RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 12 Nro. 2 2022
Uni-FACEF Centro Universitário Municipal de Franca

SISTEMA DE GESTÃO DE ESTACIONAMENTO ROTATIVO

Donato Cardoso Ávila
Graduado em Sistemas de Informação – Uni-FACEF
donatocardoso11@gmail.com

Manyfe Tereza Duarte
Graduado em Sistemas de Informação – Uni-FACEF
manyfe@hotmail.com

Débora Pelicano Diniz
Mestre em Computação - Uni-FACEF
deboradiniz@facef.br

RESUMO

As cidades, atualmente, vêm apresentando problemas com a mobilidade urbana, uma vez que o crescimento descontrolado, e a quantidade de veículos individuais transitando nas vias, aliado à diminuição da demanda de transporte público urbano, acarreta em problemas para as pessoas. Esta pesquisa em questão, visa propor uma solução tecnológica para um problema na mobilidade urbana, especificamente para estacionamentos rotativos. Para o desenvolvimento do projeto foram utilizadas ferramentas, métodos e boas práticas propostas pela Engenharia de Software tais como Diagrama de Caso de Uso, Matriz de Rastreabilidade, Diagrama de Classe e o Modelo Entidade Relacionamento. Foram realizados os processos do Ciclo de Vida do Desenvolvimento de Software, nas etapas de levantamento, documentação, análise de requisitos, a realização do projeto, a prototipação de banco de dados, de interfaces e por fim a implementação de código e testes. Deste modo o presente projeto, resultou em um protótipo funcional, que possui facilidade de acesso e tem a capacidade de manter a gestão que garanta maior rotatividade na utilização das vagas dos estacionamentos, aumentando a eficiência do sistema e como resultado dinamizando o acesso dos clientes.

PALAVRAS-CHAVE: Agilidade, Mobilidade Urbana, Gerenciamento, Estacionamento Rotativo.

ABSTRACT:

Cities currently have problems with urban mobility, since the uncontrolled growth, and the amount of individual vehicles transiting on the roads, combined with the decrease in the demand for urban public transport, causes problems for people. This research in question aims to propose a technological solution to a problem in urban mobility, specifically for rotating parking lots. For the development of the project, tools, methods and good practices proposed by Software Engineering were used, such as Use Case Diagram, Traceability Matrix, Class Diagram and the Entity Relationship Model. The processes of the Software Development Life Cycle were carried out, in the stages of survey, documentation, requirements analysis, project

realization, database prototyping, interfaces and finally the implementation of code and tests. In this way, the present project resulted in a functional prototype, which is easy to access and has the ability to maintain management that guarantees greater turnover in the use of parking spaces, increasing the efficiency of the system and as a result streamlining customer access.

KEYWORDS: *Agility, Urban Mobility, Management, Rotation Parking*.

1. INTRODUÇÃO

Com o crescimento das cidades observa-se que um dos dilemas do dia a dia é a mobilidade urbana que nas médias e grandes cidades pode ser um grande problema devido a superlotação nos centros das cidades. Para Fiorin e Stares (2019) o transporte se trata de um importante meio de conexão entre as pessoas e o seu trabalho e lazer, ou seja, suas atividades econômicas, culturais e sociais.

É nos centros urbanos que se verificam as maiores carências de vagas para estacionar os veículos que as pessoas usam diariamente para realização de sua locomoção e onde, ao mesmo tempo, diversos fatores se combinam para diminuir as chances de se encontrar um espaço livre para tal, como, por exemplo, a grande densidade de construções e o valor mais elevado dos terrenos (FIORIN E STARES, 2019).

Observando essa necessidade surgiu um novo tipo de negócio: os Estacionamentos Rotativos, que são lugares predeterminados e destinados a paradas veiculares, podendo ser em vias públicas ou em formato privado, em ambos os casos é realizada a contraprestação financeira quando utilizado.

O projeto aqui apresentado possui como principal objetivo apresentar o processo de desenvolvimento de um sistema de controle e gestão de vagas para um estacionamento privado localizado na região central da cidade de Franca. São apresentados detalhes das especificações do projeto assim como as partes interessadas no software e as suas carências que foram sanadas pelo protótipo funcional implementado.

Os requisitos do sistema foram obtidos, documentados e analisados para então ser projetado o software, com base nos conceitos, métodos e boas práticas propostos pela Engenharia de Software, elevando a qualidade do produto final e diminuindo o retrabalho na implementação do software.

A partir da versão inicial desenvolvida, percebeu-se a possibilidade de oferecer novas funcionalidades, que foram documentadas, podendo ser objeto de estudo para uma futura evolução do aplicativo.

2. MOBILIDADE URBANA

O advento da mobilidade urbana sofreu importantes transformações desde as últimas décadas do século 20, isso de forma global, não somente no Brasil. Tais mudanças redirecionam os fluxos migratórios, que antes restringiam-se aos grandes aglomerados urbanos, para as cidades médias e, de acordo com (COSTA, 2008) o tema em questão afeta abruptamente o desenvolvimento do país e a qualidade de vida da população.

Rodrigues (2006), explica que, o serviço de infraestrutura como transporte,

nele incluído o carro, é responsável por proporcionar às pessoas acesso aos locais de trabalho, estudo, lazer e outras oportunidades de consumo, viabilizando também a integração dos espaços existentes na cidade, sendo um pré-requisito básico para evolução da economia e atua como canal de distribuição para o cliente final. Para atender as necessidades de deslocamento da população, a mobilidade urbana, tem papel fundamental para suprir as atividades cotidianas, podendo ser para fins educacionais, laborais ou simplesmente lazer. E pode ser entendida como a maneira das pessoas transitarem nos espaços urbanos, seja de maneira individual: a pé, bicicletas, motocicletas e/ou carros; ou seja, de maneira coletiva: ônibus, metrô, trem e afins.

Vale ressaltar que esse tema não diz respeito apenas a deslocamentos veiculares, mas sim de um complexo que se pode traduzir como sendo políticas de transporte, nas quais o conceito é bastante abrangente. Os padrões de mobilidade vêm, inclusive, passando por fortes modificações. Devido ao acelerado processo de urbanização e evolução das cidades, o transporte individual motorizado está cada vez mais intenso (IPEA, 2010).

Nos anos 90 começa uma nova visão do mundo; o ponto de vista contextual que vê o mundo como um sistema dinâmico e complexo, com múltiplas funções, dimensões e contradições, em que o mercado está incluído, mas não se restringe só a ele. Isso é transcendido. Portanto a mobilidade toma um significado multidimensional, ou seja, suas variáveis são múltiplas e a prática sistêmica do desenvolvimento não adquire limites universais, por se tratar de um modelo composto pelas realidades, necessidades e aspirações locais (Terán, 2013 apud Souza e Silva, 2003, pág. 74). O desenvolvimento, além de atender o vetor econômico, leva em conta o desenvolvimento ambiental e ecológico, tornando uma cidade sustentável com práticas para se atingir a mobilidade urbana desejável.

Para Campos (2006), a mobilidade urbana pode ser contemplada por meio de medidas direcionadas ao melhor aproveitamento do uso e ocupação do solo, bem como propiciar a gestão de transportes de modo a garantir a acessibilidade de bens e serviços a todos os habitantes.

Os diferentes paradigmas que orientaram a mobilidade, nas décadas citadas, mostram a superposição destes no tempo e a luta por impor seus conteúdos nos diversos discursos presentes na nossa sociedade. Segundo Kuhn, (1997, pág.117) “um novo paradigma emerge antes que uma crise esteja bem desenvolvida ou tenha sido explicitamente reconhecida”. A superposição citada mostra a mudança de época, a qual demanda novos valores, novas formas de pensar e de enfocar os mais diversos problemas urbanos, como transporte, trânsito, uso e ocupação de solo, ou outros sistemas como habitação, saúde e educação; todos interdependentes, com retroações e recursividades, que farão emergir num sistema maior, à cidade, um novo atributo: a mobilidade.

Segundo Terán (2013 apud Rangel 2007), nos anos 70 começa uma transfiguração cognitiva no estudo dos processos urbanos; passa-se do ponto de vista multi fragmentado e reducionista para uma tendência de integração em sistemas complexos, vivenciando-se assim uma realidade em constante transformação. Na visão de Castell (1996, pág. 27 e 28), essa é uma época “informacional” e chega-se a ela devido às transformações qualitativas e simultâneas, na produção econômica, nas relações de poder, na experiência humana e no âmbito cultural.

