



UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ
CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

LEANDRO BESERRA DO NASCIMENTO

**UM ESTUDO EMPÍRICO SOBRE O CHATGPT APLICADO A ESCRITA E
DESENVOLVIMENTO DE REQUISITOS NO FORMATO DE HISTÓRIA DE USUÁRIO**

QUIXADÁ

2023

LEANDRO BESERRA DO NASCIMENTO

UM ESTUDO EMPÍRICO SOBRE O CHATGPT APLICADO A ESCRITA E
DESENVOLVIMENTO DE REQUISITOS NO FORMATO DE HISTÓRIA DE USUÁRIO

Trabalho de Conclusão de Curso apresentado ao
Curso de Graduação em Engenharia de software
do Campus de Quixadá da Universidade Federal
do Ceará, como requisito parcial à obtenção do
grau de bacharel em Engenharia de software.

Orientadora: Profa. Dra. Paulyne Matthews
Jucá

QUIXADÁ

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas

Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

N196e Nascimento, Leandro Beserra do.

Um estudo empírico sobre o CHATGPT aplicado a escrita e desenvolvimento de requisitos no formato de história de usuário / Leandro Beserra do Nascimento. – 2023.
43 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2023.

Orientação: Profa. Dra. Paulyne Matthews Jucá.

1. CHATGPT. 2. História de usuário. 3. Desenvolvimento de código. I. Título.

CDD 005.1

LEANDRO BESERRA DO NASCIMENTO

UM ESTUDO EMPÍRICO SOBRE O CHATGPT APLICADO A ESCRITA E
DESENVOLVIMENTO DE REQUISITOS NO FORMATO DE HISTÓRIA DE USUÁRIO

Trabalho de Conclusão de Curso apresentado ao
Curso de Graduação em Engenharia de software
do Campus de Quixadá da Universidade Federal
do Ceará, como requisito parcial à obtenção do
grau de bacharel em Engenharia de software.

Aprovada em: 12/07/2023

BANCA EXAMINADORA

Profa. Dra. Paulyne Matthews Jucá (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Jeferson Kenedy Moraes Vieira
Universidade Federal do Ceará (UFC)

Prof. Dr. Victor Aguiar Evangelista de Farias
Universidade Federal do Ceará (UFC)

RESUMO

Ferramentas de inteligência artificial vindo sendo usadas para automatizar atividades em várias áreas da engenharia de software. Uma dessas ferramentas que tem ganhado popularidade é o ChatGPT, especialmente porque permite interações em linguagem natural. Os estudos mostram que o ChatGPT possui habilidades em processamento de linguagem natural e geração de código, mas que também apresenta respostas incorretas em várias situações. Ainda não está claro a capacidade do ChatGPT em atividades da engenharia de softwares, como: na geração de código, na correção de bugs, na geração de histórias de usuário ou outras atividades relacionadas. Então, se faz necessários estudos adicionais para ampliar o entendimento do pontência do ChatGPT e suas utilidades na engenharia de software.

Nesse estudo duas etapas são conduzidas: na primeira, 50 histórias de usuário são geradas com critérios de aceitação, complexidade, estimativa de esforço e requisitos relacionados; na segunda, 8 histórias de usuário são selecionadas da lista de 50 requisitos do passo 1 e codificadas utilizando NodeJS com Typescript e Prisma. Para a geração de código são utilizados prompts em 3 níveis diferentes de descrição dependendo do retorno do ChatGPT.

Embora o ChatGPT tenha conseguido gerar os requisitos com sucesso, a geração de código apresentou bugs e erros de sintaxe, com algumas histórias exigindo correção manual. Apesar disso, os requisitos gerados mostram-se úteis como ponto de partida para o desenvolvimento de documentos de requisitos na engenharia de software.

Como resultado desse estudo é apresentado a porcentagem de sucesso que o ChatGPT apresenta na codificação de histórias de usuário, o projeto nodeJs desenvolvido pelo ChatGPT e a lista de 50 requisitos gerados e a respectiva análise geral do resultado.

Palavras-chave: ChatGPT, Code Generation, User Stories, Large Language Models

ABSTRACT

Artificial intelligence tools have been used to automate activities in various areas of software engineering. One such tool that has gained popularity is ChatGPT, especially because it allows interactions in natural language. Studies show that ChatGPT possesses capabilities in natural language processing and code generation, but it also provides incorrect responses in several situations. The capacity of ChatGPT in software engineering activities, such as code generation, bug correction, user story generation, or other related tasks, is still not clear. Therefore, additional studies are necessary to further understand the potential and utilities of ChatGPT in software engineering.

In this study, two stages are conducted: firstly, 50 user stories are generated, including acceptance criteria, complexity, effort estimation, and related requirements. Secondly, 8 user stories are selected from the list of 50 requirements in step 1 and coded using NodeJS with Typescript and Prisma. For code generation, prompts at three different levels of description are used based on ChatGPT's output.

Although ChatGPT successfully generated the requirements, the code generation process exhibited bugs and syntax errors, with some stories requiring manual corrections. Nevertheless, the generated requirements prove useful as a starting point for developing software engineering documents.

The study's results include the success percentage of ChatGPT in coding user stories, the NodeJS project developed by ChatGPT, the list of 50 generated requirements, and an overall analysis of the outcome.

Keywords: Serious Games. Serious Games in Adult Education. Youth and Adult Education

SUMÁRIO

1	INTRODUÇÃO	6
1.1	Objetivos	7
1.1.1	<i>Objetivos Geral</i>	7
1.1.2	<i>Objetivos Específicos</i>	7
2	TRABALHOS RELACIONADOS	9
2.1	How Secure is Code Generated by ChatGPT?	9
2.2	ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design	9
2.3	An Analysis of the Automatic Bug Fixing	10
2.4	Tabela de comparação entre estudos	10
3	FUNDAMENTAÇÃO TEÓRICA	12
3.1	ChatGPT	12
3.2	Prisma	13
3.3	Histórias de usuário	14
4	METODOLOGIA	16
4.1	Escrita e análise das histórias de usuário	16
4.2	Definição do escopo do projeto	19
4.3	Desenvolvimento das histórias de usuário	19
4.4	Teste exploratório no requisito desenvolvido	21
5	RESULTADO	22
5.1	Requisitos propostos	22
5.2	Estrutura final do projeto	23
5.3	Desenvolvimento das Histórias de Usuário	24
5.4	Discussão de resultado	28
6	CONCLUSÃO	29
7	AMEAÇAS A VALIDADE	31
	REFERÊNCIAS	32
	GLOSSÁRIO	32
	APÊNDICES	33
	APÊNDICE A-LISTA DE REQUISITOS PROPOSTA PELO CHATGPT	33

1 INTRODUÇÃO

Com o avanço da inteligência artificial vários modelos de linguagem em grande escala (LLM - *Large language model*) vem sendo utilizados para geração de código e artefatos relacionados a engenharia de software (WHITE *et al.*, 2023b) (SOBANIA *et al.*, 2023). O ChatGPT é uma dessas ferramentas que tem ganhado destaque nos últimos meses, onde uma importante característica é a possibilidade de conversar com o modelo de linguagem podendo ser usado para discutir o código fonte, pedir sugestões de melhorias e correções (SOBANIA *et al.*, 2023). Essa habilidade possibilita obter melhores resultados nas respostas, por exemplo: na melhoria da taxa de correção de *bugs* quando informado a mensagem de erro ou a saída esperada para uma determinada entrada (SOBANIA *et al.*, 2023) e na correção de falhas de segurança quando questionado diretamente da existências da mesma (KHOURY *et al.*, 2023).

Vários estudos relacionados para entender as capacidades e limitações do chatGPT vem sendo propostos (KHOURY *et al.*, 2023) (BORJI, 2023)(SOBANIA *et al.*, 2023) (BANG *et al.*, 2023). Como mostrado pelos autores, o chatGPT apresenta habilidades no processamento de linguagem natural e na geração de código, embora tenha o costume de apresentar respostas incorretas em várias situações. Por mais que esses estudos tragam bons ganhos para a literatura, ainda não esta claro a capacidade do ChatGPT em atividades da engenharia de softwares, como na geração de código, correção de bugs (SOBANIA *et al.*, 2023) ou sua utilidade em outras atividades relacionadas(BANG *et al.*, 2023), muito menos a qualidade das respostas geradas pela ferramenta(SOBANIA *et al.*, 2023). Então, se faz necessários estudos adicionais para ampliar o entendimento do pontência do ChatGPT e suas utilidades na engenharia de software. Nesse estudo é proposto explorar a geração de histórias de usuário e a geração de código utilizando o ChatGPT na versão *24 de Maio*.

Esse estudo é dividido em duas grandes etapas de geração de artefatos por parte do ChatGPT. Na primeira etapa, é explorado a geração de requisitos no formato de histórias de usuário, onde, por meio de um prompt proposto por esse estudo é solicitado a geração de 50 histórias de usuário dentro do contexto de uma aplicação que possibilite o usuário a encontrar um smartphone de maneira fácil. Além disso, foi solicitado para o ChatGPT escrever para cada requisitos: os critérios de aceitação, complexidade, estimativa de esforço e os requisitos relacionados. Na segunda etapa, utilizando uma lista de 8 requisitos selecionados manualmente dos 50 requisitos gerados anteriormente, é explorado a geração de código utilizando NodeJs com Typescript e Prisma, codificando a lista de histórias de usuário selecionadas. No final das

duas etapas é apresentado uma lista com 50 requisitos gerados exclusivamente pelo ChatGPT no primeiro passo e o projeto de uma API, também desenvolvida inteiramente pelo ChatGPT na versão *24 de Maio* utilizando NodeJS com Typescript e Prisma.

O ChatGPT foi capaz de gerar a lista com 50 histórias de usuário dentro do escopo proposto pelo prompt. Para todos os requisitos foram propostos os critérios de aceitação, complexidade e estima de esforço. Porém, somente para alguns requisitos foi proposto a relação entre os requisitos. Para as 8 histórias de usuário selecionadas no primeiro passo, apenas 2 delas foram codificadas pelo ChatGPT sem *bugs*, logo não se fez necessário a interferência humana. Para as outras 6 histórias codificadas pelo ChatGPT que apresentaram *bugs*, somente 1 delas foi corrigida informando a ferramenta a mensagem de erro gerada pela *IDE*, para o restante se fez necessário informar uma prompt com a mensagem detalhada do *bug*, informando o local que o *bug* estava presente.

