parte dos alunos e condutores, mais especificamente na região de Hortolânida. Existe a percepção que todo ano, entre as mudanças de semestre letivo e, consequentemente, as viradas de contratos de transporte escolar e as transferências de alunos entre instituições de ensino, é necessário um novo remanejamento dos alunos entre condutores, além da entrada de novos alunos

nesse mercado. O trabalho de divulgação ainda é realizado por meios físicos (por meio de *banners, folders*, panfletos e boca a boca) ou por plataformas mais genéricas, sem um foco específico, como as redes sociais. Esses métodos, no entanto, geram alguns problemas, já que as propagandas muitas vezes passam despercebidas, ou até geram desinformação conforme se aproxima o período de matrícula. Por se tratar de propagandas estáticas, sem atualização ao longo do tempo, ocorrem inúmeros desencontros de informação, uma vez que o mercado é dinâmico, com vagas para transporte que abrem e fecham rapidamente. Isso exige que o cliente procure outra opção caso a sua já tenha sido preenchida. Além das dificuldades enfrentadas pelos clientes, os próprios condutores também enfrentam desafios, pois arcam com o custo e o tempo para a confecção e divulgação dessas propagandas (sejam físicas ou digitais).

Essa dificuldade de encontrar vagas impactam diretamente no estudante, que em última instância, pode vir a ficar sem acesso a educação devido a não encontrar uma opção que se adeque as suas necesidades, como pode ser exemplificado por um caso em Criciúma - SC na Rádio Guarujá, onde foi relatado pelo pai de um estudante a dificuldade de se encontrar o transporte escolar em sua região e questionado a Secretaria de Educação de Orleans em 26/02/2025. No qual a porta voz da Secretaria ciotu que entende a dificuldade e a angústia do pai, mas que não teria nem como apresentar uma solução, uma vez que esse assunto não é tratado pela secretaria e sim pelas regulamentções municipais e estaduais, referentes tanto ao setor privado quanto público [Rádio Guarujá 2025].

Portanto, ao observar como esse gerenciamento e comunicação são realizados atualmente, identifica-se uma oportunidade de informatização no processo. As plataformas existentes são genéricas, tentando atender ao Brasil inteiro, mas não conseguiram uma adesão significativa por parte dos condutores, ou mesmo não conseguiram divulgar sua existência, tornando-se projetos descontinuados, como o **De van para a escola** e o **Urbes Escolar** que serão detalhados posteriormente [Urbes escolar 2024][De Van Para Escola 2024]. Desse modo, o VanApp, visa contribuir com a solução de parte destes problemas ao oferecer uma plataforma que concentra todos os condutores em um único lugar, tem como objetivo divulgar o trabalho dos prestadores de serviço e facilitar a busca por vagas de forma simples e direta.

1.1. Objetivo geral e Objetivos específicos

Diante desses desafios, o principal objetivo desta pesquisa foi propor uma solução *web* para gerenciar as vagas em linhas de transporte escolar na região de Hortolândia - SP por meio de uma plataforma *web*.

Assim sendo, os usuários podem filtrar as oportunidades que melhor atendam às suas necessidades, diminuindo o tempo gasto na pesquisa por transporte escolar. A plataforma também permite a divulgação do trabalho dos condutores e evita negociações desnecessárias (como propostas fora de rota ou casos de superlotação).

Para alcançar o objetivo principal mencionado, foi necessário que os seguintes objetivos fossem atendidos:

i **Perspectiva dos estudantes:** Os estudantes devem encontrar rapidamente vagas de transporte adequadas às suas necessidades, juntamente com as formas de contato do condutor para negociação.

ii **Perspectiva dos condutores:** Os condutores podem informar em seu perfil dados como itinerários, interesses de rotas para escolas e bairros específicos, formas de comunicação, entre outros, com o intuito de concentrar mais clientes e reduzir custos com propaganda.

2. Referencial teórico

Esta seção foi dividida em Quatro subseções: apresentação do funcionamento do negócio, termos utilizados na pesquisa, tecnologias utilizadas para o desenvolvimento da solução e trabalhos correlatos.

2.1. Negócio

O mercado de transporte escolar em Hortolândia - SP é, em grande parte, constituído por micro e pequenas empresas que dividem entre si, com o setor público e com os próprios pais/responsáveis, o transporte dos alunos mencionados no levantamento da [QEdu 2023], conforme a Tabela 1.

Tabela 1. Censo escolar em Hortolândia [QEdu 2023]

	Público	Privado
Escolas	91	31
Creche	5.402	533
Pré-escola	5.412	1.163
Fundamental I	13.998	4.178
Fundamental II	10.348	3.597
Ensino médio	8.928	1.352

Os condutores são responsáveis por buscar os alunos em casa ou em local combinado e conduzi-los até suas respectivas escolas, assim como realizar o trajeto inverso no horário de saída. Portanto, existem inúmeras combinações de profissionais, horários, escolas e bairros, sendo que cada horário realizado por um profissional, é definido como uma ”linha”. Ou seja, cada linha do condutor possui um conjunto de bairros e escolas que formam um trajeto em determinado horário.

Considerando o número de combinações e a quantidade de condutores independentes na região, é possível ter uma ideia da complexidade do problema que é procurar transporte manualmente, seja por redes sociais ou na porta das escolas.

2.2. Pesquisa

Os casos de uso servem para descrever os requisitos funcionais de um sistema e focar nas interações entre o sistema, seus usuários e outros sistemas. São uma forma de registrar o que o sistema deve realizar, retratando os comportamentos do sistema sob a visão de seus usuários [Jacobson 1992]. O diagrama de casos de uso possui dois elementos principais:

i **Ator:** Refere-se a qualquer entidade externa que se comunica com o sistema, sejam usuários, outros sistemas ou equipamentos [Jacobson 1992].

ii **Caso de uso:** Representa um conjunto de ações que o sistema efetua, trabalhando em conjunto com os atores, a fim de alcançar uma meta determinada. Cada caso de uso detalha uma função do sistema sob o ponto de vista do usuário ou ator [Jacobson 1992]..

Requisitos funcionais e não funcionais são duas categorias fundamentais para a definição das características e comportamentos de um sistema. Os requisitos funcionais especificam o que o sistema deve fazer (por exemplo, as funcionalidades do sistema). Já os requisitos não funcionais especificam de que maneira o sistema deve se comportar (por exemplo, os requisitos de desempenho, segurança e usabilidade) [Wiegers 2013].

2.3. Tecnologias utilizadas

As tecnologias utilizadas são elementos fundamentais da pesquisa, e, por isso, estão detalhadas neste tópico, dividido em banco de dados, tecnologias *front end*, tecnologias *back end* e comunicação entre *front end* e *back end*.

2.3.1. Banco de dados

O banco de dados é uma estrutura organizada onde se armazenam dados, facilitando a recuperação, manipulação ou exclusão desses dados de forma eficiente e controlada. Para o projeto em questão, foi utilizado o modelo de tabelas relacionais, um modelo amplamente utilizado e consolidado, no qual os dados são armazenados em tabelas. Cada coluna de uma tabela corresponde a uma característica do objeto representado por ela, e cada linha representa um registro com as características preenchidas [Oracle 2024].

Para começar a construção de um banco de dados, é fundamental partir de um SGBD (Sistema de Gestão de Banco de Dados), que é um *software* responsável por gerenciar todos os aspectos de um banco de dados, desde o controle de acesso até configurações mais complexas. Portanto, é comum delegar a gestão do banco de dados ao SGBD, o que facilita o desenvolvimento da aplicação [DataSUS 2019].

O MariaDB, é um SGBD gratuito e foi desenvolvido pelos mesmos criadores do *MySQL*, oferecendo comandos, interfaces, bibliotecas e APIs. Gerenciando bancos de dados relacionais por meio de do SQL, ele facilita a operação do banco de dados [MariaDB 2011]. O SQL, por sua vez, é uma linguagem de manipulação de banco de dados, padrão da *American National Standards Institute* (ANSI), utilizada com palavras-chave seguidas de uma estrutura específica para manusear as tabelas do banco, permitindo obter dados, editar, excluir, entre outras operações [W3Schools 2024a].

2.3.2. *Front end*

O termo *front end* refere-se a tudo o que o usuário pode interagir ou visualizar no sistema, como imagens, botões, gráficos, menus, entre outros [AWS 2023a]. Portanto, para a construção desses elementos, é comum o uso do HTML (*HyperText Markup Language*), que é uma linguagem de marcação, e não de programação, responsável por construir a camada mais básica da *web*, sendo o alicerce das páginas *web*. Assim, o HTML define toda a estrutura da página *web*, apoiando-se em outros complementos para estilizar seus elementos, como o CSS, que descreve a aparência e o estilo do que foi apresentado no HTML, ou o comportamento de seus componentes usando *JavaScript* [Mozilla 2024a].