Ao longo dos anos a mobilidade urbana, ganhou evidência, gerando graves problemas. Assim como afirma Ricardo Abramovay (2011), esvaziam o pátio das

montadoras, mas contribuem para aumentar os índices dos congestionamentos.

O grande fluxo de veículos e a indisponibilidade de estacionamentos vem influenciado negativamente no comércio das cidades. Mesmo estando dentro de estacionamento privado não é fácil a procura por uma vaga disponível. O problema do estacionamento é uma questão mundial. Já existem empresas produzindo ideias e soluções para o problema. Mas, o fato é que os sistemas podem não cumprir com sua real finalidade, que é garantir rotatividade dos veículos ao encerrar o tempo máximo de permanência e ininterrupta em uma mesma vaga, portanto, descaracterizando seu propósito e reduzindo a efetiva quantidade de usuários por vaga durante a operação do sistema.

Mobilidade urbana também tem relação com outros dois conceitos fundamentais: qualidade e custo de vida. A questão tornou-se tão séria que obrigou a prefeitura do Estado de São Paulo, por exemplo, a instituir e manter o sistema de rodízio, que consiste em proibir de circular em dias úteis veículos de passeio conforme o final das suas placas. Com a expansão digital, pode-se utilizar a tecnologia também para solucionar os desafios da mobilidade urbana, assim como já é feito em vários países. A mais difundida delas é o uso de apps de compartilhamento que podem ser de carona, como o Blablacar, transporte em grupo ou individual, como Uber ou de bicicletas.

> A mobilidade urbana é algo fundamental a todas as cidades do mundo, isso porque é ela que propicia as atividades comerciais, industriais, educacionais e recreativas. Além disso, sem ela, não seria possível existir o desenvolvimento urbano. Sendo assim, o transporte urbano é tão importante para a qualidade de vida da população, quanto os serviços básicos do cidadão, como o de água, esgoto e energia elétrica (FERRAZ, 2004).

Pensar na mobilidade urbana significa entender e incorporar fatores econômicos, como a renda do indivíduo, aspectos sociais de idade e sexo, as limitações físicas para utilização dos veículos, tornando-se evidente que é necessário tratar os deslocamentos não apenas como a ação de ir e vir, mas a partir de um conceito mais complexo interpretado por mobilidade.

3. ESTACIONAMENTOS ROTATIVOS

O sistema de estacionamento rotativo pago tem como função popularizar e tornar mais acessível os estacionamentos para todos os veículos nas vias públicas. Facilita a rotatividade, possibilitando redução de aglomerados urbanos e auxilia acessos aos estabelecimentos existentes nas regiões de maior circulação. O principal objetivo do estacionamento rotativo é impedir que gere transtorno no trânsito em locais com alta movimentação e pouco espaço para para permanência de veículos, pois estacionamentos inapropriados em vias públicas são capazes de conceber perigos como acidentes, avarias e até furtos. (PARADELA et al, 2015)

O Controle de estacionamento em vias públicas, conforme MAY (1999), exerce ao mesmo tempo três tipos de restrições, sendo elas: Restrições Físicas, Restrições de Controle Normativo e Restrições Financeiras.

- ➔ Restrições Físicas: Remanejamento de espaços, regulamentação de horários e tempos de permanência em estacionamentos permitidos.
- ➔ Restrições de controle normativo: Administração de espaços

específicos para usuários designados como os residentes, idosos e portadores de deficiência, e veículos prestadores de serviço.

➔ Restrições Financeiras: Aplicação de preços nos estacionamentos.

Os preços cobrados são baseados em tempos e são definidos não só pelos tamanhos das cidades mas também pela ausência de estacionamentos e a competitividade na região. Está amplamente revelado a necessidade de disciplinar o uso de estacionamento em via pública (MAY, 1999).

3.1. Funcionamento

A principal justificativa para o trabalho apresentado está associada à importância do tema no cenário atual, em consequência do surgimento de diversos problemas resultantes do crescimento de veículos em circulação nas cidades. Por esta razão, o foco desta pesquisa é uma proposta de melhoria do serviço de estacionamento rotativo, visando aperfeiçoar a organização mediante o controle e gerenciamento de empresas cujo ramo de atividade é estacionamento rotativo.

A urbanização das cidades brasileiras ocorreram de forma rápida sem o devido planejamento, fato evidente na precariedade dos transportes públicos, tornando-se determinante para o avanço da individualização do transporte por meio dos carros, sobrecarregando assim, o trânsito das cidades e elevando a demanda por estacionamento, principalmente nos centros comerciais, o que vem acentuando a saturação destas cidades. Com objetivo de amenizar os transtornos causados pela falta de vagas, vem sendo utilizado em diversos países o Estacionamento Rotativo Regulamentado.

> Trata-se de uma forma de trabalhar com um conflito interno à questão da microacessibilidade, que é o tempo de estacionamento. Criou-se assim, o conceito do estacionamento rotativo, que impede a pessoa de ocupar uma vaga além de um número estabelecido de horas. Isso garante o acesso da maioria das pessoas que desejam estacionar por um período breve, democratizando o espaço (Feder e Maciel 2007).

Nesse contexto, podemos citar Elias (2001) no seguinte contexto, o sistema de estacionamento rotativo é uma medida de racionalização das vias, que promove a constante troca dos veículos nas vagas, como benefício, pode-se citar o aumento da oferta dinâmica de vagas, a acessibilidade da área, o incentivo às atividades comerciais neste ramo, a adequação da oferta com a demanda buscada e a melhoria da organização dos estacionamentos mediante concorrentes.

3.2. Softwares Existentes

Com objetivos de analisar por completo as necessidades das empresas de estacionamento, a fim de se obter um controle geral no funcionamento integrado de todas as modalidades disponíveis, e dessa forma, organizar clientes, carros e funcionários, registros de entradas e saídas de veículos, cálculo dos preços de acordo com o período permanente, controle de vagas disponíveis, registrar os clientes e veículos com banco de dados.

> Será simplesmente impossível operar com eficiência mesmo uma pequena

> empresa sem investimentos significativos em sistema de tecnologia da informação que deem o suporte necessário para o desenvolvimento dos planejamentos e da implementação de um modelo de gestão baseado em informações de qualidade (BAPTISTA, 2013, p.18).

Podemos citar o sistema Estacione Simples, uma vez que ele tem como principal objetivo, otimizar as demandas diárias que existem dentro de um estacionamento e auxiliar no gerenciamento administrativo, por meio dele é possível automatizar muito dos processos diários, como o início e o término de uma estadia em uma vaga de estacionamento, além de armazenar todos os dados e transações financeiras que ocorrem diariamente e gerar relatórios completos sobre a situação financeira do estabelecimento.

Com a expansão das quantidades de vagas de estacionamento rotativo nas cidades de médio e grande portes no Brasil, nota-se a inserção do capital privado e do interesse do mercado financeiro nas políticas públicas da escala municipal. Ao mesmo tempo, gera o questionamento: até que ponto o planejamento urbano das áreas centrais desses municípios passa a ser pensado a partir de uma lógica de busca efetiva pela melhoria da mobilidade urbana, ou se pode começar a ocorrer pressão, por parte do capital privado e financeiro, para a expansão das áreas de Zonas Azuis, apenas como ampliação de áreas para recolhimento do pagamento pelo estacionamento rotativo, já que essa atividade passa a representar um retorno interessante para as empresas que viram aí um novo filão para exploração.

Citando outro exemplo de controle de estacionamento, tem-se o Pango (Pay, Park and Go) é um aplicativo para smartphone com a finalidade de controlar estacionamentos regulamentados públicos ou privados. Basta criar uma conta no sistema do Pango e o usuário consegue controlar a quantia gasta nos estacionamentos (PANGO, 2014). O aplicativo Pango foi desenvolvido por uma empresa israelense e está presente em vários países, podendo ser utilizado em diferentes estacionamentos, como estacionamentos privados ou condomínios.

Outro exemplo é o Stop Fácil Estacionamento, que é um sistema que oferece gerenciamento completo de estacionamentos rotativos, com compra de créditos virtuais utilizando cartões de crédito (STOP- FÁCIL, 2014). Nesse sistema os usuários do estacionamento não possuem controle sobre os cartões / tíquetes ou créditos utilizados.

Diante dos modelos apresentados, pode-se perceber que um bom sistema é aquele que permite ao usuário fazer bem o seu trabalho de forma intuitiva, sem que tenha dificuldades e é isso que o Estacione Simples visa trazer: uma interface que tenha na página principal as funcionalidades essenciais para que se realize os processos diários necessários.