Foi observado que, por mais que a ferramenta apresente uma boa capacidade de interpretação dos *prompts* em linguagem natural, foram apresentados códigos com *bugs* e erros de sintaxe, além da ferramenta constantemente perder o contexto dos artefatos e códigos do projeto. Por outro lado, os requisitos gerados atendem bem aos parâmetros solicitados e trazem um resultado satisfatório, abrindo a possibilidade de serem utilizados como um ponto de partida para o desenvolvimento do documento de requisitos.

1.1 Objetivos

1.1.1 *Objetivos Geral*

Esse estudo tem por objetivo explorar o ChatGPT quanto a geração de histórias de usuário e geração de código baseado em história de usuário utilizando NodeJS, TypeScript e Prisma

1.1.2 *Objetivos Específicos*

Como objetivos específicos estão:

1. Avaliar a proposta das histórias de usuário escritas de acordo com os parâmetros informados no *prompt*
2. Avaliar o ChatGPT quando ao desenvolvimento de requisitos relacionados
3. Avaliar como o ChatGPT se comporta com prompts em diferentes níveis de descrição:

informando somente a história de usuário, informando a mensagem de erro para um bug, informando uma mensagem descrevendo em detalhes o bug e como corrigi-lo.

2 TRABALHOS RELACIONADOS

2.1 How Secure is Code Generated by ChatGPT?

Em Khoury *et al.* (2023), o autor provê uma análise sobre a segurança do código gerado pelo ChatGPT. No artigo é realizado a geração de 21 exemplos de código cuidadosamente selecionados para explorar vulnerabilidades específicas, os programas foram escritos em 5 linguagens diferentes: Java, C++, C, Python e HTML. Os artefatos gerados são relativamente simples, grande parte possui apenas uma classe e o maior tem 97 linhas de código. O autor propõe analisar manual o código e questionar ao ChatGPT quão seguro é o código produzido. Como resultado, é apresentado que o código gerado está bem abaixo dos parâmetros de segurança aplicáveis a maioria das aplicações. Porém, quando questionado, para muitos casos o ChatGPT foi capaz de reconhecer que o código não estava seguro e propor uma abordagem mais segura se explicitamente solicitado. Ambos os trabalhos possuem uma etapa de geração de código pelo ChatGPT, mas, diferente de Khoury *et al.* (2023), nesse estudo o código é desenvolvido a partir de uma história de usuário. No presente estudo é observado o desempenho do ChatGPT no desenvolvimento de requisitos ou na correção de problemas utilizando diferentes níveis de *prompts*. Além disso, é analisado a escrita de requisitos no padrão história de usuário por parte do ChatGPT.

2.2 ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design

Em White *et al.* (2023b), o autor apresenta um catálogo com 13 padrões de prompt divididos em quatro categorias: Elicitação de Requisitos, Gerador de API de Design e Simulação de Sistema, Qualidade do código e Reestruturação. Cada padrão é proposto para combater um série de problemas comuns nos Modelos de Linguagem de Larga Escala (LLMs) e apresenta uma explicação da estrutura utilizada para documentar os mesmos. É importante destacar que o autor define *prompts* como: "Uma forma de programação usada para instruir um LLM a executar tarefas". O estudo conclui que o ChatGPT apresenta um grande potencial em automatizar tarefas comuns no ciclo de vida do software, mas tem a tendência de gerar saídas incorretas com entusiasmo, então, é necessário uma análise minuciosa dos artefatos gerados, sendo necessário um considerável envolvimento e experiência por parte do usuário. Os padrão de *prompt* podem

ajudar a mitigar problemas, porém se faz necessário desenvolver um catálogo eficaz de *prompts* e como combina-los para alavancar os resultados, além de destacar a necessidade de mais estudos em outros aspectos da engenharia de *prompt*. No presente estudo, se faz uso do padrão de *prompt* intitulado "Padrão de código baseado em princípios" proposto por White *et al.* (2023b).

2.3 An Analysis of the Automatic Bug Fixing

Em Sobania *et al.* (2023), o autor realiza uma análise do desempenho do ChatGPT na correção automática de *bugs*, para isso são utilizados os 40 problemas Python do *QuixBugs*. Além disso, é proposto uma comparação entre os resultados do ChatGPT com o Codex e outras abordagem de reparo automático de programas (*APR - Automated program repair*). O resultado mostra que o ChatGPT tem desempenho semelhante ao Codex, CoCoNut e APR baseada em *Deep Learning*. O estudo também conclui que a taxa de sucesso é aumentada para 31 dos 40 problemas avaliados quando é informado para o ChatGPT a saída esperada para uma determinada entrada ou alguma mensagem de erro existente. Como também visto nos artigos anteriores, o autor deixa o questionamento de se o custo mental necessário para validar as respostas supera as vantagens de utilizar a ferramenta. De maneira semelhante, esse trabalho também propõe analisar o desempenho do ChatGPT no desenvolvimento de código, mas, diferente de Sobania *et al.* (2023) serão utilizadas histórias de usuários para essa avaliação.

2.4 Tabela de comparação entre estudos

Na figura abaixo *Figura 1 - Tabela de comparação entre os artigos* estão listados as principais diferenças entre os estudos relacionados e esse estudo, descartado principalmente um resumo dos objetivos, metodologias e resultados.

Figura 1 – Tabela de comparação entre os artigos

Título	Objetivo	Métodos e Abordagens	Principais Resultados
How Secure is Code Generated by ChatGPT?	Analisar a segurança do código gerado pelo ChatGPT	Geração de 21 exemplos de código em 5 linguagens diferentes	Código gerado abaixo dos parâmetros de segurança em muitos casos
ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design	Apresentar um catálogo com 13 padrões de prompt para melhorar a qualidade do código e automatizar tarefas no ciclo de vida do software	Definição de 13 padrões de prompt em 4 categorias	ChatGPT com potencial em automatizar tarefas, mas tende a gerar saídas incorretas com entusiasmo
An Analysis of the Automatic Bug Fixing	Analisar o desempenho do ChatGPT na correção automática de bugs	Utilização dos 40 problemas Python do QuixBugs para análise	Comparação com o desempenho do Codex e outras abordagens de reparo automático de programas
Esse estudo	Explorar o ChatGPT quanto a escrita de histórias de usuário e geração de código	Utilizar o ChatGPT para geração de 50 requisitos no formato de história de usuário, selecionar 8 delas e desenvolver o código	ChatGPT gerou código com bug para 75% dos requisitos. Porém, a ferramenta foi capaz de resolver os bugs mediante interações adicionais informando a mensagem de erro e/ou explicação explícita do bug

Fonte: Elaborado pelo autor.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão apresentados alguns dos conceitos necessários para o entendimento e desenvolvimento do projeto proposto neste trabalho.

3.1 ChatGPT

De acordo com Shen *et al.* (2023), os Modelo de linguagem em grande escala(LLM - *Large Language Model*) são treinados utilizando uma grande massa de dados com o foco em entender e gerar linguagem natural. Os LLMs são construídos utilizando a arquitetura *Transformer* (VASWANI *et al.*, 2017) e treinados utilizando um *pipeline* de dois estágios para auxiliar no aprendizado eficiente:

1. *Pré-treinamento*: é utilizado uma abordagem de treinamento supervisionado, utilizando uma grande quantidade de dados não anotados, ou seja, os dados não possuem anotações para guiar o aprendizado da LLM.
2. *Ajuste fino*: é utilizado um pequeno conjunto de dados de tarefas específicas. Diferente do passo anterior aqui os dados são anotados, ou seja, os dados possuem anotações para guiar o aprendizado da LLM. O foco é alavancar o conhecimento adquirido no passo de pré-treinamento para executar tarefas específicas conforme o usuário final.

Assim, as LLMs alcançam alta precisão em diversas tarefas com o mínimo de rótulos fornecidos por humanos.

O ChatGPT é um modelo de linguagem em grande escala baseado na arquitetura *Transformer* (VASWANI *et al.*, 2017) desenvolvida pela OpenIA. A ferramenta foi treinada utilizando aprendizagem por reforço humano(RLHF - *Reinforcement Learning from Human Feedback*) seguindo os mesmos métodos do seu antecessor *InstructGPT* com algumas modificações na configuração da coleta de dados. Um modelo inicial foi treinado utilizando ajuste fino supervisionado, onde, treinadores humanos de Inteligência Artificial(IA) forneciam conversas compostas pelo lado do usuário e o assistente de IA. Também foi utilizado o modelo de treinamento por reforço. Para criar o modelo de recompensas foram coletados dados de comparação que consistiam em duas ou mais respostas do próprio chatGPT classificadas manualmente de acordo com a qualidade do conteúdo. Utilizando esse modelo modelo de recompensa foi possível ajustar o modelo do ChatGPT utilizando "Otimização de Política Proximal" como algoritmo de aprendizado por reforço (CHATGPT, 2023). Sendo uma evolução do GPT-3.5, que terminou o

treinamento no início de 2022 utilizando uma infraestrutura de supercomputação de IA do Azure (CHATGPT, 2023). O ChatGPT é treinado para partindo de um *prompt* fornecer uma resposta detalhada, além de utilizar os *feedback* dos usuários para aprender sobre seus pontos fortes e fracos (CHATGPT, 2023).

Como descrito por White *et al.* (2023b) os "*prompts*" são instruções em linguagem natural usadas para contextualizar o LLM e guiar suas respostas, ditando quais informações são importantes e o formato da saída deseja.

O *prompt* oferece uma interface natural e intuitiva para interação com LLM (ZHOU *et al.*, 2022), disponibilizando um forma de programação utilizada para instruir a execução de tarefas (WHITE *et al.*, 2023b). Sendo assim, o *prompt* pode ser entendido como uma propriedade não prevista e que não foi especificamente treinada no modelo, requisitando uma capacidade de adaptação por parte da LLM, (BOMMASANI *et al.*, 2021).

A qualidade do *prompt* impacta diretamente na qualidade da saída gerada, tendo o poder de influenciar as saídas das próximas interações ao fornecer um conjunto de regras e diretrizes que devem ser seguidas (WHITE *et al.*, 2023a), (ZHOU *et al.*, 2022).

3.2 Prisma

Segundo Prisma (2023), o Prisma é um novo tipo de ORM(*Object-Relational Mapping*) de código aberto que pode ser usado com qualquer versão *Node.js* ou aplicação *backend TypeScript*. O principal objetivo do Prisma é aumentar a produtividade dos desenvolvedores ao trabalhar com bancos de dados, fornecendo uma API limpa e segura para realizar consultas ao banco de dados que retornam objetos *JavaScript*.