Para estilizar o código HTML, foi necessário utilizar o CSS (*Cascading Style Sheets*), que permite ao desenvolvedor estilizar a página por meio de de uma linguagem de marcação, assim como o HTML. No entanto, enquanto o HTML é responsável

por gerenciar o conteúdo, o CSS fica encarregado de definir a aparência e o estilo da página, alterando cores, tamanhos, *layouts*, espaçamentos, e focando exclusivamente na estética [Mozilla 2024b]. Para aprimorar ainda mais a manipulação dos elementos visuais, é possível complementar a camada visual com o uso do *JavaScript*, uma linguagem de programação de alto nível, com tipagem dinâmica e fraca. Ela faz parte da tríade principal da *web*, composta por HTML, CSS e *JavaScript*, que, juntas, criam os visuais das páginas *web* [Mozilla 2024c].

O *JavaScript* não se limita apenas a funções e estilização. Trata-se de uma linguagem dinâmica baseada em multiparadigma e *thread* único (modelo de execução em que um programa realiza uma tarefa de cada vez, seguindo uma sequência linear de instruções), que suporta orientação a objetos, programação imperativa e declarativa, e também pode ser utilizada em ambientes que não são de navegador, como o *Node.js* [Mozilla 2024c].

O W3.CSS, é um *framework* gratuito voltado para estilizar páginas em diversos navegadores, além de ser responsivo em dispositivos móveis. Foi uma alternativa mais leve ao Bootstrap, já que utiliza apenas o CSS em sua composição. Basta importar os links CSS do W3.CSS no projeto e apontar os componentes da sua escolha por meio dos atributos *class* nas *tags* [W3Schools 2024b].

2.3.3. *Back end*

O termo *back end* refere-se a toda a estrutura por trás da navegação que o usuário realiza, à qual ele não tem acesso e não vê funcionando, como banco de dados, protocolos de comunicação, cálculos, integrações com outros sistemas, entre outros [AWS 2023a].

O Java é uma linguagem de programação orientada a objetos e multiplataforma. É uma linguagem versátil, utilizada em diferentes situações, como dispositivos móveis, *softwares desktop*, *big data* (processamento e análise de grandes volumes de dados), integração com *web*, entre outros [AWS 2023b]. Existem milhões de sistemas desenvolvidos em Java, e a linguagem ficou entre as três mais utilizadas por mais de uma década, destacando-se justamente pela sua facilidade e versatilidade [GitHub 2023]. Como complemento ao Java é comum o uso do *Spring Framework*, que tem como objetivo, facilitar a criação de aplicações Java, utilizando injeção de dependências. Ao integrá-lo a um projeto, obtém-se novos recursos para a aplicação, como módulos de persistência de dados, integração e desenvolvimento *web* [Spring 2024].

Para melhorar e resolver alguns problemas que podem ser encontrados durante o desenvolvimento, deixando o código mais eficiente, robusto e de fácil compreensão é comum aplicar padrões de desenvolvimento.

O MVC, ou *Model-View-Controller*, se trata de um padrão de projeto que divide o código em três partes, sendo elas o *model*, que é a parte onde ficam definidas todas as regras de negócio e lógica envolvidas na aplicação; a *view*, que é a parte que apresenta todas as informações da aplicação ao usuário, comunicando-se diretamente com ele por meio dos modelos renderizados; e a *controller*, que é a parte que pega as interações do usuário na *View* e redireciona suas operações para a *Model*, podendo retornar algo para a *View* após passar pelas regras de negócio ou não [Oracle 2007].

Dependendo do modelo de persistência de dados a ser escolhido, o processo de integração com a aplicação pode variar de acordo com o modelo adotado. Sendo assim, o DAO, ou *Data Access Object*, gerencia a conexão com o banco de dados, implementando o mecanismo de acesso necessário para o determinado modelo, e gera uma interface amigável para a aplicação acessar a base de dados, ocultando detalhes de implementação mais complexos [Oracle 2002].

O *Hibernate*, é um *framework* utilizado para mapear tabelas relacionais em Objetos JAVA, facilitando a comunicação entre a base de dados e o código da aplicação. O *Hibernate* mapeia as entidades do banco de dados em objetos JAVA, facilitando a escrita de consultas mais complexas e também as alterações dos dados [Hibernate 2025].

2.3.4. Comunicação entre *front end* e *back end*

Uma das formas de comunicação entre *front end* e *back end* é por meio de uma API (*Application Programming Interface*), que é uma forma de comunicação entre dois sistemas distintos, por meio de um acordo padrão que ambas as pontas entendem. Nesse processo, uma ponta requisita um serviço, dado ou processamento, enquanto a outra fica responsável por fornecer o que foi requisitado [RedHat 2024].

Com base na definição anterior sobre APIs, existe o modelo *REST* (*Representational State Transfer*), que é uma definição de uma dessas formas de comunicação, onde se transfere uma informação do estado da requisição ao solicitante, que acessa os serviços por *endpoints* (cabeçalhos HTTP que indicam os endereços das requisições). Essa comunicação é feita via HTTP por meio de algumas representações textuais, como o JSON [RedHat 2024]. O JSON, por sua vez, é um simples texto com uma certa formatação (baseada em objetos *JavaScript*), independente de linguagem, podendo ser usado para comunicação entre diferentes linguagens de forma leve, por se tratar de um arquivo de texto [W3Schools 2024c].

2.4. Trabalhos correlatos

Neste tópico, serão apresentados alguns trabalhos correlatos ao sistema VanApp. Uma pesquisa foi realizada por meio de do Google (sistema de busca online) com a finalidade de procurar sistemas de busca ou gerenciamento de vagas de condução escolar. Sendo assim, a seguir serão apresentados dois desses projetos: o **De van para escola** e o **Urbes Escolar**.

O sistema **De van para escola** foi criado para gerenciar o transporte escolar em todo o país. Sua proposta é bem interessante e completa. Além da funcionalidade de busca tradicional, o projeto possui funcionalidades mais complexas, divididas em dois grupos de público-alvo: para os condutores, existem as funcionalidades de criação de rota, gerenciamento de manutenção e gerenciamento financeiro; e para os estudantes, existem as funcionalidades de rastreamento de percurso, sistema de avaliação e *Chat* dentro do sistema [De Van Para Escola 2024].

O aplicativo **Urbes Escolar** explora uma visão bem mais simplista e concisa para a solução do problema, focando apenas na busca pelo transporte escolar com base no bairro e na escola pesquisados. Projetado e desenvolvido pela Urbes (empresa pública

vinculada à prefeitura de Sorocaba), o aplicativo atende exclusivamente à cidade de Sorocaba [Urbes escolar 2024].

O VanApp se destaca do **De Van para Escola** ao diminuir a barreira de entrada do usuário, pois não é necessário realizar um cadastro para utilizar as funcionalidades do *website*. Destaca-se também pelo cuidado na elaboração das funcionalidades, de modo que não existam recursos que possam impactar negativamente o condutor, como o rastreamento em tempo real e o sistema de avaliação, visando trazer maior adesão ao sistema por parte dos condutores.

O diferencial em relação ao **Urbes Escolar** está no público-alvo da cidade, uma vez que o Urbes Escolar atende exclusivamente à região de Sorocaba, e também nas funcionalidades de controle de vagas e cadastro de linhas em diferentes períodos, pois, no **Urbes Escolar**, não existe essa diferenciação de linhas.

A fim de facilitar a visualização das propostas dos trabalhos correlatos e o que este trabalho propõe, segue na Tabela 2 um comparativo das principais características de cada sistema:

Tabela 2. Comparativo entre trabalhos correlatos e o VanApp

Funcionalidades	De van para escola	Urbes escolar	VanApp
Cadastro de condutores	Sim	Sim	Sim
Cadastro de usuários	Sim	Não	Não
Pesquisa de transporte	Sim	Sim	Sim
Controle de vagas disponíveis	Sim	Não	Sim
Suporte para *mobile*	Sim	Sim	Sim*
Suporte para *web*	Sim	Não	Sim
Atende Hortolândia - SP	Sim	Não	Sim

*Sim, *website* responsivo.

3. Metodologia

As metodologias aplicadas no desenvolvimento do VanApp foram detalhadas nas subseções 3.1, sobre a metodologia de pesquisa, e 3.2, sobre a metodologia de desenvolvimento.

3.1. Metodologia de Pesquisa

O *Design Science Research* (DSR) trata-se de um modelo de pesquisa no qual se identifica um problema da comunidade com o qual o pesquisador tenha contato. A partir disso, propõe-se a criação de um artefato para a correção do problema e, após o desenvolvimento do artefato, realiza-se uma pesquisa de campo. Passando por alguns passos da metodologia, até ser possível aplicá-la e consultar seus resultados para publicação, essa divisão entre ”pesquisa e desenvolvimento da ideia”e ”aplicação prática e coleta de resultados”é chamada de ciclo interior e exterior. A Figura 1 apresenta as etapas e interações de cada etapa da pesquisa, adaptada de [Baskerville 2018].

