UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Sidney Alex de Amorim Arruda

O uso de modelos de linguagem de grande escala (LLMs) no auxílio ao desenvolvimento e qualidade de software.

RECIFE
2024

Sidney Alex de Amorim Arruda

O uso de modelos de linguagem de grande escala (LLMs) no auxílio ao desenvolvimento e qualidade de software.

Monografia apresentada ao Centro de Informática (CIn) da Universidade Federal de Pernambuco (UFPE), como requisito parcial para conclusão do Curso de Sistemas de Informação, orientada pelo(a) professor(a) Vinicius Cardoso Garcia.

RECIFE

2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Arruda, Sidney Alex de Amorim.
O uso de modelos de linguagem de grande escala (LLMs) no auxílio ao desenvolvimento e qualidade de software. / Sidney Alex de Amorim Arruda. - Recife, 2024.
63 : il., tab.

Orientador(a): Vinicius Cardoso Garcia
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Sistemas de Informação - Bacharelado, 2024.
Inclui referências, apêndices.

1. Inteligência Artificial. 2. LLMs. 3. Planejamento de Software. 4. Engenharia de Requisitos. 5. Engenharia de Software. I. Garcia, Vinicius Cardoso. (Orientação). II. Título.

000 CDD (22.ed.)

Sidney Alex de Amorim Arruda

O uso de modelos de linguagem de grande escala (LLMs) no auxílio ao desenvolvimento e qualidade de software.

Monografia submetida ao corpo docente da Universidade Federal de Pernambuco, defendida e aprovada em 20 de março de 2024.

Banca Examinadora:

_____ Orientador(a)

Prof. Dr. Vinicius Cardoso Garcia

_____ Examinador(a)

Profª Drª Carla Taciana Lima Lourenço Silva

Dedico não só este trabalho, mas toda a minha jornada até aqui a seu Manoel, meu avô, que infelizmente não está mais entre nós. Tenho certeza de que onde ele estiver agora, vai rir, vibrar e me chamar de moleque besta como fez em todas as outras etapas em que esteve ao meu lado.

AGRADECIMENTOS

Agradeço primeiramente à Deus (e a seus auxiliares), por me trazer até aqui e a me segurar em momento difíceis.

Aos meus avós, Rita e Manoel, por todo amor, carinho, incentivo e por ser a base do que eu sou e levo por toda a vida.

À minha mãe, Ana, por todo amor e por ter lutado por mim, me incentivado e ter sempre ficado no meu pé.

À minha companheira, Krislaynne, por todo amor, incentivo, companheirismo e aguentar a minha pessoa todos os dias.

À minha mãe do coração, Evelline, por todo incentivo, cumplicidade e por não ter desistido de mim, mesmo quando até eu já tinha desistido.

Ao meu amigo e orientador, Vinicius Garcia, por ter topado minhas loucuras, por todo apoio, confiança e não ter deixado essa ideia morrer.

À todas as obras de ficção científica, que construíram a minha paixão por computadores e formar o nerd que vos escreve esse texto.

E a todos que passam ou passaram pela minha vida que não foram citados aqui, mas tem a certeza de que são parte importante no que eu sou hoje.

Enfim, amo todos vocês ❤️!

“NÃO ENTRE EM PÂNICO”
(ADAMS, Douglas; O Guia do Mochileiro das Galáxias; 1979)

RESUMO

Nos últimos anos, houve grandes avanços no campo das inteligências artificiais voltadas para o processamento de linguagem natural, recentemente vimos o impacto dos modelos de linguagem de larga escala (LLMs) quando disponível para o grande público. Junto a essa disponibilidade, é observado um efeito colateral, onde parte das pessoas se volta contra o uso das inteligências artificiais pelos mais diversos motivos.

Em paralelo, o desenvolvimento de software desde o início dos tempos enfrenta problemas, desde sua idealização e planejamento, passando por seu desenvolvimento e manutenção. A deficiência ou ausência de artefatos na etapa de planejamento, como levantamento de requisitos, é um dos maiores causadores de problemas apontado pelos programadores.