A configuração principal do Prisma é realizada no arquivo de esquemas, normalmente chamado de "*schema.prisma*" e possui três partes:

1. *Modelo de dados*: especifica as entidades que serão utilizados na aplicação, seus respectivos atributos e a relação com outros modelos. Os modelos são uma peça fundamental no funcionamento do Prisma e são utilizados durante as respectivas chamadas ao *Prisma Client*. Abaixo esta um exemplo de modelo que descreve a entidade de usuário:

```

1 model User {
2   id      Int      @id @default(autoincrement())
3   email   String   @unique
4   name    String?

```

```

5   role      Role      @default(USER)
6   posts    Post []
7   profile  Profile?
8 }

```

2. *Fonte de dados*: especifica os detalhes da fonte de dados que o Prisma deve se conectar. Por exemplo, a *url* e credências de conexão para um banco de dados PostgreSQL
3. *Geradores*: pode existir mais de um gerador. Seu principal objetivo é determinar quais clientes devem ser criados junto com suas respectivas propriedades.

O Prisma pode ser dividido em três partes, sendo elas:

1. *Prisma Client*: É um cliente de banco de dados adaptado ao seu esquema do banco de dados que usa um mecanismo de consulta para executar as chamadas. Por meio do *Prisma Client* que são realizadas as chamadas para criar, atualizar, excluir e ler dados.
2. *Prisma Migrate*: É uma ferramenta para migração de *schemas* que permite manter as informações sempre atualizados de acordo com o arquivo de esquemas do Prisma(*schema.prisma*) comentado no começo da seção. Por exemplo, quando um modelo de dados é criado ou alterado é realizado uma chamada ao Prisma Migrate para sincronizar as informações.
3. *Prisma Studio*: É uma interface visual que possibilita acessar e editar as informações do banco de dados da aplicação por meio do navegador escolhido pelo usuário.

O prisma fornece uma interface abstrata para que o desenvolvedor se preocupe principalmente com o dados que precisa trabalhar e não em montar as chamadas *SQL*

Os seguintes bancos são suportados atualmente: PostgreSQL, MySQL, SQLite, MongoDB, CockroachDB e Microsoft SQL Server. O banco de dados PostgreSQL foi utilizado nesse estudo.

Nesse estudo o *Prisma* foi utilizado junto com o *NodeJs* utilizando a linguagem *typescript*. O Prisma deve ser usado como interface para as consultas no banco de dados.

3.3 Histórias de usuário

Como descrito por Cohn (2004) as histórias de usuário são uma descrição curta e simples que representam uma funcionalidade valiosa para o usuário, onde é aconselhado que seja possível codificar e testar entre um dia e duas semanas, por um ou um par de programadores. As histórias podem ser divididas em 3 partes:

1. A descrição, que será usada para o planejamento
2. Conversas relacionadas para aprofundar os detalhes da estória
3. Testes que descrevem e documentam detalhes e são utilizados para determinar quando uma história está completa

Histórias de usuário geralmente são escritas de acordo com o seguinte modelo: Como um <tipo de usuário>, eu quero <objetivo> para que <motivação> (WIEGERS; BEATTY, 2013). Além disso, deve ser escrita do ponto de vista do usuário e centradas em como podem gerar valor para o mesmo. Segundo Cohn (2004) uma boa história de usuário é foca em seis atributos, sendo esses:

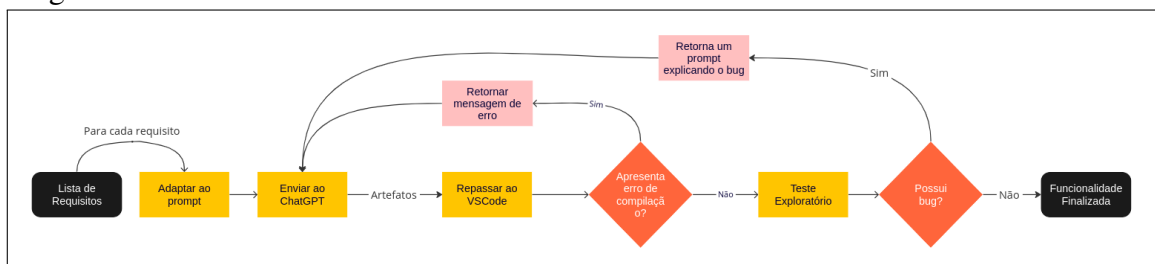
1. Independente: Uma história de usuário não deve depender de outra para ser desenvolvida, pois, acarreta problemas na priorização e planejamento.
2. Negociável: As histórias não são contratos ou requisitos que o software tem que possuir, elas são uma descrição curta das funcionalidades, cujo detalhes são negociados em conversas entre cliente e time de desenvolvimento.
3. Valiosa para usuário ou comprador: É necessário manter em mente a diferença entre usuário e comprador. Talvez uma histórias de usuário não faça sentido para o usuário, como o padrão de escrita da documentação, mas faça sentido para o comprador. O que deve ser evitado são histórias valorizadas apenas pelos desenvolvedores, ou seja, que são focadas em tecnologias ou em vantagens para os desenvolvedores. Solicitar que o cliente ou usuário escreva suas histórias é a melhor maneira de garantir que serão valiosas.
4. Estimável: O desenvolvedor deve ser capaz de estimar quanto tempo é necessário para codificar a história de usuário. Uma das 3 razões para uma história não ser estimável é ser grande demais, já que, para ser estimada é necessário que seja desagregada em histórias menores.
5. Pequena: O tamanho de uma história de usuário importa porque se for pequena demais ou grande demais, não poderá ser usada no planejamento. O tamanho ideal é baseado no time, sua capacidade e tecnologias usadas.
6. Testável: As histórias devem ser testáveis, os teste passando com sucesso significa que a histórias foi desenvolvida com sucesso. Comumente, histórias não testáveis são requisitos não funcionais, como por exemplo: "O usuário deve achar o software fácil de usar".

listings

4 METODOLOGIA

Neste capítulo, será apresentada a organização estrutural das metodologias e técnicas que melhor se adaptam aos objetivos discutidos. A pesquisa em sua natureza objetiva de forma quantitativa levantar dados estatísticos em torno da temática para o sucesso da programação e execução do software. Em resumo o estudo pode ser dividido em duas grandes etapas: Geração das Histórias de usuário e Codificação das Histórias (descrito na *Figura 2 - Fluxograma de desenvolvimento*). Na primeira etapa, será utilizado o *prompt* descrito na seção "4.1 Escrita e análise das histórias de usuário" onde é esperado que o ChatGPT na versão *24 de Maio* retorne uma lista com 50 requisitos, que devem ser analisados e selecionados para a etapa de desenvolvimento. Cada requisito selecionado deve ser adaptado ao *prompt* descrito na seção "4.3 Desenvolvimento das histórias de usuário" e enviado ao ChatGPT para desenvolvimento. Os artefatos propostos pelo ChatGPT devem ser enviados ao VSCode para compilação do código. Caso algum erro seja encontrado, a mensagem deve ser retornada para o ChatGPT para recomençar o ciclo. Caso não seja encontrado erro de compilação, os testes exploratórios devem ser executados para verificar se as regras de negócio são atendidas. Se algum bug for encontrado na execução dos testes exploratórios, uma descrição do problema deve ser retornada ao ChatGPT para o mesmo propor uma solução.

Figura 2 – Fluxo de desenvolvimento



Fonte: Elaborado pelo autor.

4.1 Escrita e análise das histórias de usuário

Para analisar o ChatGPT na versão *24 de Maio* quanto a escrita de histórias de usuário, será utilizado o *prompt* proposto por estudo de acordo com os seguintes parâmetros: (i) Ação: A atividade esperada que o ChatGPT realize; (ii) Contexto: Uma descrição curta da ideia geral do projeto; (iii) Padrão: Solicitado para utilizar o padrão de História de usuário e informado os 6 parâmetros de uma boa história de usuário descritos por Cohn (2004); (iv) Requisitos

relacionados: Foi solicitado que o ChatGPT informe os requisitos que tem relação entre si; (v) Complexidade e estimativa: Foi solicitado que a ferramenta apresente uma estimativa de esforço e complexidade do requisito; (vi)Exemplo: O padrão de saída que é esperado. Com isso, o *prompt* segue o modelo:

```
1
2 Escreva uma lista de 50 requisitos considerando os
   seguintes fatores:
3
4 1. Contexto: A aplicação tem por objetivo encontrar de
   forma fácil o Smartphone que se adeque as necessidades
   de acordo com o perfil do usuário. A ideia é que o
   administrador do sistema possa adicionar/gerenciar os
   dispositivos e suas especificações. Deve ser possível
   que o usuário realize as pesquisas cadastro mas nada alé
   m disso.
5 2. Aplicação desenvolvida: deve ser uma API Rest
6 3. Padrão: Utilizar o padrão de User Stories
7     1. Independente (não se sobrepôr a outra, sendo possí
   vel desenvolvê-las em qualquer ordem).
8     2. Negociável (Devem ser co-criadas pelo cliente e o
   time de desenvolvimento, sendo aptas a responder s
   mudancas quando necessário).
9     3. Valiosa (Precisa entregar o valor para o cliente,
   sendo que inclusive os aspectos técnicos precisam
   ser apresentados de forma que o cliente os considere
   importantes).
10    4. Estimável (A equipe precisa conseguir estimá-la e
   para isso precisa entendê-la).
11    5. Pequena (Boas histórias tendem a serem pequenas,
   reduzindo incertezas e facilitando as estimativas).
12    6. Testável (Permite ser validada e para isso,
   utilizamos critérios de aceitação).
```

- 13 4. Sempre que os requisitos forem relacionados deve estar
informado: "Relacionado com o(s) requisito(s) xxx"
- 14 5. Para cada requisito deve ser informado a complexidade e
sua estimativa.
- 15 6. Use como exemplo a seguinte estória de usuário:
- 16 1. Como um usuário não cadastrado Gostaria de realizar
o cadastro Para que eu possa acessar as
funcionalidades do aplicativo
- 17 Critérios de aceite:
- 18 Um usuário não pode se cadastrar sem preencher
todos os campos obrigatórios (nome, sobrenome,
n mero de telefone, e-mail e senha).
- 19 Um usuário não pode se cadastrar caso digite um
n mero de telefone inválido.
- 20 Um usuário cadastrado não pode se cadastrar
novamente.
- 21 Relacionado com os requisitos 1, 5 e 10
- 22 Complexidade: Media
- 23 Estimativa: 6 pontos

O *prompt* descrito é um proposta desse estudo baseando no estilo de escrita dos padrões de *prompt* propostos por White *et al.* (2023b) e permite avaliar se o ChatGPT escreveu os requisitos no formato de histórias de usuário, seguiu a solicitação de quantidade, se estão de acordo com o contexto especificado e não fugiram do escopo do projeto, a capacidade de relacionar os requisitos e entender sobre a estimativa de complexidade e esforço, além de avaliar se o retorno está de acordo com a estrutura de requisito solicitada.