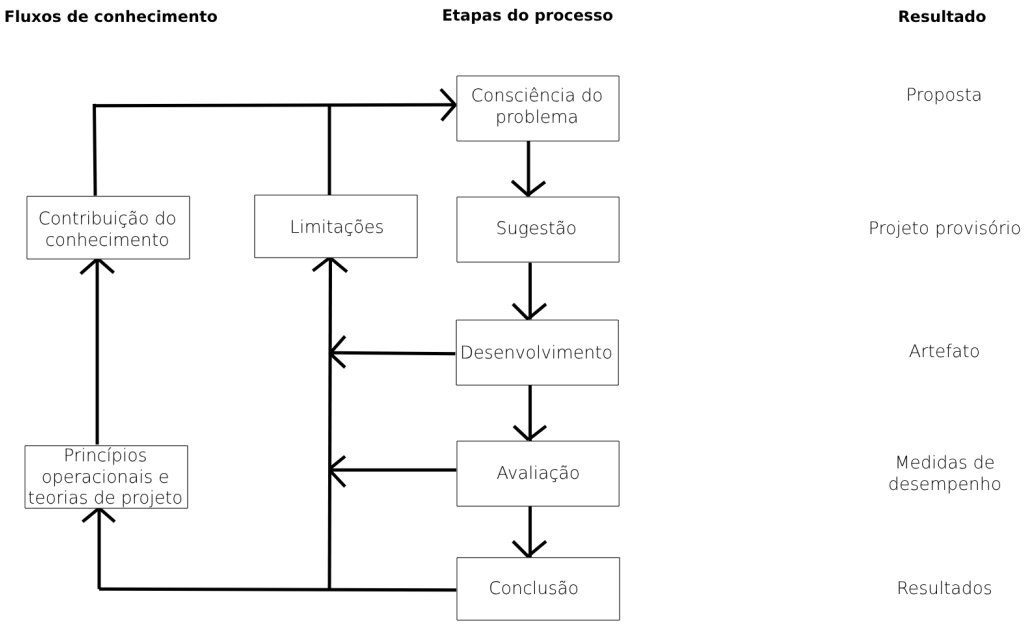

Figura 1. Diagrama *Design Science Research (Adaptado)*

Dado o limite de horas associado a esta pesquisa, na aplicação do DSR foi priorizada a finalização do ciclo interior, representado pelas etapas de conscientização do problema, sugestão e desenvolvimento, deixando a aplicação prática, ou ciclo exterior, como trabalhos futuros, representados pelas etapas de avaliação e conclusão. A etapa de conscientização do problema ocorreu na vivência do pesquisador, uma vez que ele possui contato com as duas pontas deste negócio: tanto os condutores quanto os estudantes. Por meio dessa vivência, foi realizado o levantamento de requisitos que originou a proposta da solução, uma plataforma de busca que concentrasse todos os condutores e suas informações em um único *website*.

Após essa etapa, deu-se início à etapa de sugestão, onde foi iniciada a elaboração dos diagramas de como os requisitos iriam funcionar, e o desenvolvimento começou a partir do front-end para facilitar a visualização do produto e suas funcionalidades definidas nos diagramas iniciais. Com uma visão mais clara da interação do usuário com o *website* e de como as funcionalidades iriam se comportar, iniciou-se o desenvolvimento do artefato (produto desta pesquisa), o qual será abordado mais a fundo na seção 4.

3.2. Metodologia de desenvolvimento de *software*

O Scrum é uma metodologia ágil bem conhecida para dar andamento a projetos, sendo muito útil em cenários dinâmicos e com alterações frequentes, como na criação de *softwares*. O Scrum, na prática, é um processo iterativo e incremental, que se baseia em ciclos chamados *sprints*. A cada *sprint*, você tem um resultado parcial para incrementar ao que já estava pronto anteriormente, tornando a entrega de resultados frequente e o desenvolvimento adaptável [Sutherland 2014].

Uma das características fortes do Scrum é o acompanhamento realizado entre os diferentes membros das equipes, mas isso não foi contemplado neste trabalho, pois foi

realizado sem uma equipe. Portanto, as reuniões de definição de prioridades e a reunião diária não foram utilizadas. A retrospectiva poderia ter sido realizada junto ao orientador, no entanto, devido à produtividade, esse acompanhamento foi feito por e-mail. A Figura 2 representa visualmente as etapas do Scrum, já desconsiderando as reuniões de acompanhamento e os atores que não foram incluídos nesta pesquisa. Adaptada de [Sutherland 2014].

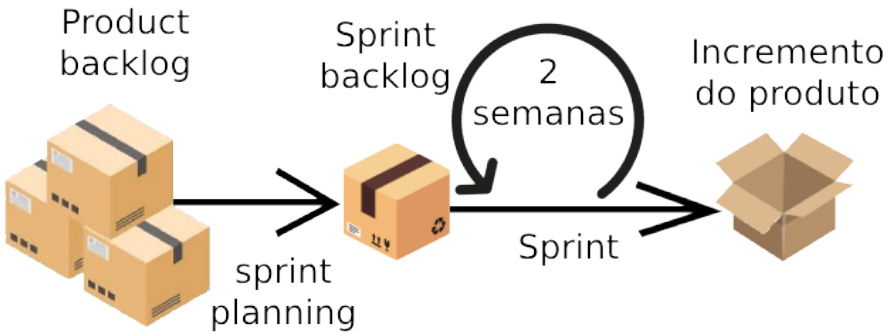

Figura 2. Scrum (Adaptado)

Seguindo esse padrão, o desenvolvimento foi dividido em 24 *sprints*, conforme o diagrama anterior, separadas em duas semanas. O tempo de cada *sprint* foi definido de acordo com a carga horária semanal disponível do pesquisador para contribuição com a pesquisa. Devido à pouca disponibilidade de tempo semanal, foi definido um *sprint* maior para o desenvolvimento de alterações consideráveis entre cada *sprint*.

4. Desenvolvimento da solução

Esta seção foi divida em oito partes: Requisitos, Diagrama de Casos de Uso, Banco de dados, Arquitetura do sistema, Estrutura de diretórios, CrudRepository e DAO, Endpoints e controller, Criptografia de senha.

4.1. Requisitos

A seguir, apresenta-se a tabela com as informações de requisitos funcionais e não funcionais definidas para o VanApp.

Tabela 3. Requisitos funcionais e não funcionais

Requisitos funcionais	Requisitos não funcionais
Pesquisa de condutores	*Back end* desenvolvido em JAVA
Filtros de condutores	*Front end* desenvolvido em HTML,CSS e *JavaScript*
Gerenciamento de condutores	Integração com banco de dados SQL
Gerenciamento de linhas	Integração com API *REST*
Requisição de cadastro	Criptografia das informações de login do condutor
Validação de cadastro	Responsividade em diferentes tamanhos de telas e aparelhos

4.2. Diagrama de Casos de Uso

Após a definição dos requisitos e elaboração da ideia, o resultado foi o casos de uso da figura 3, onde existem três atores: o estudante, o condutor e o administrador, cada um com suas funções descritas abaixo:

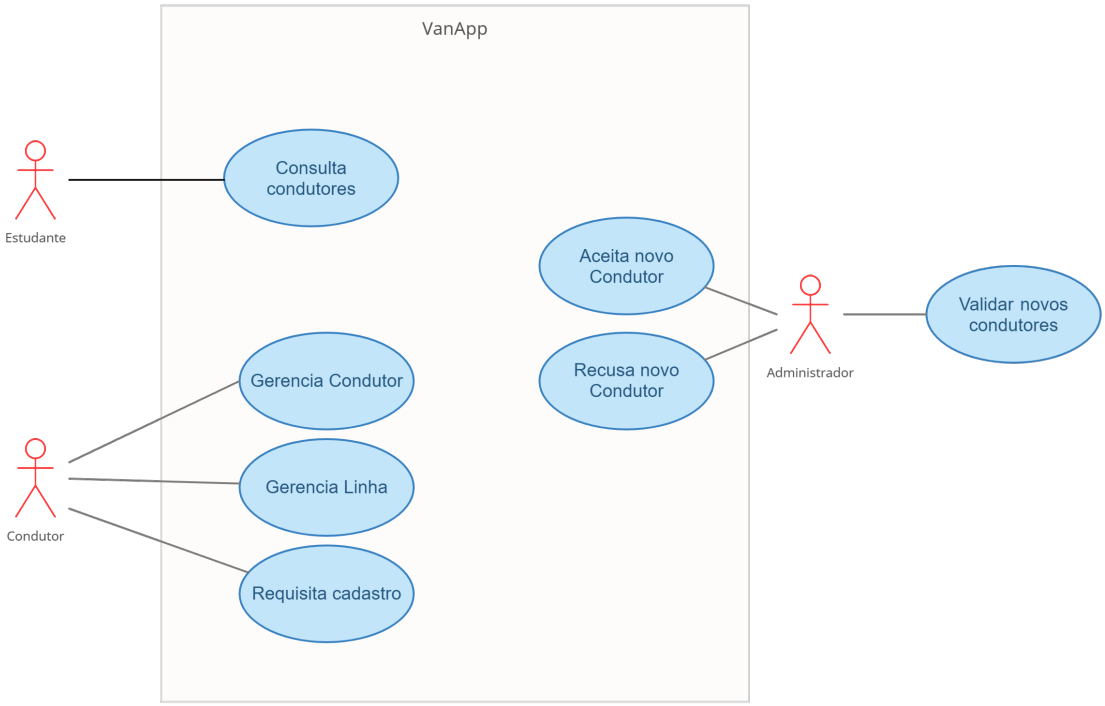

Figura 3. Diagrama de Casos de Uso

- **Cliente:** As funcionalidades relacionadas ao cliente estão voltadas para buscas, ou seja, a implementação de pesquisa e definição de filtros para localizar um condutor que atenda às necessidades.
- **Condutor:** As funcionalidades relacionadas ao condutor estão basicamente voltadas para o seu perfil, permitindo o cadastro e o gerenciamento de suas informações na plataforma.
- **Administrador:** O administrador é o terceiro ator, responsável por realizar a validação dos documentos necessários para a autenticação do condutor junto aos órgãos reguladores (todas de forma manual) e, por fim, aceitar ou recusar o cadastro do condutor.

4.3. Arquitetura do sistema

A Figura 4 apresenta um diagrama da arquitetura do sistema, onde as tecnologias envolvidas estão destacadas na seção 2.3. Assim sendo, ao lado direito é possível verificar a camada do *Front End* junto das tecnologias envolvidas; à esquerda, o Banco de dados; e, no meio, a estrutura de comunicação entre as camadas. Vale destacar também que o *Back End* é representado por toda a área indicada, junto de suas tecnologias.

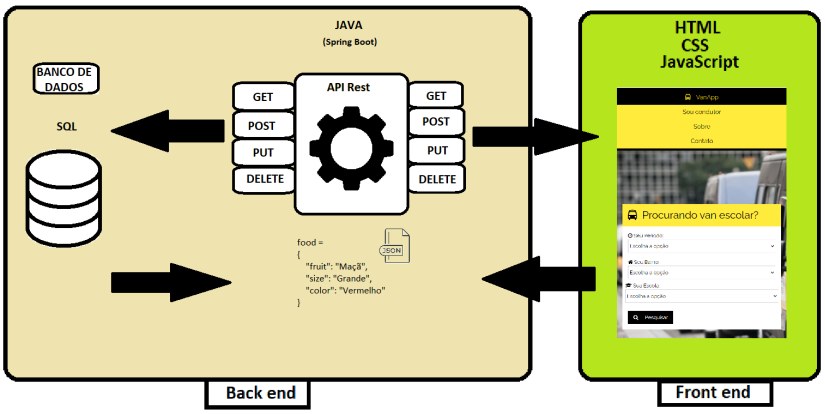

Figura 4. Estrutura técnica VanApp

4.4. Estrutura de diretórios

O resultado da estrutura de diretórios do VanApp se encontra no Apêndice A.

4.5. Banco de dados

A seguir, será apresentada a estrutura do banco de dados por meio do modelo conceitual (MER) na Figura 5.

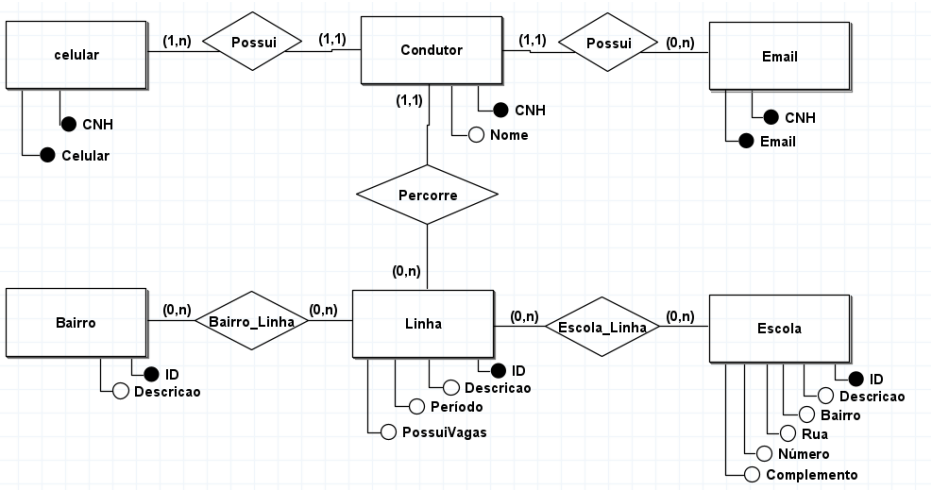

Figura 5. Modelo conceitual (MER)

Destaca-se no modelo a tabela Linha, pois possui relacionamentos N:N, que geraram tabelas auxiliares para o relacionamento na normalização. E Também a tabela Condutor, uma tabela central que liga as informações de perfil às linhas cadastradas. Como mencionado anteriormente, para interligar as tabelas do banco ao *back end* JAVA, foram utilizadas anotações *Hibernate*, que são responsáveis por mapear tabelas do banco em objetos JAVA. Um exemplo da estrutura pode ser encontrado no Objeto Linha.java e sua derivação, apresentadas na Figura 6.

Figura 6. Códigos das classes Linha e EscolaLinha no *back end*.

As principais anotações são a de *Entity* e *Table*, sendo a primeira para indicar que aquela classe se trata de um objeto que representará um elemento do banco, e a segunda para referenciar a tabela que será convertida como objeto. Para a definição das colunas da tabela, são utilizadas as anotações *Column*, onde, além do nome da coluna indicado, é possível definir parâmetros da coluna, como a quantidade máxima de caracteres, se aceita valor nulo, etc.

Por fim, temos algumas anotações especiais, como a anotação *Id*, responsável por indicar a chave primária da entidade mapeada, destacando a entidade BairroLinha, que possui uma chave composta. Portanto, foi necessário criar mais uma classe mapeando os campos da chave primária, indicada pela notação *IdClass*. A última anotação a ser destacada é a *GeneratedValue*, que gera um auto-incremento no campo mencionado, não sendo necessário preencher o campo manualmente ao inserir uma nova entidade.

4.6. *CrudRepository* e DAO

O padrão adotado para realizar a comunicação com o banco foi o padrão DAO, responsável por fornecer uma interface que abstrai o acesso ao banco de dados, encapsulando os dados de modo que as classes que utilizarem essa interface não possuam visão de como esse acesso ocorre. Com base nessa escolha, foi utilizada uma ferramenta do *Spring Boot* chamada *CrudRepository*, que é responsável por fornecer métodos já pré-configurados de acesso ao banco, apenas com a informação da entidade em que foi mapeada a tabela e do tipo da chave primária da entidade, como é possível observar na Figura 7.

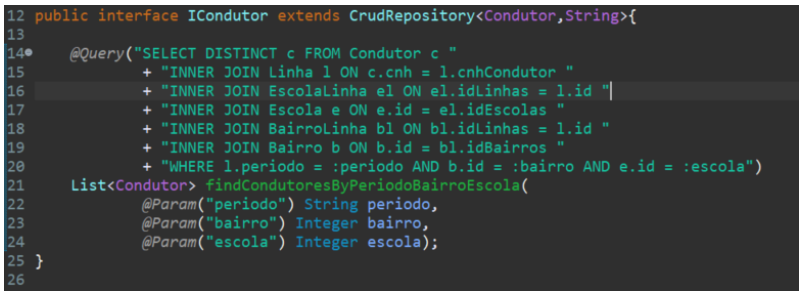

Figura 7. Código da Interface ICondutor.

Os métodos já pré-configurados por meio dessa ferramenta estão descritos a seguir: *save*(), *saveAll*(), *findById*(), *existsById*(), *findAll*(), *count*(), *deleteById*(), *delete*(), *deleteAllById*(), *deleteAll*(). Além dos métodos pré-definidos, é possível criar métodos personalizados utilizando combinações de nomes de funções com regras já pré-definidas. Essas regras podem ser encontradas na documentação do *Spring Boot*. Também é possível redefinir a *query* a ser executada por meio da anotação @*Query*, como demonstrado na Figura 7.