4. METODOLOGIAS ÁGEIS

As metodologias ágeis tornaram-se populares em 2001 quando houve o encontro de dezessete especialistas em processos de desenvolvimento de software, representando os métodos Scrum, Extreme Programming, DSDM, Crystal entre outros, eles estabeleceram os princípios comuns que foram compartilhados por todos esses métodos. O resultado desse encontro foi o estabelecimento do “Manifesto Ágil” (AGILE MANIFESTO, 2004).

Foram abordados, de acordo com o AGILE MANIFESTO, 2004, alguns conceitos chave, são os tópicos mencionados a seguir:

- Indivíduos e interações ao invés de processos e ferramentas.
- Software executável ao invés de documentação.
- Colaboração do cliente ao invés de negociação de contratos.
- Respostas rápidas a mudanças ao invés de seguir planos.

As organizações, de todas as naturezas, dependem cada vez mais de sistemas de informação, consequentemente de profissionais qualificados, capazes de criar e manter soluções coerentes com seu planejamento estratégico. Segundo Pressman (2006) é importante ter um processo de software bem definido, um roteiro que deve ser seguido para criar um produto de software de alta qualidade. Muitos processos de desenvolvimento de software foram criados e adaptados ao longo do tempo.

Nos contextos de mudanças, os métodos ágeis têm se destacado, pois oferecem respostas rápidas ao ambiente de desenvolvimento, essas metodologias têm conseguido maior espaço devido a mudanças no perfil dos projetos de software, de modo que as abordagens tradicionais são pouco propícias às mudanças e atrapalha em muitos casos as necessidades do projeto. No entanto, a realidade atual é composta por alterações a todo o momento e requisitos em constante evolução, assim surge a necessidade de metodologias que se adaptem a esse paradigma (AMBLER, 2004).

Os métodos ágeis, constituem uma classe imprescindível para o desenvolvimento de softwares, o mercado atual possui uma crescente demanda por processos mais ágeis, leves e com ciclos de desenvolvimento cada vez mais curtos (ABRAHAMSSON, 2003).

4.1. SCRUM

Segundo a Scrum Alliance (2011), o Scrum é um framework ágil destinado a projetos complexos e foi originalmente formalizado para projetos de desenvolvimento de software, mas funciona bem para qualquer escopo de trabalho inovador. Tem como objetivo fornecer um processo conveniente para projeto e desenvolvimento orientado a objeto. Apresenta uma abordagem que aplica ideias da teoria de controle de processos industriais para o desenvolvimento de softwares, reintroduzindo as ideias de produtividade, flexibilidade e adaptabilidade. Tem como foco encontrar uma forma de trabalho produzir softwares de forma flexível e em um ambiente em constante mudança.

Pressman (2006), explica que o Scrum é um modelo ágil de processo, desenvolvido por Jeff Sutherland e por sua equipe no início da década de 1990, nesse modelo há um padrão de processo em que ocorrem as tarefas de trabalho. De modo que o desenvolvimento de softwares envolve muitas variáveis técnicas, como por exemplo: requisitos, recursos e tecnologia, e eles podem sofrer mudanças no decorrer do processo. Isto torna o processo de desenvolvimento imprevisível e complexo, requerendo flexibilidade para acompanhar as mudanças, Schwaber (1995).

Tabela 1: Práticas ágeis da metodologia Scrum Aplicadas no projeto.

PRÁTICA	DESCRIÇÃO	JUSTIFICATIVA
Product Backlog	Documentação inicial do projeto,contém as informações sobre o escopo do projeto, é acessível por todos os membros da equipe.	Documentação que engloba uma lista de todos os módulos do sistema com a descrição de suas funcionalidades..
Sprints	Formato de divisão do desenvolvimento, cujo período das iterações foi no decorrer de duas, contendo uma lista de funcionalidades a serem desenvolvidas.	Os sprints tratam-se de mapa para o desenvolvimento, no prazo estipulado, norteia a equipe com o plano a ser seguido.
Sprints Backlog	No planejamento da sprint, retiram-se as funcionalidades que serão desenvolvidas, repassada e expõe no quadro de atividades da equipe para facilitar o acesso e visibilidade.	Para cada sprint é realizado uma lista das funcionalidades a serem desenvolvidas, e assim possibilitando o acompanhamento das tarefas da equipe.
Sprint Retrospective	Ao final de uma sprint é realizada uma retrospectiva, onde são discutidos os problemas encontrados, para que não ocorram novamente.	Para melhor desenvolvimento do projeto e também o crescimento do time é importante avaliar os erros e acertos.
Product Owner	Exerce papel fundamental no gerenciamento do projeto, auxilia na tratativa dos problemas, delega responsabilidades, conduz reuniões e tem contato com stakeholders.	Liderança na equipe, sua função principal é facilitar o trabalho dos membros da equipe.
Daily Scrum Meeting	Reuniões curtas e diárias com todos os integrantes da equipe, é abordado o status do projeto e problemas que interfiram no tempo do sprint.	As dailys são importantes para o projeto, pois são abordadas as informações sobre o status atual do projeto e repassada a todos..

(**Fonte:** Autoria própria)

A metodologia Scrum é focada no gerenciamento, planejamento e auxilia na parte estratégica do projeto. Para o desenvolvimento do sistema de gerenciamento de estacionamentos foram utilizadas todas as cerimônias, inclusive a organização deles através do Kanban.

4.2. KANBAN

O KANBAN surgiu no Japão na década de 70, e seu desenvolvimento é

creditado à Toyota Motor, que buscava um método de controle da produção de veículos com a demanda específica de diferentes modelagens e cores e com o mínimo de atraso (SHINGO, 1996).

Segundo Vale (2009), esta técnica começou a ser utilizada em desenvolvimento de software com a introdução das metodologias ágeis. Um quadro com cartões é utilizado para sinalizar o status do trabalho em andamento. Quando a equipe move, por exemplo, um cartão para área de concluído do quadro, ela sinaliza que está pronta para o próximo trabalho. Isso permite a criação de fluxo dentro de uma iteração e impede que a demanda seja empurrada para a equipe em grandes lotes. As vantagens abordadas por Vale (2009) a respeito do Kanban são as seguintes:

- Controle visual das atividades, do que está em andamento ou na fila para iniciar.
- Colabora para a identificação de problemas no processo de desenvolvimento.
- Promove uma equipe multifuncional.
- Estimula a melhoria contínua e a redução dos desperdícios.
- Diminui a burocracia na metodologia de desenvolvimento.
- Estimula o *just in time* que é sobre as atividades serem feitas no tempo correto.

Neste trabalho, foi de fundamental importância para a organização de todo o planejamento do projeto, a ferramenta utilizada foi o Trello[1], e com ele ficou muito evidente todas as atividades que estavam pendentes, as que estavam sendo executadas e o que já estava entregue. É um framework de controle que coloca a equipe toda em evidência, pois, é possível através dos *cards* que são criados delimitar os tempos gastos para desenvolvimento de determinada tarefa e quem é o responsável pela entrega.

5. FERRAMENTAS DE DESENVOLVIMENTO

Nessa seção serão comentadas as ferramentas e procedimentos utilizados durante o desenvolvimento do projeto.

5.1. Git e Github

O Git é um sistema de controle de versão de código aberto, foi projetado e desenvolvido por Linus Torvalds em meados de 2005, sua criação foi[1] direcionada a desenvolvedores de software, contudo sua utilização pode registrar histórico de edições de qualquer tipo de arquivo. Com sua utilização, é possível manter todo o histórico de alterações no código de projetos e facilmente retornar para qualquer ponto anterior para saber como o código estava naquela data. Existem as ramificações do git, que são denominadas de *branchs,* e são formas de se ter uma determinada versão do código que sofreu alterações.No Git, é possível o

[1] Trello: é uma ferramenta que possibilita o gerenciamento visual de projetos, monitoramento de atividades ou fluxos de trabalho, é possível trabalhar de forma personalizada, anexar mídias, ou seja, é uma forma de organização muito eficiente.

recebimento de *commits* de diferentes fontes e por diferentes desenvolvedores, e assim os trabalhos desenvolvidos em equipe, que possui um número indeterminado de pessoas, se tornam mais organizados (GIT, 2022).

Quanto ao Github, foi desenvolvido por Chris Wanstrath, J. Hyett, Tom Preston Werner e Scott Chacon usando as linguagens Ruby e Rails, em fevereiro de 2008 (GITHUB, 2022). Trata-se de uma plataforma de hospedagem de código-fonte, nele contém um controle de versão, onde é utilizado o git.