A quantidade de 50 requisitos é uma escolha desse estudo baseado em experiências anteriores com o chatGPT. A quantidade se mostra suficiente pois permite explorar o chatGPT em mais cenários. A escolha de mais requisitos se tornou inviável dado o tempo para o desenvolvimento do estudo. Menos requisitos podem enviesar os resultados, já que, os dados gerados podem não ser suficiente para uma análise precisa.

4.2 Definição do escopo do projeto

Nessa etapa, utilizando a lista de 50 requisitos propostos na etapa anterior serão selecionados 8 requisitos para desenvolvimento. Para garantir uma maior cobertura das possibilidades reais de desenvolvimento é importante que a lista de requisitos selecionados atenda os seguinte parâmetros: (i) Possuir diferentes níveis de complexidade; (ii) Requisitos que compartilham o mesmo módulo ou entidade da aplicação; (iii) Requisitos fundamentais para o funcionamento da aplicação; (iv) Atendam os parâmetros listados abaixo:

1. Independente (não se sobrepor a outra, sendo possível desenvolvê-las em qualquer ordem).
2. Negociável (Devem ser co-criadas pelo cliente e o time de desenvolvimento, sendo aptas a responder às mudanças quando necessário).
3. Valiosa (Precisa entregar o valor para o cliente, sendo que inclusive os aspectos técnicos precisam ser apresentados de forma que o cliente os considere importantes).
4. Estimável (A equipe precisa conseguir estimá-la e para isso precisa entendê-la).
5. Pequena (Boas histórias tendem a serem pequenas, reduzindo incertezas e facilitando as estimativas).
6. Testável (Permite ser validada e para isso, utilizamos critérios de aceitação).

A quantidade de 8 requisitos foi escolhida por conta do tempo para desenvolvimento do estudo, de modo que, deve ser desenvolvido primeiro os requisitos fundamentais para o funcionamento da aplicação. Posteriormente deve ser considerado a cobertura de diferentes cenários: requisitos com diferentes funcionalidades, que podem compartilhar módulos e que apresentam diferente níveis de complexidade. Os demais requisitos não foram incluídos dado o pouco tempo disponível para desenvolvimento, já que na ordem de escolha não são histórias de usuário fundamentais para o funcionamento da aplicação proposta.

4.3 Desenvolvimento das histórias de usuário

No início da etapa de desenvolvimento, deve ser enviado ao ChatGPT o prompt 1 - *Prompt para preparação do ChatGPT no desenvolvimento dos requisitos* listado abaixo. De modo, a preparar o ChatGPT para o desenvolvimento das 8 histórias de usuário selecionadas no passo anterior. Cada um dos 8 requisitos selecionados na etapa anterior, deve ser adaptado de acordo com o prompt 2 - *Prompt para desenvolvimento do requisito pelo ChatGPT* descrito abaixo e enviado para o ChatGPT na versão 24 de Maio. Como resultado o ChatGPT deve gerar

os artefatos de código que serão inserido no *VSCode* com o objetivo de desenvolver a solução final. A proposta é não interferir no fluxo de desenvolvimento. Caso o código proposto apresente algum erro de compilação, a mensagem de erro gerada será informado ao ChatGPT. Se mesmo assim, após algumas interações, não for capaz de propor uma solução correta será realizado uma interferência direta, seja informando explicitamente o erro ou modificando o código. Para o desenvolvimento das histórias de usuário foi utilizado o seguinte *prompt*:

Código-fonte 1 – Prompt para preparação do ChatGPT no desenvolvimento dos requisitos

```

1 Vou te passar uma lista de requisitos e para cada um gere o
   código de maneira a atender os seguintes par metros:
2 1. Utilize NodeJs, typescript e Prisma
3 2. A aplicação deve ser uma API Rest
4 3. Sempre que você escrever, refatorar ou revisar o código,
   certifique-se de que ele esteja seguindo o PADRÃO DE MÓ
   DULOS para estruturar o projeto. Além disso, escreva um
   script python que cria ou realiza as modificações de
   maneira automática de acordo com o código proposto.

```

Código-fonte 2 – Prompt para desenvolvimento do requisito pelo ChatGPT

```

1 Agora para o seguinte requisito, siga os mesmo par metros:
2 <Requisito>

```

O ponto 3 do prompt 1 - *Prompt para preparação do ChatGPT no desenvolvimento dos requisitos* acima foi adaptado do artigo White *et al.* (2023b). Note que para o prompt 2 - *Prompt para desenvolvimento do requisito pelo ChatGPT*, o parâmetro *<Requisito>* deve ser substituído pela descrição e critérios de aceitação do requisito gerado pelo ChatGPT, seguindo o modelo:

Código-fonte 3 – Exemplo de adaptação do requisito ao prompt 2

```

1 Agora para o seguinte requisito, siga os mesmo par metros:
2 Como um usuário não cadastrado Gostaria de realizar o
   cadastro Para que eu possa acessar as funcionalidades do

```

```
aplicativo
3 Critérios de aceite:
4   Um usuário não pode se cadastrar sem preencher todos os
   campos obrigatórios (nome, sobrenome, n mero de
   telefone, e-mail e senha).
5   Um usuário não pode se cadastrar caso digite um n mero
   de telefone inválido.
6   Um usuário cadastrado não pode se cadastrar novamente.
```

4.4 Teste exploratório no requisito desenvolvido

Para verificar a validade da solução de código proposta na etapa anterior, ao final de cada requisito desenvolvido sem erros de sintaxe, será aplicado a técnica de teste exploratório focando na regra de negócio e nos critérios de aceitação informados no requisito. Não serão escritos casos de teste e o escopo dos testes será baseado na experiência do testador, na proposta do requisito e nos critérios de aceitação escritos pelo ChatGPT durante a definição do requisito. Com isso, asseguramos que o código desenvolvido além de não possuir erros de sintaxe também atende o caminho feliz solicitado no requisito.

5 RESULTADO

Neste capítulo estão presentes os resultados encontrados no decorrer do experimento. A primeira parte apresenta a estrutura e fluxo do projeto proposta pelo ChatGPT na versão *24 de Maio*. A segunda parte contém um explicação dos resultados encontrados agrupado por requisito.

5.1 Requisitos propostos

O ChatGPT na versão *24 de Maio* foi capaz de propor os 50 requisitos solicitados no *prompt*, todos utilizando o padrão de história de usuário e o *template* de saída informado. A maioria dos requisitos estão dentro do escopo proposto e atendem as seguintes funcionalidades: CRUD de usuário e dispositivo, filtro e ordenação de dispositivo, login, notificação, gerar relatórios de vendas, carrinho de compras, cadastrar feedback. Todos os requisitos gerados estão listados no "APÊNDICE A – LISTA DE REQUISITOS PROPOSTA PELO CHATGPT", no final desse estudo. Para todos os requisitos o ChatGPT na versão *24 de Maio* foi capaz de propor os critérios de aceitação, indo de acordo com o *template* de saída solicitado, alguns exemplos são: "[1] Como administrador gostaria de adicionar dispositivo ao sistema", "[2] Como usuário não cadastrado, gostaria de pesquisar smartphones", "[5] Como um usuário cadastrado, gostaria de fazer login no aplicativo" e "[27] Como um usuário cadastrado, gostaria de poder visualizar meu histórico de compras de smartphones". Para os requisitos [2] e [27] é interessante destacar que o ChatGPT não listou todos os parâmetros da entidade relacionada, utilizando "etc." para garantir a cobertura dos demais, o que se mostra como um problema, pois pode gerar subjetividade dos parâmetros que devem ser cobertos. Em contra partida, o requisito [1] cobre todos parâmetros de dispositivo de maneira explícita.

Quanto ao relacionamento entre requisitos, o ChatGPT foi capaz de propor somente para parte deles. Tomando como o exemplo o requisito "[21] Como um usuário cadastrado, gostaria de poder realizar a compra de smartphones diretamente pelo aplicativo" onde foi proposto a relação com os requisitos "[2] Como um usuário não cadastrado, gostaria de pesquisar smartphones" e "[20] Como um usuário cadastrado, gostaria de poder adicionar smartphones ao meu carrinho de compras", respectivamente "pesquisa de um smartphone" e "adicionar smartphone ao carrinho". O relacionamento entre requisito é subjetivo de cada pessoa, podendo levar a diferentes opiniões. É válido ressaltar que os requisitos [2] e [20] podem fazer parte do fluxo de compra de um novo smartphone, já que, em um dos fluxos se faz necessário pesquisar e

adicionar o dispositivo ao carrinho para realizar uma compra.

O ChatGPT na versão *24 de Maio* foi capaz de propor a estimativa de complexidade e esforço para todos os requisitos, por mais que tenha apresentado um comportamento um tanto otimista em suas estimativas. Por exemplo, para o requisito "[45] Como um usuário cadastrado, gostaria de poder receber notificações sobre ofertas especiais, descontos ou promoções de smartphones" que envolve notificações para o usuário a complexidade foi dada como baixo, sendo que para o requisito "[42] Como um usuário cadastrado, gostaria de poder visualizar as avaliações e comentários de outros usuários sobre os smartphones" que envolve visualizar avaliações e comentários a complexidade foi dada como média e a mesma estimativa de esforço foi dada para os 2 requisitos. Segundo Cohn (2004), a estimativa é relativa e depende de cada time, mas é válido ressaltar que o requisito [42] envolve atividades de coleta de informações enquanto o requisito [45] se mostra necessário a utilização de bibliotecas de terceiros.

De maneira geral, os requisitos propostos pela ferramenta apresentam uma taxa de sucesso e podem ser utilizados, por parte do usuário, com um ponto de partida para desenvolver o documento de requisitos da aplicação.