4.7. *Endpoints* e *controller*

Para construir as APIs, foram utilizadas novamente ferramentas fornecidas pelo *Spring Boot*, que facilitam o desenvolvimento de APIs *REST* por meio de anotações no código Java. Essas anotações constroem de maneira simplificada os *endpoints* para as requisições HTTP. Esses *endpoints* são acessados pelo *front end* por meio de *JavaScript*. Os *endpoints* foram configurados no *controller* de cada objeto, visando a comunicação entre o cliente (*front end*) e o servidor (*back end*), conforme a Figura 8, onde é exemplificada a construção da classe Controller para o objeto condutor.

```java
@RestController
@CrossOrigin("*")
@RequestMapping("/condutores")
public class CondutorController {

    @Autowired
    private ICondutor dao;
    private PasswordEncoder passwoedEncoder;

    public CondutorController(ICondutor dao) {
        this.dao = dao;
        this.passwoedEncoder = new BCryptPasswordEncoder();
    }

    @PostMapping("/uploadImage")
    public ResponseEntity<String> uploadImage(@RequestParam("image") MultipartFile image) {
        if (image.isEmpty()) {
            return ResponseEntity.badRequest().body("Nenhuma imagem recebida.");
        }

        try {
            //String diretorioRelativo = ".\\img\\users-image\\";
            //Path path = Paths.get(diretorioRelativo + image.getOriginalFilename());
            Path path = Paths.get("C:\\xampp\\htdocs\\vanApp\\img\\users-image\\"+image.getOriginalFilename());
            Files.copy(image.getInputStream(), path, StandardCopyOption.REPLACE_EXISTING);
            return ResponseEntity.ok("Imagem salva com sucesso!");
        } catch (IOException e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Erro ao salvar a imagem.");
        }
    }

    @PostMapping
    public Condutor cadastrarCondutor(@RequestBody Condutor condutor) {
        String encoder =this.passwoedEncoder.encode(condutor.getSenha());
        condutor.setSenha(encoder);
        Condutor condutorNova = dao.save(condutor);
        return condutorNova;
    }
```

Figura 8. *Controller*

As requisições HTTP utilizadas no desenvolvimento do VanApp são:

- ***GET*:** Por meio da anotação @*GetMapping* é possível obter dados do servidor para consumo no cliente.
- ***POST*:** A anotação @*PostMapping* é responsável por receber novos dados do cliente para cadastro no servidor.

- ***PUT*:** A anotação *@PutMapping* é responsável por receber dados do cliente para alteração no servidor de dados já existentes.
- ***DELETE*:** A anotação *@DeleteMapping* é responsável por receber dados do cliente para deleção no servidor de dados já existentes.

4.8. Criptografia de senha

Para a realização da criptografia de senha dos usuários, foi utilizada uma ferramenta de criptografia, também disponibilizada pelo *framework Spring Boot*, chamada *Password Encoder*. Essa ferramenta permite armazenar as senhas de forma segura no momento da criação, para que possam ser posteriormente comparadas durante a autenticação no login, de acordo com o valor inserido pelo usuário. A senha é definida no momento do cadastro do usuário e verificada com o valor fornecido durante o login. No banco de dados, a senha permanece criptografada, conforme ilustrado nas Figuras 9 e 10.

```
62      @PostMapping
63      public Condutor cadastrarCondutor(@RequestBody Condutor condutor) {
64          String encoder =this.passwoedEncoder.encode(condutor.getSenha());
65          condutor.setSenha(encoder);
66          Condutor condutorNova = dao.save(condutor);
67          return condutorNova;
68      }
69
70      @PostMapping("/login")
71      public Condutor validarSenha(@RequestBody Condutor condutor){
72          String senha = dao.findById(condutor.getCnh()).get().getSenha();
73          Boolean valid = passwoedEncoder.matches(condutor.getSenha(), senha);
74          if (valid) {
75              return condutor;
76          }else {
77              return null;
78          }
79
80      }
```

Figura 9. Criptografia

	CNH	Name	imgPath	senha
Editar Copiar Remover	1111111111	Lucas	sprinter.jpg	$2a$10$pzMgMggEAPUHMBxiBndguuD95UTvm91XkE5wR2esfHd...

Figura 10. Criptografia no Banco

5. Resultados

O VanApp foi desenvolvido e atendeu tanto ao objetivo principal quanto aos objetivos específicos. Durante a pesquisa e o desenvolvimento, surgiram novos questionamentos sobre as funcionalidades, com ênfase na usabilidade do sistema. Dessa forma, algumas funcionalidades desenvolvidas poderão passar por alterações visuais ou operacionais no futuro. Contudo, o desenvolvimento realizado atendeu a todas as propostas para os atores cliente e condutor, conforme mencionado na Figura 3, resultando nas seguintes funcionalidades: pesquisa e filtro de condutores, gerenciamento de condutores e gerenciamento de linhas, que serão detalhados no próximo tópico.

5.1. Software desenvolvido

Esta subseção apresenta a solução sobre as perspectivas dos condutores e dos estudantes.

5.1.1. Perspectiva do condutor

Ao selecionar a aba 'Sou Condutor' na barra de navegação, o condutor tem acesso ao *container* de cadastro e login. Conforme ilustrado na Figura 11, para se cadastrar, o usuário precisa inserir as informações de CNH, nome, até dois números de celular, até dois e-mails, além da senha e sua confirmação. Após preencher essas informações, ao clicar em 'Próximo', uma nova aba é aberta para o *upload* da foto de perfil. Uma vez que a foto seja carregada, basta selecionar o botão 'Próximo' novamente e, caso não haja informações obrigatórias em branco, o cadastro será finalizado.

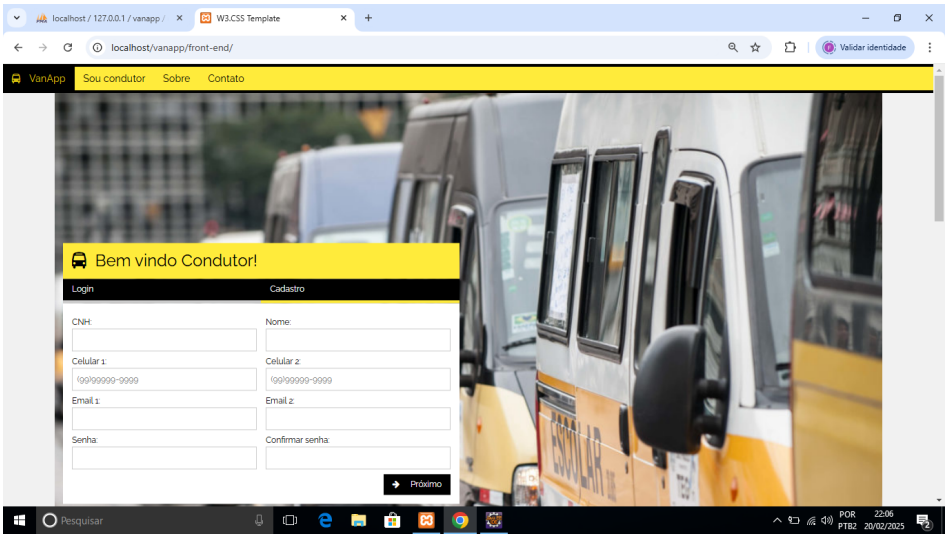

Figura 11. Cadastro - condutor

Como demonstrado na Figura 12, para que o usuário tenha acesso ao seu perfil, é necessário informar a CNH e a senha previamente registrada e pressionar o botão 'Entrar'. Caso as informações estejam corretas, o usuário é direcionado para a página do perfil, caso contrário, recebe uma mensagem informando que a CNH ou a senha estão incorretas.

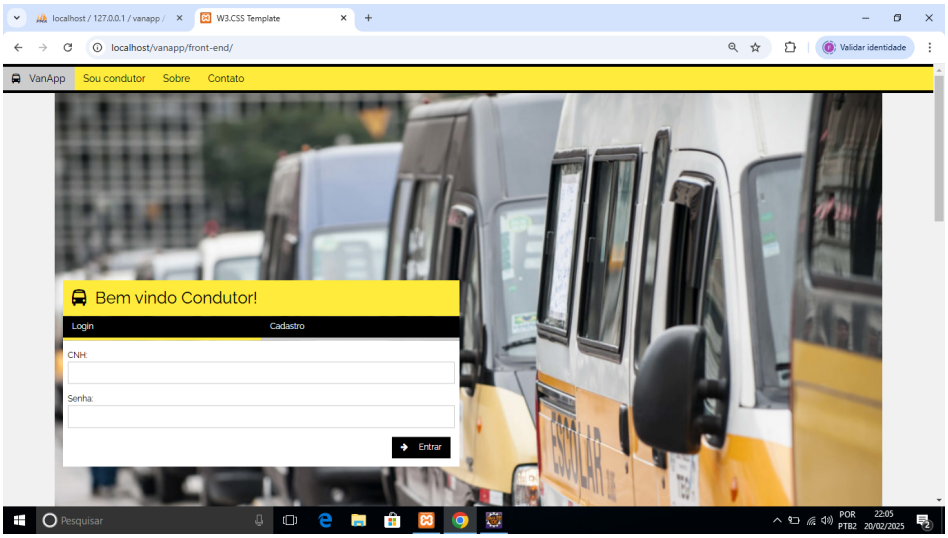

Figura 12. Login - condutor

Uma vez acessado seu perfil, o condutor pode alterar seus dados de contato e marcar se há ou não vagas em determinado período, além de ter acesso às ferramentas de edição de linha, conforme mostrado na Figura 13.