5.2. VSCode

A Ferramenta Visual Studio proporciona que se crie uma interface de forma rápida e fácil por meio de blocos, permitindo o uso de bibliotecas com ferramentas diferentes e criativas (MICROSOFT, 2019).

> O ambiente de desenvolvimento integrado do Visual Studio é um painel de inicialização criativo que você pode usar para editar, depurar e compilar o código e, em seguida, publicar um aplicativo. Um IDE (ambiente de desenvolvimento integrado) é um programa repleto de recursos que pode ser usado por muitos aspectos do desenvolvimento de software. Além do editor e do depurador padrão fornecidos pela maioria dos IDEs, o Visual Studio inclui compiladores, ferramentas de preenchimento de código, designers gráficos e muitos outros recursos para facilitar o processo de desenvolvimento de software (MICROSOFT, 2019).

Como um dos principais objetivos que o sistema Estacione Simples tem é proporcionar uma boa experiência do usuário, por meio dessa ferramenta foi possível a criação da interface com boa usabilidade, podendo fazer modificações em formatos e cores de forma simples, trazendo uma interface bonita e usual.

5.3. Typescript

O Typescript é uma modificação do javascript, mas como implementam as mesmas especificações pode-se facilmente programar Javascript dentro do código fonte (FENTON, 2014). Deste modo, ao utilizar o Typescript, tem-se a possibilidade de aplicar a tipagem estática à programação juntamente com a orientação de objetos, ou seja, pode-se turbinar as aplicações.

Esta linguagem de programação é compilada para javascript no momento da criação do projeto, o servidor sobe, convertendo-se para node js, assim é possível utilizar todas as vantagens de uma linguagem tipada, unindo os benefícios de uma aplicação node (praticidade, leveza e multiplataforma).

5.4. React

De acordo com NETO e ÁVILA (2020) a documentação oficial, React é um *framework* em *TypeScript,* criado pelo *Facebook*. É utilizada para desenvolver *softwares* de *FrontEnd* para multiplataformas convertendo o código *TypeScript* para *JavaScript*, com o intuito de resolver problemas que causavam considerável transtorno na manutenção de elementos que compõem a interface do usuário.

Assim foi desenvolvida a biblioteca com o objetivo de renderizar páginas HTML tornando aplicativos web mais rápidos, escaláveis e simples. Aplicações em React podem manipular e alterar dados sem a necessidade de carregar páginas.

5.5. SQLite

O SQLite é uma biblioteca em linguagem C, desenvolvida por D. Richard Hipp, implementa um mecanismo de banco de dados SQL embutido, transacional e independente, eliminando consultas e processos separados, não utiliza servidores, seu código é *open source*. (VIEIRA,2022).

Ainda de acordo com VIEIRA (2022), Para o desenvolvimento do sistema, o banco de dados é desenvolvido em tempo de execução, ou seja, no momento em que o projeto é rodado cria-se um arquivo que é o próprio banco que é auto gerenciado. O banco lê e grava diretamente em arquivos de disco comuns, nele podem conter várias tabelas, índices, gatilhos e visualizações em um único arquivo de disco, seu formato de arquivo é multiplataforma.

Dentre as considerações de Bicalho(2014) o SQLite é possível criar e manter diversas tabelas de banco de dados, utilizando-se dos mesmos comandos do SQL. Os dados na tabela são manipulados por meio de comandos DML (*INSERT, UPDATE e DELETE*) e são consultados utilizando o *SELECT*. Algumas características do SQLite:

- Todo o banco de dados é guardado localmente em um arquivo de extensão ".db".
- Não oferece integridade referencial (chaves estrangeiras).
- Suporta o uso de transações (*COMMIT, ROLLBACK*).
- Não deve ser utilizado nos seguintes casos: aplicações de alta concorrência e sistemas ou aplicações web de grande porte.

O uso do SQLite é recomendado quando a simplicidade da administração, implementação e manutenção são mais importantes que vários recursos que SGBDs mais voltados para aplicações complexas implementam. E essa simplicidade é amplamente requisitada hoje em dia (BICALHO, 2014).

5.6. Jest

O *Jest* é um poderoso *framework open source* de testes em *JavaScript* com um foco na simplicidade. Ele visa trabalhar fora da caixa, sem configuração, na maioria dos projetos *JavaScript*. Fazer testes que tenham objetos grandes com facilidade. Os testes são paralelos e executados em seus próprios processos para maximizar o desempenho. (JEST, 2022)

Neste trabalho o *Jest* foi usado para implementação dos testes unitários do *Backend* que foram escritos em *TypeScript*. Criado pela equipe do *Facebook, o framework,* necessita de pouca configuração, possui uma comunidade ativa e uma grande documentação, oferece uma grande quantidade de recursos para a escrita de testes de forma rápida e simples para códigos em *TypeScript* (SILVA, 2021).

6. DESENVOLVIMENTO DO PROJETO

A elaboração e o desenvolvimento de um sistema, seja qual for a sua natureza, passa pelos ciclos básicos de planejamento, análise, projeto e implementação (Dennis; Wixon, 2005). Em geral, na fase de planejamento busca-se saber os motivos de se construir o sistema, o problema que será resolvido, a viabilidade técnica, operacional e financeira de construí-lo e por fim a definição da equipe, entre outros aspectos.

6.1. Elicitação de Requisitos

Segundo Alexa e Avasilcai (2018), uma das etapas da engenharia de requisitos é a elicitação de requisitos (ER), na qual ocorre a extração das necessidades e desejos dos usuários, que são especificados como requisitos. “Um requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar, para atingir os seus objetivos” (Pfleeger, 2004, p. 111).

Esses requisitos são divididos em funcionais, que representam as funcionalidades que um sistema deve ter, e não funcionais, que qualificam determinados aspectos do sistema (PAULA FILHO, 2019).

Na Figura 1 estão apresentados os requisitos funcionais identificados durante o processo de licitação do sistema Estacione Fácil. A documentação completa gerada para o sistema pode ser obtida por meio dos links:

- FrontEnd: web em (ÁVILA, DUARTE, 2022)
- BackEnd: api em (ÁVILA, DUARTE, 2022)
- Documentação: doc em (ÁVILA, DUARTE, 2022)

Figura 1: Requisitos Funcionais do Sistema.

[RF001] - Tipo de veículo - O sistema deve permitir o cadastro, alteração, exclusão e visualização dos tipos de veículos. Os campos que compõem essa funcionalidade são: identificador, descrição, valor mensalidade, quantidade de horas diárias, valor 5 minutos, valor 10 minutos, valor 15 minutos, valor 20 minutos, valor 25 minutos, valor 30 minutos, valor 35 minutos, valor 40 minutos, valor 45 minutos, valor 50 minutos, valor 55 minutos, valor 60 minutos, primeira hora fracionada, criado em, atualizado em, deletado em ativo em.

[RF002] - Tipo de pagamento - O sistema deve permitir o cadastro, alteração, exclusão e visualização dos tipos de pagamento, os campos que compõem essa funcionalidade são: identificador, descrição, criado em, atualizado em, deletado em ativo em.

[RF003] - Tipo de desconto - O sistema deve permitir o cadastro, alteração, exclusão e visualização dos tipos de pagamento, os campos que compõem essa funcionalidade são: identificador, quantidade de paradas, porcentagem, tipo do veículo, criado em, atualizado em, deletado em ativo em.

[RF004] - Tipo de cliente- O sistema deve permitir o cadastro, alteração, exclusão e visualização dos tipos de cliente, os campos que compõem essa funcionalidade são: identificador, descrição, criado em, atualizado em, deletado em ativo em.

[RF005] - Mensalistas - O sistema deve permitir o cadastro, alteração, exclusão e visualização dos clientes que pagam mensalmente, os campos que compõem essa funcionalidade são: identificador, tipo do veículo, placa, nome do proprietário, telefone, cpf, dia pagamento, criado em, atualizado em, deletado em ativo em.

[RF006] - Conveniado - O sistema deve permitir o cadastro, alteração, exclusão e visualização das empresas que são conveniadas, os campos que compõem essa funcionalidade são: identificador,

pacote de horas, quantidade de hora completa, hora parcial, nome da empresa, telefone, cnpj, dia pagamento, criado em, atualizado em, deletado em ativo em.