5.2 Estrutura final do projeto

Como descrito na seção 4. *Metodologia*, cada história de usuário da lista de requisitos selecionada na seção 4.2 *Definição do escopo do projeto* foi convertida em um *prompt* e enviado para o ChatGPT codificar. Ao final do experimento, após diversas iterações com o ChatGPT a aplicação possui a seguinte estrutura de pastas apresentada abaixo. Vale ressaltar que a estrutura e conteúdo das pastas foi criado pelo ChatGPT. É observado que existe o arquivo *index.ts*, comumente utilizado em aplicações *NodeJs* e implementa a inicialização da API e o gerenciamento das rotas do projeto. A pasta "Models" onde estão as interfaces de dispositivo e usuário criadas aparentemente para serem utilizadas nas chamadas ao *framework* Prisma. A pasta "Routes" contem os arquivos de dispositivo e usuário que implementam as chamadas ao *framework* Express utilizado, em suma, para criação dos *endpoints* da aplicação que por sua vez fazem a chamada para as respectivas funções implementadas nos arquivos de "Controllers". Por fim, as pastas "Controllers" e "Services" que implementam seus respectivos arquivos de dispositivo e usuário, onde, o ChatGPT definiu para os arquivos de "Controllers" a responsabilidade de validarem as informações recebidas pelos arquivos de rota, checando se são nulas e invocando as funções dos respectivos arquivos de "Service". Por sua vez, os arquivos de *service* realizam as

respectivas chamadas de consolidação ou coleta de dados ao *framework* Prisma.

```
src/  
  controllers/  
    DispositivoController.ts  
    UsuarioController.ts  
  models/  
    Dispositivo.ts  
    Usuario.ts  
  routes/  
    authRoute.ts  
    dispositivosRoute.ts  
    usuariosRoute.ts  
  services/  
    DispositivoService.ts  
    UsuarioService.ts  
  index.ts
```

5.3 Desenvolvimento das Histórias de Usuário

A escolha da lista das 8 histórias de usuário implica no desenvolvimento de uma aplicação considera de complexidade baixa em um contexto que se resume a atividades de CRUD de dispositivo e usuário, filtro e ordenação de dispositivo e login de usuário. O ChatGPT na versão *24 de Maio* se mostrou capaz de entender o contexto de cada história de usuário solicitada para desenvolvimento, mas, isso não significa que foi capaz de codificar corretamente a solução.

Da lista de 8 histórias de usuário, 25% foram desenvolvidas sem bugs (o que totaliza 2 histórias de usuário) e 75% apresentaram bugs na proposta do ChatGPT (o que totaliza 6 histórias de usuário). Das 6 histórias de usuário proposta com bugs, 16,6% foi corrigida informando ao ChatGPT a mensagem de erro gerada pela *IDE* e 83,3% se fez necessário informar uma *prompt* com a mensagem detalhada do *bug*, informando o local que o *bug* estava presente.

Após o desenvolvimento de algumas funcionalidades foi observado que o ChatGPT não foi capaz de manter o contexto da aplicação e apresentou erros de sintaxe, como o nome de funções inexistentes, além de não seguir o padrão de escrita que vinha sendo proposto nos códigos

desenvolvidos anteriormente. Abaixo estão listados os requisitos utilizados no experimento em ordem cronológica de desenvolvimento.

(a) *Cadastro de um novo dispositivo*: O primeiro requisito escolhido para desenvolvimento foi o cadastro de dispositivo, onde, o ChatGPT foi capaz de propor uma solução compilável sem a necessidade de interferência humana e que atendeu corretamente os testes exploratórios. A proposta consistiu na criação das pastas *controllers*, *services*, *models* e *routes* e os respectivos arquivos *DispositivoController.ts*, *DispositivoService.ts*, *Dispositivo.ts* e *dispositivosRoute.ts* que foram explicados na seção anterior. Dado que, foi solicitado a utilização do Prisma como ferramenta ORM o ChatGPT propôs a criação do *model* de dispositivo, informando seus respectivos atributos e tipagem.

(b) *Ver informações de um dispositivo*: Assim como o requisito anterior esse também é considerado simples. O ChatGPT foi capaz de propor uma solução compilável. Foram realizados os testes exploratórios e a solução proposta atende o caminho feliz esperado para o requisito. Porém, foi observado que os arquivos *index.ts* e *dispositivosRoute.ts* estavam com informações conflitantes, ambos realizavam o controle de rotas para dispositivo, ao informar o problema ao ChatGPT utilizando o *prompt* "o arquivo *index.ts* e *dispositivosRoute.ts* estão conflitando" a ferramenta prontamente forneceu uma solução plausível para o problema, o arquivo *dispositivosRoute.ts* ficou responsável pelo controle de rotas para dispositivo, enquanto *index.ts* ficou responsável por agrupar as rotas de todo o projeto. Esse ponto mostra que é necessário a vistoria por parte do usuário, por mais que a solução seja compilável e atenda a regra do requisito foi implementado um inconsistências nos arquivos.

(c) *Filtro de dispositivos*: Esse é um dos requisitos principais da aplicação e é esperado a possibilidade de filtrar os dispositivos cadastrados por seus respectivos atributos. O ChatGPT não foi capaz de propor uma solução compilável sem interferência humana, já que, as soluções apresentaram vários erros de sintaxe, sobrescrevendo funções e erros de tipagem. Após algumas interações informando as mensagens de erro a ferramenta foi capaz de propor uma solução compilável.

Para esse requisito em particular, são observados alguns pontos de melhoria, principalmente porque a solução desenvolvida não permite a busca por qualquer atributo de dispositivo e permite a busca por somente um valor de atributo de dispositivo por vez. Dados isso, primeiro foi solicitado ao ChatGPT por meio de um *prompt* com linguagem informal para implementar a buscar por qualquer atributo de dispositivo. A ferramenta foi capaz de propor uma solução

compilável sem interferência humana e a funcionalidade continua funcionando como esperado. A segunda solicitação também foi feita de maneira informal e requisitado ao ChatGPT para permitir a busca por uma lista de valores no caso de um atributo tipo texto (marca, modelo, sistema operacional, ...) ou uma faixa de valores para atributos do tipo numérico (preço, câmera ou bateria). A primeira solução apresentou erro de tipagem dos dados, mas, após informar a mensagem de erro foi retornado uma solução compilável que atende os testes solicitados.

Pode ser observado a capacidade do ChatGPT em entender um *prompt* informal e realizar as modificações necessária no código.

(d) *Ordenação de dispositivos*: Esse é um requisito que não esta presente na lista gerada pelo ChatGPT e foi criado manualmente, onde a ferramenta não foi capaz de criar uma solução sem interferência humana. É esperado que a solução desenvolvida ordene os dispositivos pesquisados por preço seguindo o parâmetro informado pelo usuário. A primeira solução proposta é compilável e foi proposto a criação de duas novas funções nos arquivos *DispositivoController.ts* e *DispositivoService.ts* e a criação de uma nova rota no arquivo *dispositivosRoute.ts*. Porém, a proposta apresenta erro na lógica, já que, a pesquisa feita anteriormente é desconsiderada e são retornados todos os dispositivos do banco de dados de maneira ordenada. Como o erro é referente à lógica da implementação, foi informado um resumo do problema ao GPT, sendo proposto uma nova função que mescla as funções de filtro e ordenação. Porém, a solução apresenta 2 problemas: A nova função gera duplicidade já que realiza a mesma lógica de uma função já existente e a nova função não realiza o filtro corretamente. Após várias interações sem sucesso, utilizando a última abordagem, foi enviado um *prompt* que propõe diretamente editar a função criado no requisito "(c) Filtro de dispositivos" e seu respectivo código para incorporar a funcionalidade de ordenação, assim, o ChatGPT foi capaz propor a solução que atende os testes exploratórios.bele

(e) *Cadastro de um novo usuário*: Para esse requisito era esperado que o ChatGPT criasse um novo usuário semelhante o que foi feito no requisito "(a) Cadastro de um novo dispositivo". O ChatGPT não foi capaz de propor uma solução compilável sem interferência humana, já que, pulou o passo de criação do *model* de usuário no arquivo *schema.prisma* que é um passo essencial para o Prisma trabalhar com a entidade de usuário. Mesmo informando as mensagens de erro retornadas pelo *nodeJs* o ChatGPT não foi capaz de identificar que os erros eram ocasionados pela falta do *model*. Após algumas interações, o *output* passou a ser repetido. Com isso, foi necessário utilizar uma nova abordagem e foi enviado um *prompt* informando

diretamente do problema: "é necessário criar o model de usuário". Somente após isso é que o ChatGPT foi capaz de criar as dependências e resolver o problema. Outros problemas simples ocorreram. No arquivo *UsuarioController.ts*, era realizada a chamada para uma função não existente no arquivo *UsuarioService.ts*, mas, ao informar a mensagem de erro retornado pelo *nodeJs* uma nova solução que resolve o problema foi proposta. Assim, como citado no artigo Khoury *et al.* (2023) foi observado que o ChatGPT introduziu uma vulnerabilidade séria ao inserir a senha do usuário sem qualquer tipo de criptografia no banco de dados. Isso é considerado uma falha, já que, os invasores que tenham acesso ao banco de dados ou troca de informações entre usuário e servidor podem ter acesso as senhas de maneira explícita. Vale ratificar, que durante o desenvolvimento do requisito "(g) Autenticação de usuário na aplicação" o ChatGPT propôs a utilização da biblioteca *bcrypt* para comparação das senhas de usuário o que abriu caminho para que uma solução que corrige a vulnerabilidade ser proposta indiretamente.

(f) *Editar informações de usuário*: Seguindo o CRUD de usuário, era esperado desse requisito a possibilidade de editar as informações de usuário. O ChatGPT não foi capaz de propor uma solução compilável. O problema se deu na tipagem esperada para o campo ID, no arquivo *usuarioService.ts*. Era esperado o ID como *String* mas no arquivo *usuarioController.ts* era enviado como *Int*. Informando a mensagem de erro do console da IDE o ChatGPT não foi capaz de resolver o problema e após alguma interações sem uma solução foi enviado uma explicação resumida do problema: "o problema é que na função `public async atualizarUsuario(id: string, dadosAtualizados: Partial<Usuario>): Promise<Usuario>` do arquivo *UsuarioService.ts* esta esperando uma string, sendo que o id é um number". Somente assim, a ferramenta foi capaz de corrigir o código. Também, foi observado que a solução implementa um *bug* grave ao permitir que seja possível modificar o *ID* do usuário, na requisição *PUT /usuarios/:id* é possível informar algum id (não utilizado ainda) e esse passa a ser o novo id do usuário. Ao informar diretamente ao ChatGPT do problema ele foi capaz de propor uma solução.

(g) *Autenticação de usuário na aplicação*: Esse é um requisito delicado por envolver a segurança da aplicação. Para esse requisito, era esperada uma implementação da validação de *email* e senha do usuário com o objetivo de realizar o controle de acesso. O ChatGPT foi capaz de propor uma solução compilável após algumas interações. A solução desenvolvida utiliza biblioteca a *NodeJs bcrypt* para criptografia e comparação de senhas e a biblioteca *jsonwebtoken* para geração de token de acesso, o que demonstra uma divergência da solução desenvolvida para o requisito "(e) Cadastro de um novo usuário" onde a senha do usuário não é salva criptografada.