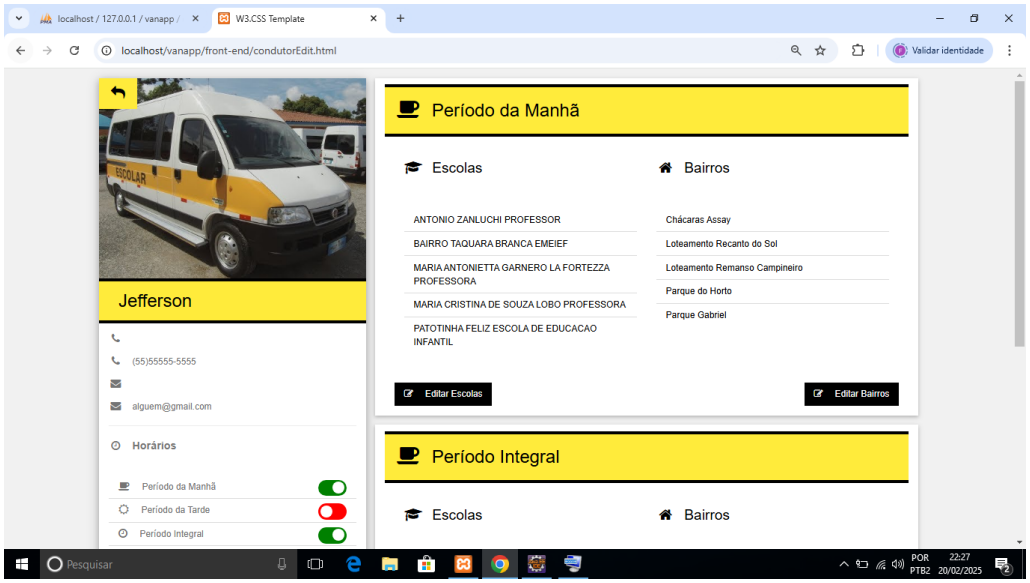

Figura 13. Perfil - condutor

Selecionando os botões ’editar escolas’ ou ’editar bairros’, o condutor tem acesso a duas listas: uma com todas as escolas ou bairros cadastrados no sistema (exceto os que já estão cadastrados em sua linha) e outra com as escolas ou bairros que já fazem parte de sua linha. O condutor pode mover os elementos de uma lista para a outra livremente. Após selecionar ’salvar’, todas as escolas ou bairros selecionados são salvos, e os que foram removidos são excluídos da linha, conforme o exemplo com as escolas na Figura 14.

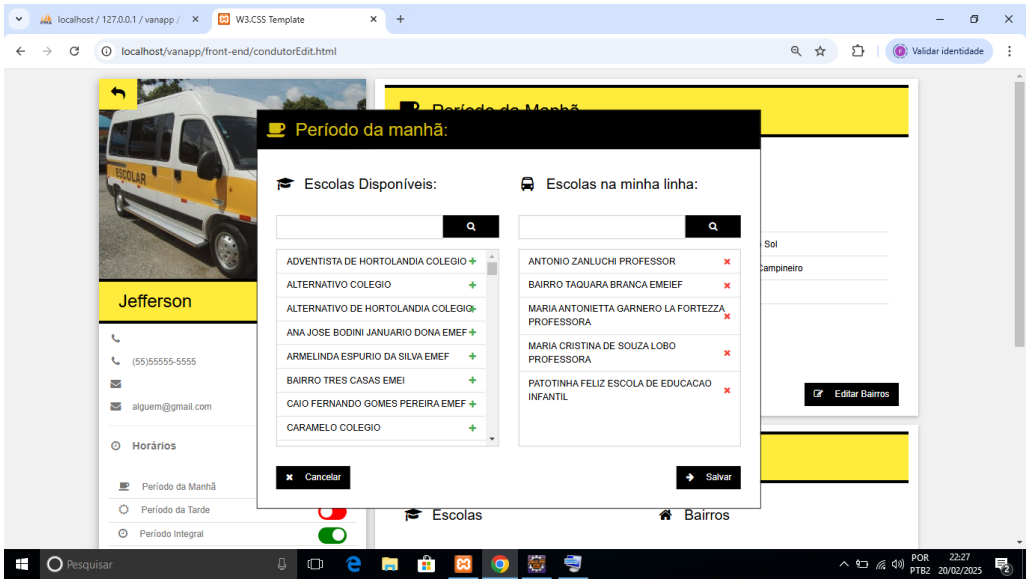

Figura 14. Edição de linhas - condutor

5.1.2. Perspectiva do estudante

Ao abrir o VanApp, o usuário se depara com a tela apresentada na Figura 15, que é a tela principal do sistema, onde aparece o container de busca, conforme mostrado na imagem.

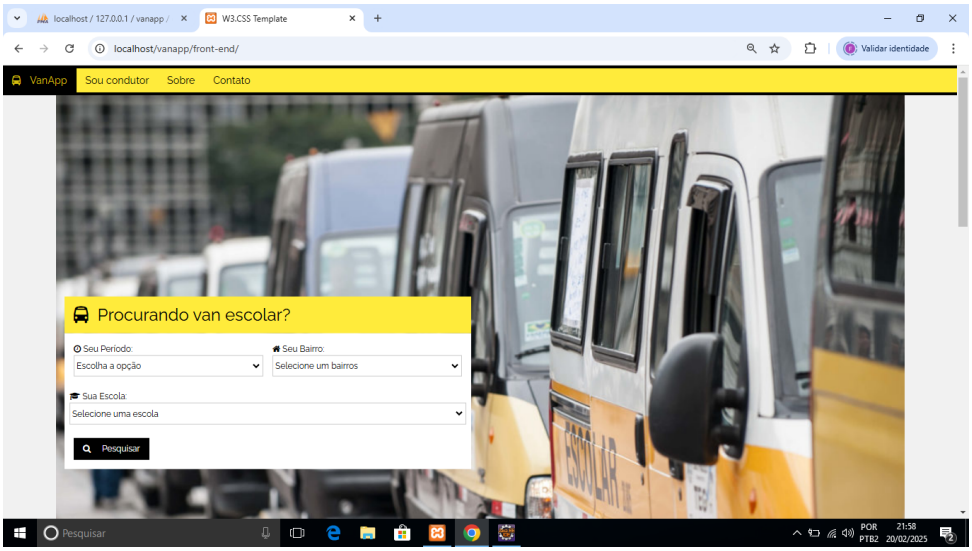

Figura 15. Iníco - estudante

No *container*, o usuário fornece as informações sobre o período de aula, escola e bairro do estudante. Ao selecionar o botão 'pesquisar', o software realiza uma busca por todas as linhas cadastradas que atendam à necessidade do usuário e que tenham vagas disponíveis, retornando uma lista com os condutores responsáveis por essas linhas, incluindo informações de contato, conforme a Figura 16.

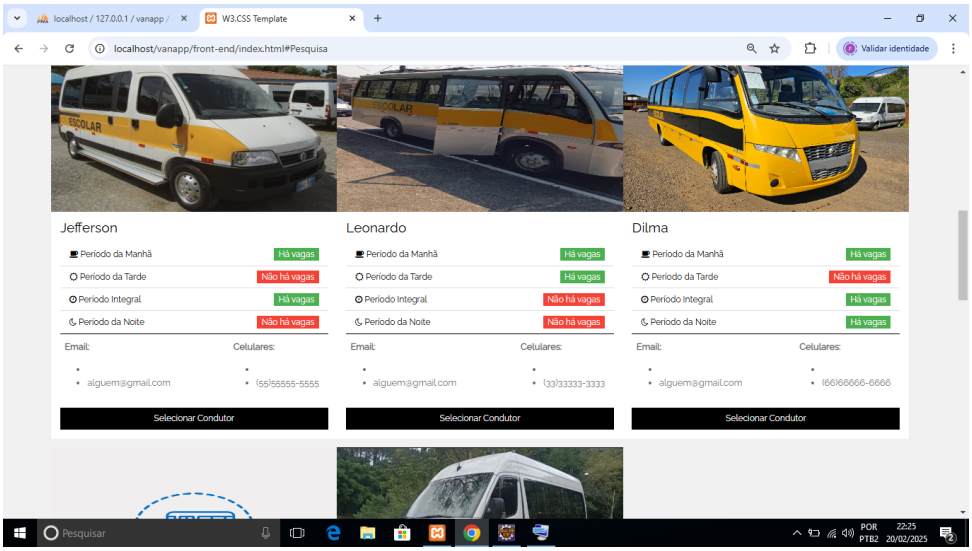

Figura 16. Resultado - estudante

Selecionando um condutor, o usuário pode visualizar o perfil completo do condutor, com todos os seus itinerários e informações de perfil, conforme a Figura 17.