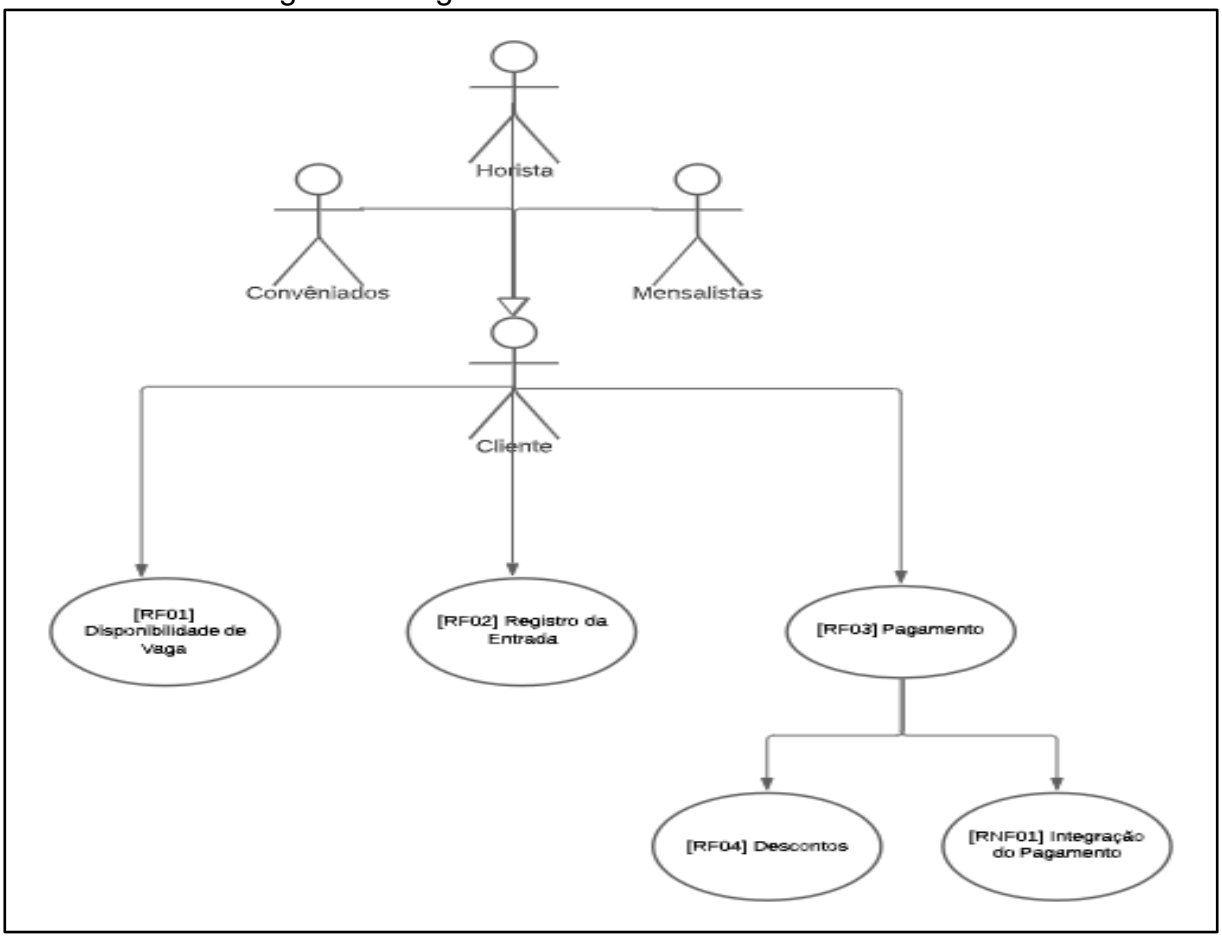
[RF007] - Horistas - O sistema deve permitir o cadastro, alteração, exclusão e visualização dos clientes que pagam por hora, os campos que compõem essa funcionalidade são: identificador, tipo de veículo, placa, hora de entrada, hora de saída, tipo de pagamento, desconto, tipo de cliente, criado em, atualizado em, deletado em ativo em.

(**Fonte:** Autoria própria)

6.2. Diagrama de Caso de Uso

É a representação das funcionalidades externamente observáveis do sistema e dos elementos externos ao sistema e, que com ele interagem (BEZERRA, 2007). Trata-se de um instrumento eficiente para determinação e documentação das funcionalidades a serem desempenhadas pelo sistema e é também um bom meio para comunicação com os clientes no processo de definição dos requisitos do sistema. Na Figura 2 está apresentado o Diagrama de Casos de Uso do sistema Estacione Fácil.

Figura 2: Diagrama de Casos de Uso do Sistema.



(**Fonte:** Autoria própria)

6.3. Matriz de Rastreabilidade

A matriz de rastreabilidade assegura que cada requisito agregue valor de negócio através da sua ligação aos objetivos de negócio e do projeto, assim é mapeado um meio de rastreamento do início ao fim do ciclo de vida do projeto, ajudando a garantir que os requisitos aprovados na documentação sejam entregues no final do projeto, também apresenta uma estrutura de gerenciamento das mudanças do escopo do produto.

Quanto aos atributos associados aos requisitos, eles devem ser registrados na matriz de rastreabilidade de requisitos, eles contribuem para a definição de informações chave a respeito do requisitos. Os atributos mais usuais são: um identificador único, uma descrição textual do requisito, os argumentos para sua inclusão, proprietário, fonte, prioridade, versão, status (se está ativo, cancelado, aprovado, ou concluído) e a data do status. Abaixo está demonstrada uma matriz de rastreabilidade de requisitos com seus atributos associados:

Tabela 2: Matriz de Rastreabilidade do Sistema.

[RF01]	Verificar disponibilidade de vaga
Descrição	Assim que o cliente chegar, seja ele horista ou mensalista, é necessário identificar se há vagas disponíveis para ocupação.
Prioridade	Alto
Caso de uso relacionado	[UC01]
[RF02]	**Registro da Entrada**
Descrição	Após a identificação da vaga, o cliente precisa fazer o registro do veículo através da placa e a partir de então será computado o valor a ser cobrado pelo período de permanência.
Prioridade	Alto
Caso de uso relacionado	[UC02]
[RF03]	Pagamento da Estadia
Descrição	O sistema deverá realizar o cálculo, onde estão previstos os parâmetros para pagamento.
Prioridade	Alto
Caso de uso relacionado	[UC03]
[RF04]	Descontos
Descrição	Deverão ser previstos os parâmetros de hora cheia ou hora fracionada. A partir de então deverá ser realizada a cobrança para o cliente
Prioridade	Alto
Caso de uso relacionado	[UC04]

[RNF01]	Integração do Pagamento
Descrição	O software precisa ter integração para que o pagamento seja processado
Prioridade	Alto
Caso de uso relacionado	[UNC01]

(**Fonte:** Autoria Própria)

6.4. Diagrama de Classe

De acordo com Guedes (2011, p. 103), não é realmente obrigatório que uma classe apresente as três divisões, pois, pode haver classes que não tenham atributos ou que não contenham métodos, ou pode acontecer ainda que seus atributos e métodos não precisem ser apresentados no diagrama, já que é recomendado apresentar apenas atributos relevantes ao diagrama para evitar, por exemplo, tornar o diagrama muito poluído. O Diagrama de classes desenvolvido para o projeto está apresentado na Figura 3 e está disponível na pasta docs em (ÁVILA, DUARTE, 2022)

6.5. Modelo Entidade Relacionamento

O relacionamento é uma associação entre várias entidades e representa a maneira como essas entidades podem estar logicamente relacionadas. Já os atributos têm a função de mapear um conjunto de entidades num domínio e identifica, qualifica e descreve esse conjunto de entidades. O Modelo Entidade Relacionamento do projeto está apresentado na Figura 4 e está disponível na pasta doc em (ÁVILA, DUARTE, 2022)