Essa inconsistência gerou um *bug* na aplicação, onde, mesmo o usuário informando *email* e senha corretos não era possível realizar login, já que, a senha informada pelo usuário era criptografada e comparada com a senha não criptografada salva no banco. Esse é um problema delicado porque não gera mensagem de erro no *NodeJs*, requer uma análise do código desenvolvido anteriormente e a solução não necessariamente será proposta nos artefatos desenvolvidos para o requisito atual. Como primeira entrada para o ChatGPT, foi enviada a mensagem de exceção gerada pelo código: "error": "Credenciais inválidas". Mas, a ferramenta não foi capaz de propor uma solução. Como segunda entrada, foi enviada uma explicação resumida do problema: "Verificando a senha do usuário no banco de dados, esta salvo como 123. Parece que no cadastro de usuário a senha não esta sendo salva criptografada". O ChatGPT foi capaz de resolver o problema, criando o cadastro de usuário com senha criptografada, o que mostra a capacidade da ferramenta de interpretar o contexto da entrada feita em texto livre.

(h) *Deletar um dispositivo*: Seguindo o CRUD de dispositivo, era esperado desse requisito a possibilidade de excluir um dispositivo já existente. O ChatGPT foi capaz de propor a criação da função nos arquivos *DispositivoController.ts* e *DispositivoService.ts* de maneira correta, porém, durante a criação da rota para funcionalidade foi proposto a criação de um novo arquivo nomeado *Routes.ts* ainda não existente no projeto. Após algumas interações o ChatGPT não foi capaz de propor uma solução. Somente após indicar diretamente para utilizar o arquivo *DispositivoRoutes.ts* que a ferramenta foi capaz de propor uma solução que atende às especificações do requisito e os testes exploratórios.

5.4 Discussão de resultado

O ChatGPT se mostra como uma ferramenta promissora na capacidade de entender o contexto em linguagem natural de um *prompt* e converter em histórias de usuário e código, podendo se tornar uma ferramenta de apoio para o desenvolvimento de software. Porém, foi observado que é necessário um trabalho de supervisão e manutenção do código gerado, o que não elimina o trabalho manual na utilização do ChatGPT. Além disso, a possibilidade de diálogo com a ferramenta apresenta vantagens e desvantagens consideráveis. Como discutido por Sobania *et al.* (2023) e Khoury *et al.* (2023) a possibilidade de interação com o ChatGPT amplia a qualidade dos resultados. Porém, como também comentado por White *et al.* (2023a) e Zhou *et al.* (2022) a escolha do *prompt* é um passo que afeta diretamente a qualidade das respostas, então se faz necessário a atividade de planejando e escolha dos *prompts*.

6 CONCLUSÃO

O chatGPT na versão *24 de Maio* se mostrou capaz de propor soluções de código compiláveis para os requisitos propostos. De modo geral, os requisitos compiláveis atendem o caminho feliz da regra de negócio. Entre tanto, a ferramenta apresentou sérios problemas relacionados a manter o contexto do projeto. Em várias situações, a proposta de código sobrescreve as funções já existentes nos arquivos, cria arquivos ou funções duplicados desconsiderando os artefatos criados anteriormente e não segue o padrão de escrita do código anterior. Fica como proposta, avaliar a possibilidade de adaptação do ChatGPT nas IDE's de desenvolvimento de código focando principalmente em melhorar a visibilidade dos artefatos que envolvem o projeto.

Como já apresentado no artigo Khoury *et al.* (2023) o ChatGPT introduziu 2 problemas graves de segurança. Durante o desenvolvimento do "requisito (e) Cadastro de um novo usuário", na estrutura proposta pela ferramenta a senha do usuário é salva no banco de dados no formato *plain text*, ou seja, não é utilizado nenhum tipo de criptografia. Esse é um problema de segurança já conhecido e abre espaço para diversas vulnerabilidades, como por exemplo, caso o invasor tenha acesso ao banco de dados ou esteja interceptando a comunicação do cliente com o servidor, ele tem acesso de maneira fácil a senha do usuário. É válido ressaltar que posteriormente durante o desenvolvimento do requisito que consiste em implementar a funcionalidade de login, o ChatGPT se redime do problema e propõe a utilização da biblioteca *nodeJs bcrypt*.

Uma solução factível para o problema de escrita de funções fora do padrão já utilizado no projeto é o desenvolvimento de um *prompt* que fixe um padrão de escrita, ou seja, definir um modelo de escrita para as funções. Como trabalho futuro, fica o estudo de um *prompt* específico voltado para aumentar a capacidade do ChatGPT na versão *24 de Maio* se contextualizar dos arquivos, suas respectivas funções e utilidade para o projeto. Como visto no estudo, boa parte dos problemas estão relacionados a perda de contexto dos artefatos e padrões utilizados no projeto, o que acarreta retornos incorretos por parte do ChatGPT.

Esse estudo apresenta uma análise sobre a geração de código pelo ChatGPT na versão *24 de Maio* e trouxe um estudo empírico sobre o desenvolvimento baseado em histórias de usuário no contexto de uma aplicação simples utilizando *nodeJs*, *typescript* e *Prisma*. Além disso, foi realizado uma análise da utilização do ChatGPT na proposta e escrita de requisitos baseados no modelo de histórias de usuário, onde foi solicitado a seguir 6 diretrizes para garantir a qualidade dos requisitos. Não foi encontrado outro artigo com a mesma proposta de estudo:

utilizar histórias de usuário para desenvolver um projeto do zero, analisando a capacidade do ChatGPT de desenvolver novos requisitos e relacionar as novas funcionalidade com os arquivos já existentes. De modo geral esse estudo destaca que o ChatGPT se mostrou capaz de entender o contexto dos *prompts* mas não significa que foi capaz de apresentar uma proposta de código de maneira correta, sendo necessário interações adicionais para atingir resultados mais satisfatórios. Se mostrando necessário um usuário com olhar atento e experiência prévia em desenvolvimento de software e requisitos para guiar os *prompts* conforme necessário, possibilitando assim atingir um resultado mais assertivo. Sem interferência humana o ChatGPT não se mostrou capaz de desenvolver os requisitos de maneira correta, inserindo erros de sintaxe e *bugs*.

Todo o projeto desenvolvido pelo ChatGPT durante esse estudo está disponível no github pelo https://github.com/leandrobesserra11/API_Developed_ChatGPT

Toda conversa realizado com o ChatGPT esta disponível pelo link https://drive.google.com/file/d/1d_4119ewbyGP7_qiiojMwO8SyZwoqrdo/view?usp=sharing

7 AMEAÇAS A VALIDADE

Vale a pena notar que o ChatGPT está atualmente em desenvolvimento ativo. Versões futuras podem produzir resultados diferentes. Junto a isso, o ChatGPT permite conversar com seus usuário. A utilização de *prompts* diferentes dos desse estudo pode acarretar em resultados diferentes. Os resultados podem variar dependendo da linguagem de programação, requisitos utilizados e quantidade de interações. A classificação dos resultados foi feita manualmente e, portanto, representa a avaliação subjetiva do autor.

REFERÊNCIAS

- BANG, Y.; CAHYAWIJAYA, S.; LEE, N.; DAI, W.; SU, D.; WILIE, B.; LOVENIA, H.; JI, Z.; YU, T.; CHUNG, W. *et al.* A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. **arXiv preprint arXiv:2302.04023**, 2023.
- BOMMASANI, R.; HUDSON, D. A.; ADELI, E.; ALTMAN, R.; ARORA, S.; ARX, S. von; BERNSTEIN, M. S.; BOHG, J.; BOSSELUT, A.; BRUNSKILL, E. *et al.* On the opportunities and risks of foundation models. **arXiv preprint arXiv:2108.07258**, 2021.
- BORJI, A. A categorical archive of chatgpt failures. **arXiv preprint arXiv:2302.03494**, 2023.
- CHATGPT, O. T. **About ChatGPT**. 2023. Disponível em: <https://openai.com/blog/chatgpt>. Acesso em: 01 jun. 2023.
- COHN, M. **User Stories Applied: For Agile Software Development**. [S. l.]: Pearson Education, Inc, 2004. v. 13th.
- KHOURY, R.; AVILA, A. R.; BRUNELLE, J.; CAMARA, B. M. How secure is code generated by chatgpt? **arXiv preprint arXiv:2304.09655**, 2023.
- PRISMA, I. **About Prisma**. 2023. Disponível em: <https://www.prisma.io/docs/concepts>. Acesso em: 25 jun. 2023.
- SHEN, Y.; HEACOCK, L.; ELIAS, J.; HENTEL, K. D.; REIG, B.; SHIH, G.; MOY, L. **ChatGPT and other large language models are double-edged swords**. [S. l.]: Radiological Society of North America, 2023. e230163 p.
- SOBANIA, D.; BRIESCH, M.; HANNA, C.; PETKE, J. An analysis of the automatic bug fixing performance of chatgpt. **arXiv preprint arXiv:2301.08653**, 2023.
- VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.
- WHITE, J.; FU, Q.; HAYS, S.; SANDBORN, M.; OLEA, C.; GILBERT, H.; ELNASHAR, A.; SPENCER-SMITH, J.; SCHMIDT, D. C. A prompt pattern catalog to enhance prompt engineering with chatgpt. **arXiv preprint arXiv:2302.11382**, 2023.
- WHITE, J.; HAYS, S.; FU, Q.; SPENCER-SMITH, J.; SCHMIDT, D. C. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. **arXiv preprint arXiv:2303.07839**, 2023.
- WIEGERS, K.; BEATTY, J. **Software requirements**. [S. l.]: Microsoft Press, 2013. v. 3rd.
- ZHOU, Y.; MURESANU, A. I.; HAN, Z.; PASTER, K.; PITIS, S.; CHAN, H.; BA, J. Large language models are human-level prompt engineers. **arXiv preprint arXiv:2211.01910**, 2022.

APÊNDICE A – LISTA DE REQUISITOS PROPOSTA PELO CHATGPT

1. Como um administrador do sistema, gostaria de poder adicionar dispositivos ao sistema com suas respectivas especificações, para que os usuários possam encontrar smartphones adequados às suas necessidades.

Critérios de aceite:

- a) O administrador deve poder adicionar a Marca, Modelo, Sistema Operacional, Tamanho da Tela, Armazenamento Interno, RAM, Câmera, Bateria, Preço
- b) O administrador deve poder adicionar imagens do dispositivo.
- c) Relacionado com os requisitos 2, 3 e 11.

Complexidade: Baixa

Estimativa: 3 pontos

2. Como um usuário não cadastrado, gostaria de pesquisar smartphones com base em minhas preferências e necessidades, para encontrar um dispositivo adequado. Critérios de aceite:

- a) O usuário deve poder selecionar critérios de pesquisa, como preço, marca, tamanho de tela, capacidade de armazenamento, etc.
- b) O sistema deve retornar uma lista de smartphones que correspondam aos critérios de pesquisa.
- c) Relacionado com os requisitos 1 e 8.