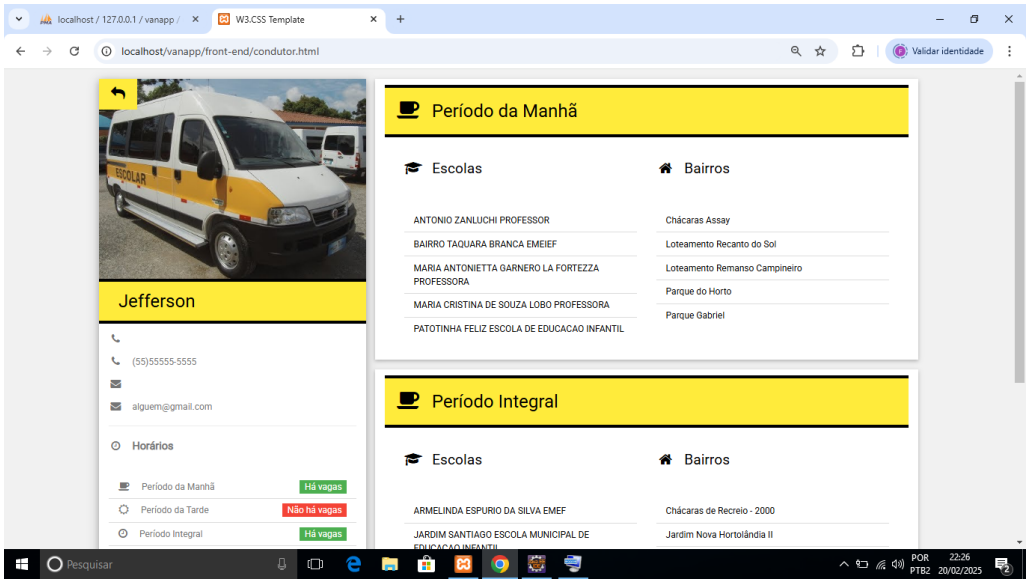

Figura 17. Perfil - estudante

5.2. Código desenvolvido

O código-fonte do VanApp pode ser encontrado no *GitHub* por meio do link: https://github.com/LVP-Vilela/VanApp.

6. Considerações finais

Nesta pesquisa, foi desenvolvido o sistema VanApp, cujo objetivo foi o desenvolvimento de uma plataforma de busca de transporte escolar na região de Hortolândia -SP da forma mais simples possível para o estudante, auxiliando os estudantes a encontrar o condutor que melhor atenda às suas necessidades. O sistema também permite que os condutores compartilhem seus serviços sem custos adicionais, digitalizando a divulgação desse tipo de serviço, que até então era feito de maneira predominantemente física ou por indicação pessoal.

O método de pesquisa utilizado foi o *Design Science Research*. Para o desenvolvimento do sistema, foi adotada a metodologia Scrum, adaptada para manter o ciclo de desenvolvimento, mas sem as reuniões, por se tratar de um trabalho realizado por apenas um pesquisador com acompanhamento do orientador. O desenvolvimento do *software* foi dividido entre *front end* e *back end*, com destaque para a comunicação via API *REST* entre as partes e a utilização dos padrões MVC e DAO no *back end*. O sistema foi projetado para ser acessível em computadores, tablets e celulares por meio de uma interface *web*, sendo responsivo às limitações de tamanho de cada dispositivo. A pesquisa atendeu tanto ao objetivo geral quanto aos objetivos específicos detalhados anteriormente.

O primeiro desafio encontrado durante o desenvolvimento começou na escolha dos requisitos, pois foi necessário filtrar as possibilidades de modo a resolver o problema de busca de forma objetiva e intuitiva, sem extrapolar a liberdade do condutor, uma vez que o sucesso do *software* depende da adesão deles à plataforma. Outro desafio ocorreu durante o desenvolvimento da arquitetura do *software*, uma vez que a arquitetura não se tratava de um modelo familiar para o pesquisador, além da elaboração de *designs* voltados

à usabilidade, devido à grande quantidade de opções a serem selecionadas em uma única tela, como ocorre no gerenciamento de linhas, apresentado na Figura 14.

Entre as disciplinas do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas utilizadas nesta pesquisa, destacam-se: Engenharia de *Software*, Análise Orientada a Objetos, Banco de Dados I e II, Linguagem de Programação I e II, Arquitetura de *Software*, Programação Orientada a Objetos, Metodologia de Pesquisa Científica e Tecnológica, Desenvolvimento *Web* e Desenvolvimento de Sistemas *Web* e Projeto de Sistemas I e II.

Quanto aos trabalhos futuros, as sugestões para os próximos passos, com o intuito de completar a metodologia DSR, tanto no ciclo interior quanto no ciclo exterior, incluem a realização de testes de usuário para verificar a usabilidade da plataforma, a implementação da validação do cadastro de condutores junto aos órgãos de trânsito e órgãos municipais, por meio de integração entre sistemas ou por intermédio de profissional responsável (o ator ”administrador”na Figura 3), a realização de pesquisa de campo antes de disponibilizar o VanApp à comunidade, a implantação do software para disponibilização à comunidade, a realização de pesquisa de campo após adesão dos condutores e período de matrículas escolares, para comparação com a pesquisa anterior e coleta de resultados e por fim desenvolver recursos de acessibilidade.

Referências

AWS (2023a). Front-end x back-end — Diferença entre desenvolvimento de aplicativos — AWS. https://aws.amazon.com/pt/compare/the-difference-between-frontend-and-backend/: :text=O

AWS (2023b). O que é Java? – Explicação sobre a linguagem de programação Java – AWS. https://aws.amazon.com/pt/what-is/java/. Acesso em 18 de setembro de 2024.

Baskerville, R. (2018). Design science research contributions finding a balance between artifact and theory. *Journal of the Association for Information Systems*, pages 359–376.

DataSUS (2019). SISTEMA GERENCIADOR DE BANCO DE DADOS (SGBD). https://datasus.saude.gov.br/glossario/sistema-gerenciador-de-banco-de-dados-sgbd/: :text=Em

De Van Para Escola (2024). De Van Para Escola. https://www.devanpraescola.com.br/. Acesso em 23 de julho de 2024.

GitHub (2023). The top programming languages — The State of the Octoverse. https://octoverse.github.com/2022/top-programming-languages. Acesso em 18 de setembro de 2024.

Hibernate (2025). Your relational data. Objectively. https://hibernate.org/orm/. Acesso em 23 de fevereiro de 2025.

Jacobson, I. (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley.

MariaDB (2011). Sobre o Mariadb. https://mariadb.com/kb/pt-br/sobre-o-mariadb/. Acesso em 23 de fevereiro de 2025.

Mozilla (2024a). Introduction to HTML. https://developer.mozilla.org/pt-BR/docs/Learn/HTML/Introduction_to_HTML/Getting_started. Acesso em 23 de julho de 2024.

Mozilla (2024b). CSS. https://developer.mozilla.org/pt-BR/docs/Web/CSS. Acesso em 23 de julho de 2024.

Mozilla (2024c). JavaScript. https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em 23 de julho de 2024.

Oracle (2002). Core J2EE Patterns - Data Access Object. https://www.oracle.com/java/technologies/dataaccessobject.html. Acesso em 23 de fevereiro de 2025.

Oracle (2007). Java SE Application Design With MVC. https://www.oracle.com/technical-resources/articles/java/java-se-app-design-with-mvc.html. Acesso em 23 de fevereiro de 2025.

Oracle (2024). O que é um Banco de Dados? — Oracle Brasil. https://www.oracle.com/br/database/what-is-database/. Acesso em 23 de setembro de 2024.

QEdu (2023). QEdu. https://qedu.org.br/municipio/3519071-hortolandia/censo-escolar. Acesso em 31 de julho de 2024.

RedHat (2024). O que é uma API REST (Representational State Transfer)? https://www.redhat.com/pt-br/topics/api/what-is-a-rest-apio-que-significa-rest. Acesso em 23 de setembro de 2024.