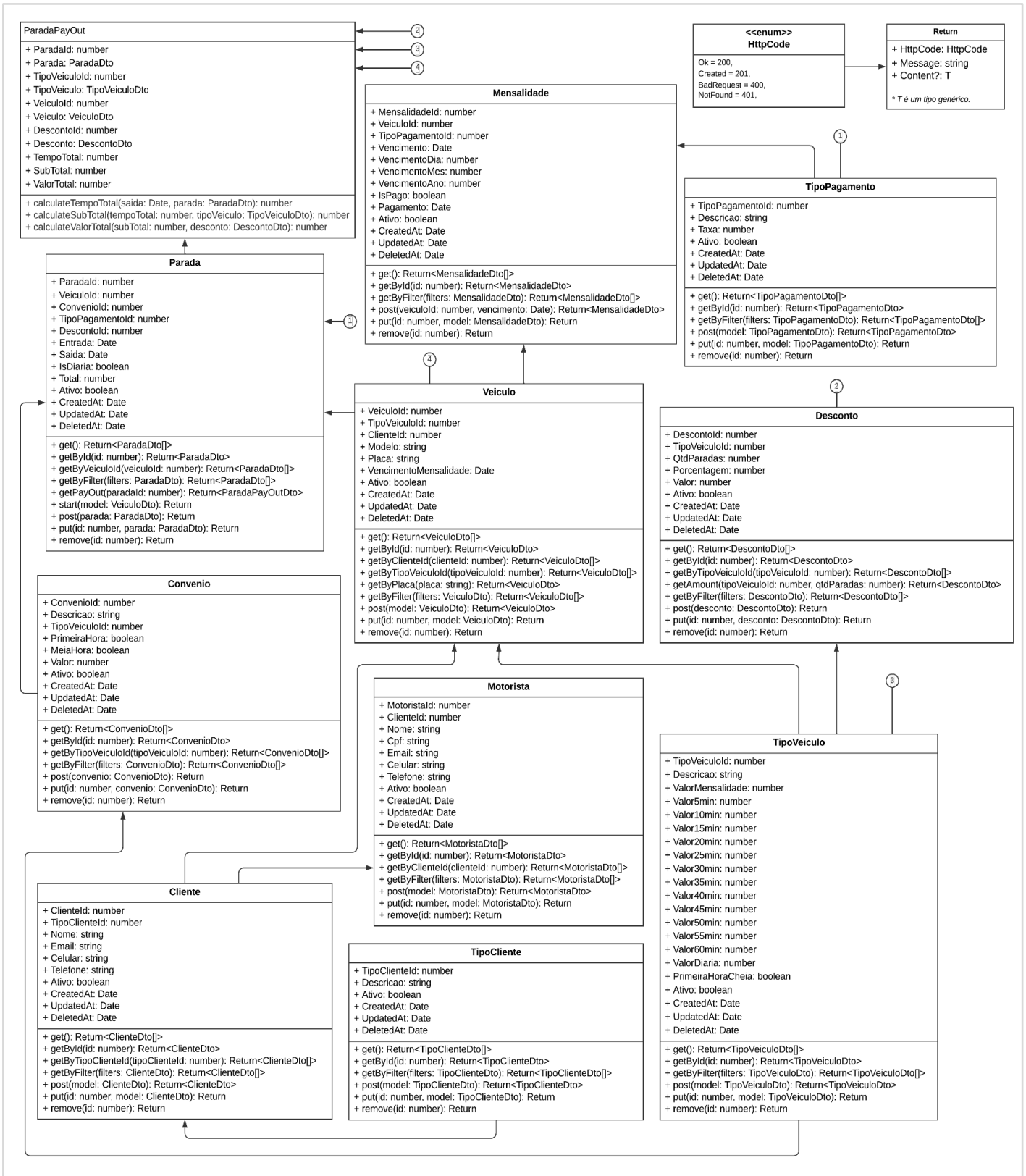
6.6. Prototipação e Codificação

Durante a execução de todo projeto, foi seguida uma prototipação de telas, e seu principal objetivo é simular a interação entre o homem e a máquina, encontra-se disponível na pasta doc em (ÁVILA, DUARTE, 2022). Também foi desenvolvida uma aplicação *FrontEnd*, cuja linguagem é baseada no *Typescript*, utilizando o *framework React*, e uma aplicação *BackEnd*, desenvolvida na linguagem *typescript*.

No desenvolvimento da aplicação *FrontEnd* foram criados arquivos que possuem o sufixo *Application*, que são os arquivos responsáveis pela comunicação da aplicação com a *API*. Na Figura 5 está apresentado um componente que irá apresentar as mensagens de alerta do sistema que são retornadas pela *API*.

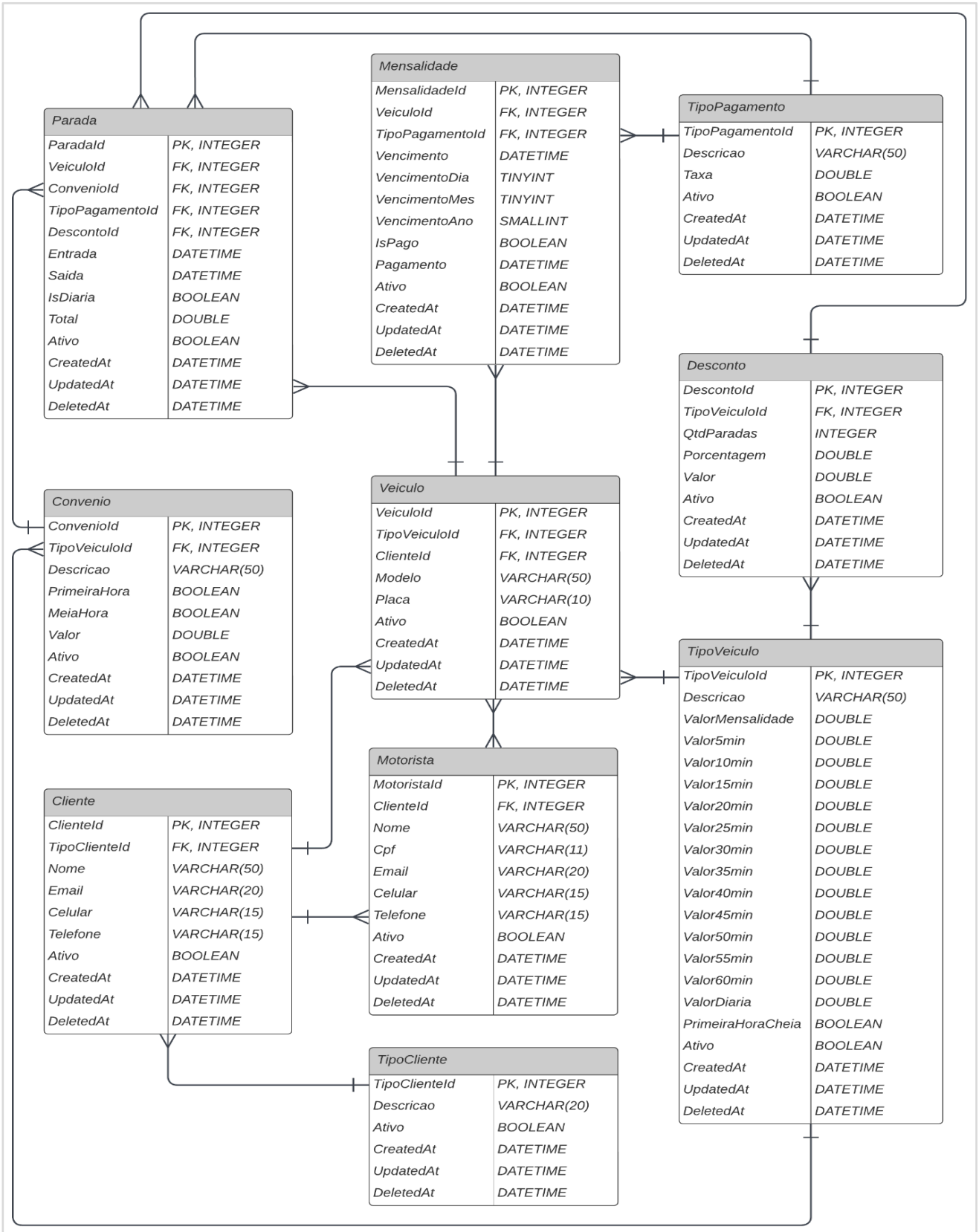
A aplicação *BackEnd* foi criada seguindo o padrão de desenvolvimento das *API's RESTful*, com os principais métodos *HTTP's* que são os seguintes: *get* (exibe os dados), *post* (envia os dados), *put* (altera os dados) e *delete* (remove os dados). Na Figura 6 pode-se identificar um método que foi criado na API para verificação de execução da API.

Figura 3: Diagrama de Classes do Sistema.



(**Fonte:** Autoria Própria)

Figura 4: MER do Sistema.