Complexidade: Média

Estimativa: 5 pontos

3. Como um usuário não cadastrado, gostaria de ver informações detalhadas sobre um smartphone específico, para tomar uma decisão informada. Critérios de aceite:

- a) O usuário deve poder selecionar um smartphone da lista de resultados da pesquisa para ver mais informações.
- b) O sistema deve exibir detalhes como especificações técnicas, imagens, avaliações dos usuários, etc.
- c) Relacionado com o requisito 2.

Complexidade: Baixa

Estimativa: 3 pontos

4. Como um usuário não cadastrado, gostaria de realizar o cadastro para acessar as funcionalidades do aplicativo.

Critérios de aceite:

- O usuário não pode se cadastrar sem preencher todos os campos obrigatórios (nome, sobrenome, número de telefone, e-mail e senha).
- O usuário não pode se cadastrar caso digite um número de telefone inválido.
- Um usuário cadastrado não pode se cadastrar novamente.
- Relacionado com os requisitos 1, 5 e 10.

Complexidade: Média

Estimativa: 6 pontos

5. Como um usuário cadastrado, gostaria de fazer login no aplicativo para acessar minha conta.

Critérios de aceite:

- O usuário deve poder inserir seu e-mail e senha para fazer login.
- O sistema deve verificar as credenciais e permitir o acesso apenas a usuários cadastrados.
- Relacionado com o requisito 4.

Complexidade: Baixa

Estimativa: 4 pontos

6. Como um usuário cadastrado, gostaria de poder editar meu perfil para atualizar minhas informações.

Critérios de aceite:

- O usuário deve poder editar campos como nome, sobrenome, número de telefone, e-mail e senha.
- As informações atualizadas devem ser salvas no sistema.
- Relacionado com o requisito 5.

Complexidade: Baixa

Estimativa: 3 pontos

7. Como um usuário cadastrado, gostaria de adicionar smartphones à minha lista de desejos, para acompanhar os dispositivos que me interessam.

Critérios de aceite:

- O usuário deve poder adicionar smartphones à sua lista de desejos.
- O sistema deve armazenar a lista de desejos do usuário.
- Relacionado com o requisito 2.

Complexidade: Média

Estimativa: 5 pontos

8. Como um usuário cadastrado, gostaria de poder comparar as especificações técnicas de diferentes smartphones, para facilitar minha decisão de compra.

Critérios de aceite:

- O usuário deve poder selecionar dois ou mais smartphones para comparar.
- O sistema deve exibir uma tabela comparativa das especificações técnicas dos smartphones selecionados.
- Relacionado com o requisito 2.

Complexidade: Média

Estimativa: 5 pontos

9. Como um usuário cadastrado, gostaria de poder avaliar e deixar comentários sobre os smartphones que adquiri, para compartilhar minha experiência com outros usuários.

Critérios de aceite:

- O usuário deve poder avaliar e deixar comentários para os smartphones que adquiriu.
- O sistema deve exibir as avaliações e comentários dos usuários.
- Relacionado com o requisito 2.

Complexidade: Média

Estimativa: 4 pontos

10. Como um usuário cadastrado, gostaria de receber recomendações personalizadas de smartphones com base no meu perfil e histórico de compras, para encontrar dispositivos adequados às minhas preferências.

Critérios de aceite:

- O sistema deve analisar o perfil e histórico de compras do usuário para fornecer recomendações relevantes.
- As recomendações devem ser exibidas ao usuário de forma clara.
- Relacionado com os requisitos 2 e 7.

Complexidade: Alta

Estimativa: 8 pontos

11. Como um administrador do sistema, gostaria de poder gerenciar as informações dos dispositivos cadastrados, para mantê-las atualizadas.

Critérios de aceite:

- O administrador deve poder editar as informações dos dispositivos, incluindo nome,

marca, modelo, especificações técnicas e imagens.

- As alterações devem ser salvas no sistema.
- Relacionado com o requisito 1.

Complexidade: Média

Estimativa: 5 pontos

12. Como um administrador do sistema, gostaria de poder remover dispositivos do sistema quando não forem mais relevantes, para manter o catálogo atualizado.

Critérios de aceite:

- O administrador deve poder remover dispositivos do sistema.
- Os dispositivos removidos não devem ser exibidos aos usuários.
- Relacionado com o requisito 1.

Complexidade: Baixa

Estimativa: 3 pontos

13. Como um administrador do sistema, gostaria de poder visualizar estatísticas e relatórios sobre o uso da aplicação, para acompanhar o desempenho e identificar possíveis melhorias.

Critérios de aceite:

- O administrador deve ter acesso a gráficos e relatórios com informações como número de usuários, número de dispositivos cadastrados, pesquisas mais frequentes, etc.
- As estatísticas e relatórios devem ser apresentados de forma clara e intuitiva.
- Relacionado com o requisito 1.

Complexidade: Alta

Estimativa: 7 pontos

14. Como um usuário não cadastrado, gostaria de poder entrar em contato com o suporte da aplicação para obter ajuda ou esclarecer dúvidas.

Critérios de aceite:

- O usuário não cadastrado deve ter acesso a um formulário de contato ou chat para enviar suas perguntas ao suporte.
- O suporte deve responder de forma adequada e dentro de um prazo razoável.

Complexidade: Baixa

Estimativa: 4 pontos

15. Como um usuário cadastrado, gostaria de receber notificações sobre ofertas especiais, descontos ou lançamentos de smartphones, para aproveitar oportunidades de compra.

Critérios de aceite:

- O usuário deve poder optar por receber notificações por e-mail ou push.
- As notificações devem ser relevantes e personalizadas.
- Relacionado com o requisito 10.

Complexidade: Média

Estimativa: 5 pontos

16. Como um usuário cadastrado, gostaria de poder compartilhar informações sobre smartphones nas redes sociais, para interagir com outros usuários e obter opiniões.

Critérios de aceite:

- O usuário deve poder compartilhar detalhes de smartphones, como links, especificações técnicas, imagens, etc., nas redes sociais.
- O sistema deve fornecer botões ou opções para facilitar o compartilhamento.
- Relacionado com o requisito 2.

Complexidade: Baixa

Estimativa: 3 pontos

17. Como um usuário cadastrado, gostaria de poder filtrar os resultados da pesquisa por preço, avaliações dos usuários, data de lançamento, etc., para refinar os resultados e encontrar smartphones que atendam melhor às minhas preferências.

Critérios de aceite:

- O usuário deve poder aplicar filtros aos resultados da pesquisa com base em critérios como preço, avaliações dos usuários, data de lançamento, etc.
- O sistema deve atualizar os resultados da pesquisa de acordo com os filtros selecionados.
- Relacionado com o requisito 2.

Complexidade: Média

Estimativa: 5 pontos

18. Como um usuário não cadastrado, gostaria de visualizar as avaliações e comentários de outros usuários sobre smartphones, para obter opiniões e insights antes de tomar uma decisão de compra.

Critérios de aceite:

- O usuário não cadastrado deve poder visualizar as avaliações e comentários dos usuários para os smartphones.

- O sistema deve exibir as avaliações e comentários de forma clara e organizada.
- Relacionado com o requisito 2.

Complexidade: Baixa

Estimativa: 3 pontos

19. Como um usuário não cadastrado, gostaria de ver as melhores ofertas de smartphones em destaque, para aproveitar descontos ou promoções especiais.

Critérios de aceite:

- O usuário não cadastrado deve ver uma seção ou destaque com as melhores ofertas de smartphones.
- As ofertas devem ser atualizadas regularmente e exibidas de forma atraente.

Complexidade: Baixa

Estimativa: 4 pontos

20. Como um usuário cadastrado, gostaria de poder adicionar smartphones ao meu carrinho de compras, para realizar compras futuras ou comparar preços.

Critérios de aceite:

- O usuário deve poder adicionar smartphones ao seu carrinho de compras.
- O sistema deve armazenar o conteúdo do carrinho de compras do usuário.
- Relacionado com o requisito 2.

Complexidade: Baixa

Estimativa: 3 pontos

21. Como um usuário cadastrado, gostaria de poder realizar a compra de smartphones diretamente pelo aplicativo, para uma experiência de compra conveniente e integrada.

Critérios de aceite:

- O usuário deve poder selecionar smartphones, visualizar preços, adicionar ao carrinho de compras e finalizar a compra.
- O sistema deve integrar-se a um serviço de pagamento seguro.
- Relacionado com os requisitos 2 e 20.

Complexidade: Alta

Estimativa: 8 pontos

22. Como um usuário cadastrado, gostaria de poder acompanhar o status do meu pedido de smartphones, para saber quando será entregue.

Critérios de aceite:

- O usuário deve poder acessar informações sobre o status do pedido, como data de entrega estimada, número de rastreamento, etc.
- O sistema deve atualizar o status do pedido à medida que ele progride.
- Relacionado com o requisito 21.

Complexidade: Média

Estimativa: 6 pontos

23. Como um usuário cadastrado, gostaria de poder solicitar a devolução ou troca de smartphones, caso não atendam às minhas expectativas ou estejam com defeito.

Critérios de aceite:

- O usuário deve poder iniciar um processo de devolução ou troca de smartphones.
- O sistema deve fornecer instruções e orientações sobre como proceder.
- Relacionado com o requisito 21.

Complexidade: Média

Estimativa: 5 pontos

24. Como um usuário cadastrado, gostaria de poder entrar em contato com o suporte pós-venda para obter assistência técnica ou resolver problemas relacionados à compra ou uso de smartphones.

Critérios de aceite:

- O usuário deve ter acesso a um suporte pós-venda para obter assistência técnica ou resolver problemas.
- O suporte deve responder de forma adequada e dentro de um prazo razoável.

Complexidade: Baixa

Estimativa: 4 pontos

25. Como um usuário não cadastrado, gostaria de poder acessar informações sobre a empresa por trás do aplicativo, para conhecer sua reputação e confiabilidade.

Critérios de aceite:

- O usuário não cadastrado deve ter acesso a informações sobre a empresa, como história, missão, valores, etc.
- As informações devem ser apresentadas de forma clara e transparente.

Complexidade: Baixa

Estimativa: 3 pontos

26. Como um usuário cadastrado, gostaria de poder salvar meus endereços de entrega para

agilizar o processo de compra.

Critérios de aceite:

- O usuário deve poder adicionar, editar e excluir endereços de entrega.
- Os endereços salvos devem ser exibidos como opções durante o processo de compra.