Rádio Guarujá (2025). Após pai relatar dificuldade em encontrar transporte para filho que estuda no IFSC em Criciúma, secretaria de Orleans explica limitações. https://www.ligadonosul.com.br/educacao/apos-pai-relatar-dificuldade-em-encontrar-transporte-para-filho-que-estuda-no-ifsc-em-criciuma-secretaria-de-orleans-explica-limitacoes/. Acesso em 07 de março de 2025.

Spring (2024). Spring — Why Spring. https://spring.io/why-spring. Acesso em 18 de setembro de 2024.

Sutherland, J. (2014). *Scrum: A Arte de Fazer o Dobro de Trabalho na Metade do Tempo*. Crown Business.

Urbes escolar (2024). Urbes escolar. https://agencia.sorocaba.sp.gov.br/aplicativo-da-urbes-auxilia-busca-por-vans-de-transporte-de-alunos/. Acesso em 23 de julho de 2024.

W3Schools (2024a). SQL Introduction. https://www.w3schools.com/sql/sql_intro.asp. Acesso em 18 de setembro de 2024.

W3Schools (2024b). W3.CSS. https://www.w3schools.com/w3css/default.asp. Acesso em 23 de julho de 2024.

W3Schools (2024c). JSON Introduction. https://www.w3schools.com/js/js_json_intro.asp. Acesso em 18 de setembro de 2024.

Wiegers, K. E. (2013). *Engenharia de Requisitos: Um Guia Prático para o Desenvolvimento de Software.* Bookman.

A. APÊNDICE

ESTRUTURA DE DIRETÓRIOS

Autor: Lucas Vilela Piza

Este documento aborda a versão final da estrutura de diretórios do *front end* e do *back end*.

No desenvolvimento do *front end*, tivemos três páginas principais e mais duas pastas contendo os respectivos arquivos CSS e *JavaScript*, incluindo os códigos CSS do framework W3.css, conforme a Figura 18.

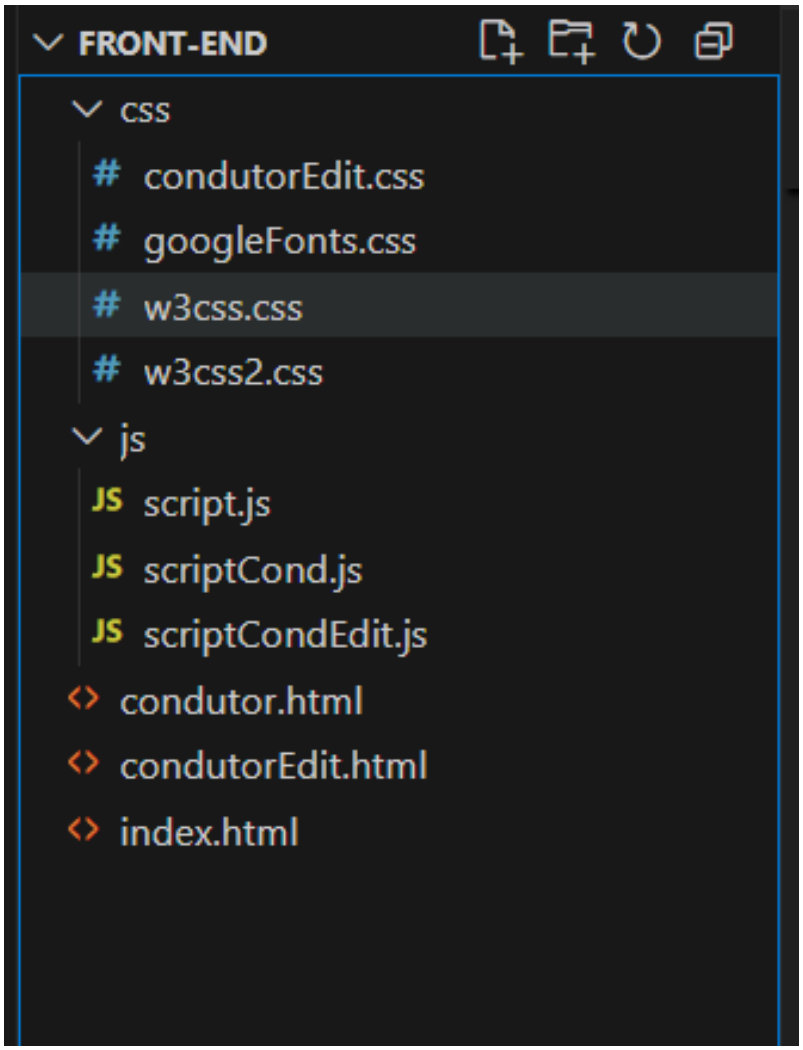

Figura 18. Organização do código no *Front End*

No desenvolvimento do *back end*, as classes foram organizadas em pacotes de acordo com sua função dentro dos padrões adotados, sendo o pacote *controller* responsável pela comunicação do *back end* com o *front end*, o pacote DAO que possui as classes que fazem a comunicação com o banco de dados, e o pacote *model*, responsável pelos objetos utilizados nas demais classes e pela lógica do código *back end*, conforme a Figura 19.

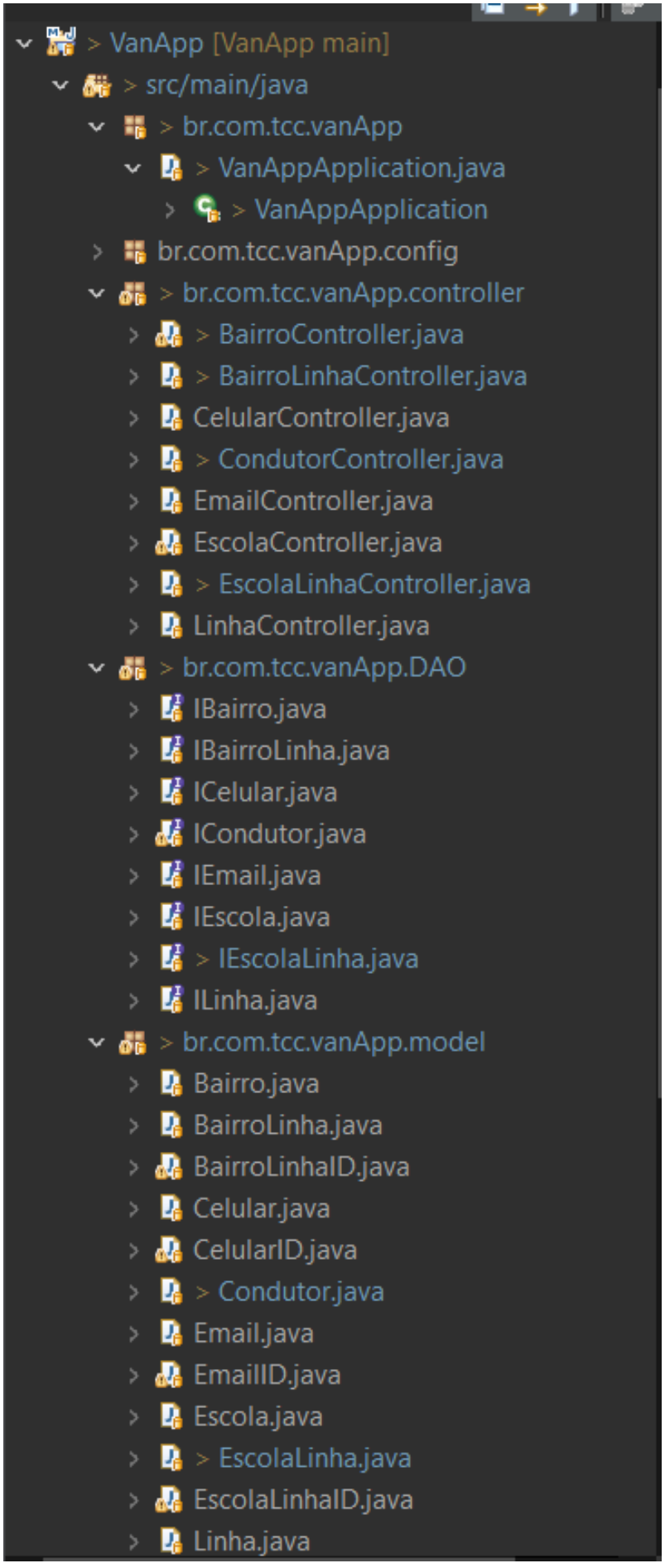

Figura 19. Organização do código no *Back End*

Documento Digitalizado Público

Trabalho de conclusão de curso (TCC) de Lucas Vilela Piza: versão final (após Banca de Defesa)

Assunto: Trabalho de conclusão de curso (TCC) de Lucas Vilela Piza: versão final (após Banca de Defesa)
Assinado por: Alexandre Grotta
Tipo do Documento: Relatório
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Alexandre Grotta**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 13/03/2025 21:27:26.

Este documento foi armazenado no SUAP em 13/03/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1966080
Código de Autenticação: 45067f1e8e