(**Fonte:** Autoria própria)

Figura 5: Componente de Exibição de Alertas do Sistema.

```tsx
import { Variant } from "react-bootstrap/esm/types";

import React from "react";
import { Alert } from "react-bootstrap";

import { AlertaBackground } from "@components/alerta/styled";

import { LayoutContext } from "@layouts/main";

export interface AlertaProps {
  show: boolean;
  variant?: Variant;
  message?: string;
}

const AlertaComponent: React.FC<AlertaProps> = (props) => {
  const { Alerta } = React.useContext(LayoutContext);

  return (
    <AlertaBackground className="bg-alerta" show={props.show}>
      <Alert show={props.show} variant={props.variant}>
        <span>{props.message}</span>

        <i className="bi bi-x-lg icone" onClick={() => Alerta.Hide()}></i>
      </Alert>
    </AlertaBackground>
  );
};

export default AlertaComponent;
```

(**Fonte:** Autoria Própria)

Figura 6: Método de verificação de execução da API.

```ts
import { Get, JsonController } from "routing-controllers";
import Return from "../dtos/Return";
import HttpCode from "../enums/HttpCode";

@JsonController("", { transformResponse: true })
export class PingController {
  @Get("")
  async get() {
    const PORT = process.env.PORT || 8080;

    return new Return(HttpCode.Ok, `Servidor online na porta ${PORT}...`, {
      date: new Date(),
    });
  }
}
```

(**Fonte:** Autoria Própria)

6.7. Testes

O teste de software é uma parte crucial do desenvolvimento de um software. Ele irá, como o nome sugere, testar e verificar se o software consegue entregar corretamente tudo que ele propõe.

> O processo de desenvolvimento de software envolve uma série de atividades nas quais, apesar das técnicas, métodos e ferramentas empregados, erros no produto ainda podem ocorrer. Atividades agregadas sob o nome de Garantia de Qualidade de Software têm sido introduzidas ao longo de todo o processo de desenvolvimento, entre elas atividades de VV&T – Verificação, Validação e Teste, com o objetivo de minimizar a ocorrência de erros e riscos associados.
> Dentre as técnicas de verificação e validação, a atividade de teste é uma das mais utilizadas, constituindo-se em um dos elementos para fornecer evidências da confiabilidade do software em complemento a outras atividades, como por exemplo o uso de revisões e de técnicas formais e rigorosas de especificação e de verificação (BARBOSA, 2000).

Esses testes podem ser realizados em qualquer uma das partes do software, desde a unidade pequena até seu funcionamento como um todo, analisando também o número de dados e a sua segurança.

6.7.1. Teste Automatizado

Juntamente com a construção do projeto, foram escritos testes unitários cujo o objetivo é auxiliar no desenvolvimento e garantir que o código escrito funcione como o esperado, ou seja, resume-se em comparar dados válidos e inválidos via I/O (entrada/saída). Os testes foram escritos utilizando uma biblioteca chamada *JEST*.

Para cada função a ser testada foram desenhados dois cenários, os mesmos cenários poderiam se aplicar a outras entidades. Os cenários estão exemplificados na Tabela 3, com o cenário "Tipos de Clientes".

Tabela 3 - Cenários de Testes

Nome	Cenário
T1 - Buscar Todos os Tipos de Clientes	GET - Buscar Todos os Tipos de Clientes, somente para os registros com Ativo igual a *true*.
T2 - Buscar Pelo Id o Tipo de Cliente	GET - Buscar Pelo Id o Tipo de Cliente, somente para os registros com Ativo igual a *true*.
T3 - Buscar Personalizada os Tipos de Clientes	GET - Buscar Personalizada os Tipos de Clientes, somente para os registros com Ativo igual a *true*.
T4 - Cadastrar um Tipo de Cliente	POST - Cadastrar um Tipo de Cliente, respeitando a restrição de Descrição única.
T5 - Atualizar um Tipo de Cliente	PUT - Atualizar um Tipo de Cliente, respeitando a restrição que o registro já esteja cadastrado.
T6 - Remover um Tipo de Cliente	DELETE - Remover um Tipo de Cliente, respeitando a

	restrição que o registro já esteja cadastrado.

(**Fonte:** Autoria Própria)

Para uma melhor qualidade dos testes, cada um dos cenários (de T1 a T6), possui 2 casos de testes, com o objetivo de testar como o código escrito no *BackEnd* se comporta em situações diferentes. Os casos estão exemplificados na Tabela 4, para o cenário "T1 - Buscar Todos os Tipos de Clientes".

Tabela 4 - Casos de Testes para cada Cenário

Nome do Caso de Teste	Objetivo do Teste	Entrada da Função	Saída da Função	Resultado Esperado na Execução
Deve buscar todos os tipos de clientes que estão ativos	Para a entrada válida a função deve ser executada de acordo com a forma esperada	Não é necessário	Uma lista com todos os tipos ativos de clientes.	Sucesso
Não deve buscar nenhum tipo de cliente	Para a entrada válida a função deve ser executada de acordo com a forma esperada	Não é necessário	Uma lista vazia, pois todos os tipos de cliente estão inativos	Sucesso

(**Fonte:** Autoria Própria)

Na Figura 7 é possível visualizar a escrita dos testes do fluxo de Tipo de Cliente para o cenário "T1 - Buscar Todos os Tipos de Clientes" e seus casos de testes já implementados.

Figura 7: T1 - Buscar Todos os Tipos de Clientes.

```
describe("Busca Todos os Tipos de Clientes (somente ativos)", () ⇒ {
  test("Retorna uma Lista com os Tipos Ativos", async () ⇒ {
    // arrange
    const tipoCliente = new TipoClienteEntity();
    const tiposClientes = [tipoCliente];
    const resposta = new Return(HttpCode.Ok, "Ok", tiposClientes);

    jest.mocked(TipoClienteRepository).Get.mockResolvedValue(resposta);

    // act
    const response = await new TipoClienteController().get();

    // accert
    expect(response).toBe(resposta);
  });

  test("Retorna uma Lista Vazia (todos os regristros estão inativos)", async () ⇒ {
    // arrange
    const tiposClientes = [];
    const resposta = new Return(HttpCode.Ok, "Ok", tiposClientes);

    jest.mocked(TipoClienteRepository).Get.mockResolvedValue(resposta);

    // act
    const response = await new TipoClienteController().get();

    // accert
    expect(response).toBe(resposta);
  });
});
```

(**Fonte:** Autoria Própria)

Na Figura 8 é possível visualizar os resultados dos testes do fluxo de Tipo de

Cliente para o cenário "T1 - Buscar Todos os Tipos de Clientes" e seus casos de testes já implementados.

Figura 8: Resultado dos Testes Automatizados do Fluxo de Tipo de Cliente.

```
$ yarn test
yarn run v1.22.19
$ jest
PASS __tests__/repositories/TipoCliente.Test.ts
  Testando o Fluxo de Tipo de Cliente
    Busca Todos os Tipos de Clientes (somente ativos)
      √ Retorna uma Lista com os Tipos Ativos (2 ms)
      √ Retorna uma Lista Vazia (todos os regristros estão inativos)
    Busca Personalizada os Tipos de Clientes (somente ativos)
      √ Retorna uma Lista com os Tipos Ativos (1 ms)
      √ Retorna uma Lista Vazia (não encontrado de acordo com os filtros) (3 ms)
    Busca Pelo Id o Tipo de Cliente (somente ativo)
      √ Retorna um Tipo de Cliente
      √ Não Retorna um Tipo de Cliente (não encontrado)
    Cadastra um Tipo de Cliente
      √ Retorna sucesso na execução (1 ms)
      √ Retorna falha na execução (duplicidade de descrição)
    Atualiza um Tipo de Cliente
      √ Retorna sucesso na execução (1 ms)
      √ Retorna falha na execução (duplicidade de descrição)
    Remove um Tipo de Cliente
      √ Retorna sucesso na execução
      √ Retorna falha na execução (não encontrado) (1 ms)

Test Suites: 1 passed, 1 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        1.446 s, estimated 2 s
Ran all test suites.
Done in 2.94s.
```

(**Fonte:** Autoria Própria)

7. CONSIDERAÇÕES FINAIS

O projeto ocorreu devido ao aprimoramento e desenvolvimento de uma ideia, com o levantamento de requisitos através de especialistas da área, a fim de termos um protótipo, prosseguiremos para o próximo passo de planejamento de execução do projeto, que incluíam definição de escopo, implementação de metodologia de desenvolvimento e artefatos a serem realizados por meio da Engenharia de Software.

A partir da definição do escopo, foram mapeados os entregáveis de cada Sprint, utilizando a metodologia de desenvolvimento ágil Scrum, para o controle de tarefas e prazo foi utilizada a metodologia Kanban. Os artefatos de engenharia de software e o desenvolvimento do aplicativo, foram iniciados e seguidos de acordo com o planejamento mensal.