Complexidade: Baixa

Estimativa: 3 pontos

27. Como um usuário cadastrado, gostaria de poder visualizar meu histórico de compras de smartphones, para referência futura ou controle financeiro.

Critérios de aceite:

- O usuário deve poder acessar seu histórico de compras de smartphones.
- O histórico de compras deve exibir informações como datas, smartphones adquiridos, preços, etc.

Complexidade: Média

Estimativa: 5 pontos

28. Como um usuário cadastrado, gostaria de poder salvar smartphones como favoritos, para acessá-los facilmente no futuro.

Critérios de aceite:

- O usuário deve poder adicionar smartphones aos seus favoritos.
- O sistema deve fornecer uma lista de favoritos acessível ao usuário.

Complexidade: Baixa

Estimativa: 3 pontos

29. Como um usuário cadastrado, gostaria de poder realizar pesquisas avançadas por palavras-chave ou categorias específicas, para encontrar smartphones com base em critérios detalhados.

Critérios de aceite:

- O usuário deve poder realizar pesquisas avançadas por palavras-chave ou categorias específicas, como marca, sistema operacional, tamanho da tela, etc.
- O sistema deve retornar resultados relevantes com base nas pesquisas avançadas.
- Relacionado com o requisito 2.

Complexidade: Média

Estimativa: 5 pontos

30. Como um usuário cadastrado, gostaria de poder receber notificações sobre a disponibili-

dade de smartphones que estão fora de estoque, para ser informado quando eles estiverem disponíveis novamente.

Critérios de aceite:

- O usuário deve poder optar por receber notificações quando smartphones fora de estoque estiverem disponíveis novamente.
- As notificações devem ser enviadas de forma clara e oportuna.

Complexidade: Baixa

Estimativa: 4 pontos

31. Como um usuário cadastrado, gostaria de poder visualizar as informações de contato da empresa por trás do aplicativo, para obter suporte ou informações adicionais.

Critérios de aceite:

- O usuário deve poder visualizar as informações de contato da empresa, como telefone, e-mail, endereço, etc.
- As informações de contato devem ser apresentadas de forma clara e acessível.

Complexidade: Baixa

Estimativa: 3 pontos

32. Como um usuário não cadastrado, gostaria de poder acessar uma seção de perguntas frequentes (FAQ) para obter respostas às minhas dúvidas mais comuns sobre smartphones e o aplicativo.

Critérios de aceite:

- O usuário não cadastrado deve ter acesso a uma seção de perguntas frequentes com respostas relevantes.
- As perguntas frequentes devem ser organizadas e de fácil acesso.

Complexidade: Baixa

Estimativa: 4 pontos

33. Como um usuário não cadastrado, gostaria de poder visualizar as avaliações e comentários dos usuários sobre a empresa por trás do aplicativo, para obter insights e opiniões.

Critérios de aceite:

- O usuário não cadastrado deve poder visualizar as avaliações e comentários dos usuários sobre a empresa.
- As avaliações e comentários devem ser apresentados de forma clara e transparente.

Complexidade: Baixa

Estimativa: 3 pontos

34. Como um usuário cadastrado, gostaria de poder realizar pagamentos parcelados ao comprar smartphones, para facilitar minha capacidade de pagamento.

Critérios de aceite:

- O usuário deve poder selecionar a opção de pagamento parcelado ao comprar smartphones.
- O sistema deve fornecer informações claras sobre as condições de pagamento parcelado.
- Relacionado com o requisito 21.

Complexidade: Alta

Estimativa: 7 pontos

35. Como um administrador do sistema, gostaria de ter uma interface de administração para adicionar, editar e gerenciar dispositivos e suas especificações, para manter a base de dados atualizada.

Critérios de aceite:

- O administrador deve ter acesso a uma interface de administração protegida por autenticação.
- O administrador deve poder adicionar, editar e excluir dispositivos e suas especificações.
- O sistema deve validar os dados inseridos pelo administrador para garantir a consistência dos dados.

Complexidade: Alta

Estimativa: 8 pontos

36. Como um administrador do sistema, gostaria de poder visualizar relatórios e estatísticas sobre o uso do aplicativo, para obter insights sobre o desempenho e o comportamento dos usuários.

Critérios de aceite:

- O administrador deve poder acessar relatórios e estatísticas sobre o uso do aplicativo, como número de usuários cadastrados, número de compras realizadas, dispositivos mais populares, etc.
- Os relatórios e estatísticas devem ser apresentados de forma clara e compreensível.

Complexidade: Média

Estimativa: 6 pontos

37. Como um administrador do sistema, gostaria de poder gerenciar as contas de usuário, para adicionar, editar ou excluir contas quando necessário.

Critérios de aceite:

- O administrador deve ter acesso a uma interface de administração protegida por autenticação.
- O administrador deve poder adicionar, editar e excluir contas de usuário.
- O sistema deve fornecer mecanismos para proteger a privacidade e segurança das informações dos usuários.

Complexidade: Média

Estimativa: 6 pontos

38. Como um administrador do sistema, gostaria de poder definir níveis de acesso e permissões para diferentes funções de usuário, para garantir a segurança e o controle adequados.

Critérios de aceite:

- O administrador deve poder definir níveis de acesso e permissões para diferentes funções de usuário, como administradores, suporte, usuários comuns, etc.
- O sistema deve aplicar corretamente as permissões definidas, limitando o acesso e as ações disponíveis para cada função de usuário.

Complexidade: Média

Estimativa: 5 pontos

39. Como um administrador do sistema, gostaria de poder gerenciar as configurações gerais do aplicativo, como idioma padrão, moeda, configurações de envio, etc., para personalizar a experiência do usuário.

Critérios de aceite:

- O administrador deve ter acesso a uma interface de administração protegida por autenticação.
- O administrador deve poder configurar o idioma padrão, a moeda, as opções de envio, etc.

Complexidade: Baixa

Estimativa: 4 pontos

40. Como um usuário não cadastrado, gostaria de poder ver um guia ou tutorial introdutório sobre como usar o aplicativo, para entender suas funcionalidades e navegação.

Critérios de aceite:

- O usuário não cadastrado deve ter acesso a um guia ou tutorial introdutório.
- O guia ou tutorial deve ser fácil de seguir e apresentar as principais funcionalidades e navegação do aplicativo.

Complexidade: Baixa

Estimativa: 3 pontos

41. Como um usuário cadastrado, gostaria de poder compartilhar informações sobre smartphones ou o aplicativo em redes sociais, para recomendar a outras pessoas.

Critérios de aceite:

- O usuário deve poder compartilhar informações sobre smartphones ou o aplicativo em redes sociais, como Facebook, Twitter, etc.
- O compartilhamento deve ser fácil de realizar e permitir que outras pessoas acessem informações relevantes.

Complexidade: Baixa

Estimativa: 4 pontos

42. Como um usuário cadastrado, gostaria de poder visualizar as avaliações e comentários de outros usuários sobre os smartphones, para obter opiniões e insights adicionais antes de fazer uma compra.

Critérios de aceite:

- O usuário deve poder visualizar as avaliações e comentários de outros usuários sobre os smartphones.
- As avaliações e comentários devem ser apresentados de forma clara e organizada.

Complexidade: Média

Estimativa: 5 pontos

43. Como um usuário cadastrado, gostaria de receber recomendações personalizadas de smartphones com base no meu perfil e histórico de compras, para facilitar minha escolha.

Critérios de aceite:

- O sistema deve analisar o perfil e o histórico de compras do usuário para fazer recomendações personalizadas de smartphones.
- As recomendações devem ser relevantes e apresentadas de forma clara ao usuário.

Complexidade: Alta

Estimativa: 7 pontos

44. Como um usuário não cadastrado, gostaria de poder ver uma lista dos smartphones mais vendidos ou populares, para ter uma ideia dos modelos mais procurados.

Critérios de aceite:

- O usuário não cadastrado deve ter acesso a uma lista dos smartphones mais vendidos ou populares.
- A lista deve ser atualizada regularmente e apresentar informações relevantes sobre os smartphones.

Complexidade: Baixa

Estimativa: 4 pontos

45. Como um usuário cadastrado, gostaria de poder receber notificações sobre ofertas especiais, descontos ou promoções de smartphones, para aproveitar oportunidades de compra.

Critérios de aceite:

- O usuário deve poder optar por receber notificações sobre ofertas especiais, descontos ou promoções de smartphones.
- As notificações devem ser enviadas de forma oportuna e apresentar informações claras sobre as ofertas.

Complexidade: Baixa

Estimativa: 5 pontos

46. Como um usuário cadastrado, gostaria de poder adicionar smartphones ao meu carrinho de compras e retomar a compra posteriormente, para facilitar o processo de compra em etapas.

Critérios de aceite:

- O usuário deve poder adicionar smartphones ao seu carrinho de compras.
- O carrinho de compras deve ser salvo e permitir que o usuário retome a compra posteriormente.

Complexidade: Baixa

Estimativa: 4 pontos

47. Como um usuário cadastrado, gostaria de poder fornecer feedback sobre o aplicativo ou os smartphones, para compartilhar minhas experiências e contribuir para melhorias.

Critérios de aceite:

- O usuário deve ter um meio de fornecer feedback sobre o aplicativo ou os smartphones.

- O sistema deve receber e registrar o feedback fornecido pelo usuário.

Complexidade: Baixa

Estimativa: 3 pontos

48. Como um usuário não cadastrado, gostaria de poder ver informações sobre a política de privacidade e termos de uso do aplicativo, para entender como meus dados são tratados.

Critérios de aceite:

- O usuário não cadastrado deve ter acesso às informações sobre a política de privacidade e termos de uso do aplicativo.
- As informações devem ser apresentadas de forma clara e compreensível.

Complexidade: Baixa

Estimativa: 3 pontos

49. Como um usuário cadastrado, gostaria de poder solicitar suporte ou assistência técnica relacionada a smartphones ou ao aplicativo, para obter ajuda quando necessário.

Critérios de aceite:

- O usuário deve ter acesso a um canal de suporte ou assistência técnica.
- O canal de suporte deve fornecer informações claras sobre como entrar em contato e obter ajuda.

Complexidade: Baixa

Estimativa: 4 pontos

50. Como um administrador do sistema, gostaria de poder gerar relatórios de vendas e faturamento, para obter informações sobre o desempenho financeiro do aplicativo.

Critérios de aceite:

- O administrador deve ter acesso a relatórios de vendas e faturamento.
- Os relatórios devem fornecer informações claras e abrangentes sobre o desempenho financeiro do aplicativo.

Complexidade: Média

Estimativa: 6 pontos