O projeto que teve como objetivo, o desenvolvimento de um software para estacionamento rotativo pago, a fim de obter um controle por meio da sistematização do serviço prestado, como por exemplo o registro dos clientes e veículos, controle de vagas disponíveis, registros de entradas e saídas de veículos, cálculo dos preços de acordo com o período de permanência.

No processo de desenvolvimento foram identificadas duas dificuldades, sendo a definição do banco de dados, pois era necessário um repositório gratuito, de código livre e multiplataforma. A outra dificuldade encontrada foi a responsividade das telas no FrontEnd, devido a possibilidade de acesso via mobile ou desktop dessa forma fazendo com que haja variação no tamanho das telas.

Os resultados obtidos juntamente com a utilização das metodologias ágeis foram fundamentais para o desenvolvimento do projeto e a organização da equipe, sendo determinante para o alcance dos objetivos propostos. E assim, resultando em melhorias com relação à gestão do tempo, à organização e visando maior lucratividade. Por fim, conclui-se que a proposta do protótipo é viável e trará benefícios de gestão para os estacionamentos rotativos.

REFERÊNCIAS

ALEXA, L.; AVASILCAI, S. The requirement elicitation process of designing a collaborative environment – the CRE@TIVE.BIZ case. MATEC Web of Conferences, 184 , 1-4. 2018. Disponível em: The requirement elicitation process of designing a collaborative environment – The CRE@TIVE.BIZ case. Acesso em 18 ago 2022.

AMBLER, Scott W. Modelagem Ágil: Práticas eficazes para a Programação Extrema e o Processo Unificado. In: Porto Alegre, Bookman, 2004.

ABRAHAMSSON, P. et al. New directions on agile methods: a comparative analysis. In:INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 25., 2003, Portland. Anais eletrônicos[...]. Portland: ICSE, p. 244-254, 2003

ABRAMOVAY, R. Mobilidade versus carrocentrismo. Folha de S. Paulo, São Paulo, 14 dez. 2011.

Agile Manifesto, http://agilemanifesto.org/, acessado em 25 de março de 2022.

ÁVILA, D. C.; DUARTE, M.T. Repositório do projeto de TCC no GITLAB. Disponível em https://gitlab.com/shiro-shiro. Último acesso: Novembro de 2022.

BARBOSA, Ellen Francine et al. Introdução ao teste de software. Minicurso apresentado no XIV Simpósio Brasileiro de Engenharia de Software (SBES 2000), 2000.

BATISTA, Emerson. Sistema de Informação. 2 Ed. São Paulo: Saraiva, 2013. Disponível em: <https://computacao.ucpel.edu.br/wp-content/uploads/2016/06/2011_7.pdf>. Acesso em: 24 de março de 2022.

BEZERRA, Eduardo. Princípios de análise e Projeto de sistemas com UML. 2. ed., Rio de Janeiro: Elsevier, 2007.

BICALHO, Conrado. Banco de Dados em Dispositivos Móveis Instituto de Ciências Exatas e Biológicas. Universidade Federal de Ouro Preto – UFOP, 2014.

BOONLERTVANICH, K. Extended CONWIP KANBAN System - Control and Performance Analysis. Georgia Institute of Technology, 2005.

CAMPOS, Vânia Barcellos Gouvêa. Uma visão da mobilidade urbana sustentável. Revista dos Transportes Públicos, v. 2, n. 99-106, p. 4, 2006.

CASTELL, Manuel. “A Sociedade em Rede”. São Paulo. SP. Ed. Paz e Terra, 1999.

COSTA, Marcela da Silva. Um índice de mobilidade urbana sustentável. 2008. 274 f. 2008. Tese (Doutorado) – Tese (doutorado em Planejamento e Operação de Sistema de Transporte) - Escola. Acesso em: 20 nov. 2022.

DENNIS, Alan; WIXON, Barbara. Análise e projeto de sistemas. Rio de Janeiro: LTC, 2005.

FEDER, M. MACIEL, L. B. Panorama da Zona Azul no Brasil. 16º Congresso Brasileiro de Transporte e Trânsito. Maceio, AL, 2007.

FENTON, Steve. Pro TypeScript: Application-Scale JavaScript Development. New York: Apress, 2014. 248 p.

FERRAZ, Antônio Clóvis Pinto. TORRES, Isaac Guillermo Espinosa. Transporte Público Urbano. 2a Ed. Rima. São Carlos, 2004.

FIORIN, Diogo Celeste; STARES, Sérgio Constantino. Análise da ocupação das vagas de estacionamento na área central do município de Joaçaba, SC. Conhecimento em Construção, v. 6, p. 21-34, 2019.

GIT. About. 2022. Disponível em: <https://git-scm.com/about>. Acesso em: 25 fev. 2022.

GITHUB. About. 2022. Disponível em: <https://github.com/about>. Acesso em: 25 fev. 2022.

GUEDES, Gilleanes T. A. UML2: uma abordagem prática. 2. ed. São Paulo: Novatec Editora, 2011.

JEST. Página Inicial, Disponível em: <https://jestjs.io/pt-BR>. Acesso em: 11 de out. 2022.

KNIBERG, H. Scrum e XP direto das trincheiras: como nós fazemos Scrum. São Paulo: InfoQ, 2007.

KUHN, Thomas S. “A Estrutura das Revoluções Científicas”. São Paulo. SP. Ed. Perspectiva, 1997.

IPEA – INSTITUTO DE PESQUISA ECONÔMICA APLICADA. A nova lei de diretrizes da Política Nacional de Mobilidade Urbana. Brasília: Ipea, 2012. (Comunicado do Ipea, n. 128).

MAY, A. D. Car use and traffic management measures in the policy package. ECMT/OECD Workshop on Managing Car Use for Sustainable Urban Travel. Dublin, Ireland, 1999. 19p.

NETO, Cícero Alvarenga Santos; ÁVILA, Donato Cardoso; SRM - SOLUÇÃO PARA GESTÃO DE RELACIONAMENTO COM ESTUDANTE. Revista Eletrônica de Computação Aplicada, Uni-FACEF, Franca, v. 1, n. 2, 2020.

PANGO. Sistema Pango no Brasil. 2014. Disponível em: Acesso em: 20 out 2022.

PARADELA, Carolina Soares Matuck. el al. Estacionamento rotativo: uma abordagem ampla a partir do exemplo de Belo Horizonte. Engenharia, v.3, n.1, Jan/2015.

PAULA FILHO, W. de P. Engenharia de software: produtos. 4. ed. Rio de Janeiro: LTC, 2019.

PFLEEGER, L. Engenharia de Software: teoria e prática. 2. ed. São Paulo: Prentice Hall; 2004.

PRESSMAN, R. S. Engenharia de software. 6. ed. São Paulo: McGraw-Hill, 2006.

RODRIGUES, M. O. AVALIAÇÃO DA QUALIDADE DO TRANSPORTE COLETIVO DA CIDADE DE SÃO CARLOS. 2006. Dissertação de Mestrado–Universidade de São Paulo, São Carlos, 2006.

SCRUM ALLIANCE. Learn about Scrum. [S.l:s. n], 2011. Disponível em: <https://www.scrumalliance.org/why-scrum>. Acesso em: 10 out 2022.

SILVA, Matheus Rodrigues da. Gnuplot on web (GOW): uma aplicação para uso das funcionalidades do gnuplot em fóruns de ambientes virtuais de aprendizagem. 2021.

SOMMERVILLE, I. Engenharia de software. 8.ed. São Paulo: Pearson Addison Wesley, 2007.

SCHWABER, K. "Scrum Development Process", OOPSLA'95 Workshop on Business Object Design and Implementation. Springer-Verlag. (1995)

SHINGO, Shingeo. O Sistema Toyota de Produção – Do ponto de vista da engenharia de produção. 2ª ed. Porto Alegre: Bookman, 1996. 282 p.

TERÁN, José Ángel. “CET 30 Anos”, São Paulo. SP. Imprensa Oficial do Estado de São Paulo, 2006.

STOPFACIL. Stopfacil Gerenciador de Estacionamentos. 2014. Disponível em: Acesso em: 15 nov 2022.

VALE, A.; Software Zen. 2009. Disponível em: https://softwarezen.me/alisson-vale/. Acesso em: 22 set 2022.

VIEIRA, Danielle. 2022. Disponível em: https://www.hostgator.com.br/blog/sqlite-o-que-e-como-funciona-e-qual-e-a-diferenca-entre-o-mysql. Acesso em: 04 out 2022.