O objetivo desse trabalho é desenvolver e documentar uma aplicação a servir de exemplo de como o uso de LLMs pode ser benéfico ao planejamento de software e levantamento de requisitos, atuando como um facilitador e criando artefatos que podem ser utilizados como um ponto de partida para iniciar o desenvolvimento de um software.

Palavras-chave: LLMs, Planejamento de software, Levantamento de requisitos, Engenharia de requisitos, IA, Inteligência artificial

ABSTRACT

In recent years, there have been significant advancements in the field of artificial intelligence focused on natural language processing. We have recently seen the impact of large-scale language models (LLMs) when made available to the general public. Along with this availability, a side effect is observed where part of the population turns against the use of artificial intelligence for various reasons.

In parallel, software development has faced problems since its inception, from its ideation and planning, through its development and maintenance. The deficiency or absence of artifacts in the planning stage, such as requirement gathering, is one of the major causes of problems pointed out by programmers.

The objective of this work is to develop and document an application to serve as an example of how the use of LLMs can be beneficial to software planning and requirement gathering, acting as a enabler and creating artifacts that can be used as a starting point to initiate the development of software.

Keywords: LLMs, Software planning, Requirement gathering, Requirement engineering, AI, Artificial intelligence

Sumário

1 Introdução 12
1.1 Objetivos 13
2 Estado da Arte 14
2.1 LLMs 14
2.2 ChatGPT e similares 14
3 Implementação do MVP e Funcionamento 15
3.1 Tecnologias 15
3.1.1 OpenAI 15
3.1.2 Backend 16
3.1.3 Frontend 19
3.2 A solução em funcionamento 20
3.2.1 Prompt Expandido 21
3.2.2 Descrição do Projeto 22
3.2.3 Documento de Requisitos 23
3.2.4 Plano de Projeto 23
3.2.5 Documento de Arquitetura 24
3.2.6 Mapa de Telas 24
3.3 Por que artefatos em Markdown? 26
4 Pesquisa, resultados e futuro 27
4.1 Pesquisa e feedback 27
4.2 Trabalhos futuros 31
5 Bibliografia 32
6 Apêndices 34
6.1 Apêndice A – Descrição do projeto gerada pela IA 34
6.2 Apêndice B - Documento de requisitos gerado por IA 39
6.3 Apêndice C – Plano de projeto gerado por IA 46
6.4 Apêndice D – Documento de arquitetura gerado por IA 55
6.5 Apêndice E – Código da representação visual gerada por IA 63

Lista de Figuras

Figura 1 - ilustração do método Agile - https://jaenhosjsaon.medium.com/agile-and-scrum-methodology-32a14ba3746f 12
Figura 2- [5] - A Whole New Way of Working 13
Figura 3 - Um texto Lorem Ipsum de 8000 palavras convertidas para tokens em https://platform.openai.com/tokenizer. 16
Figura 4 - Estrutura de pastas do backend 17
Figura 5 - exemplo do modelo User representado no schema.prisma 18
Figura 6 - operação Upsert realizada no banco de dados utilizando a API do Prisma 18
Figura 7 - Tela de criação de projeto 20
Figura 8 - Acompanhamento do progresso pelo usuário 20
Figura 9 - Exemplo de prompt expandido 21
Figura 10 - Representação visual do documento de arquitetura gerado pela IA 24
Figura 11 - Mapas de tela, cada item equivale a um épico 25
Figura 12 - Cabeçalho de um mapa de tela com descrição e resumo das funcionalidades. 25
Figura 13 - História de usuário renderizada na tela. 25
Figura 14 - BDD montado por IA 26
Figura 15 - Gráfico que compara o tempo de experiência na área com o hábito de documentar requisitos. 27
Figura 16 - Quantidade total de respostas da pergunta sobre o hábito de escrita de requisitos 28
Figura 17 - gráfico de percepção da importância da etapa de planejamento 29
Figura 18 – Respostas da pergunta: Indique o quão você concorda com a seguinte afirmação: "Consegui aprender a usar o SoFIA rapidamente." 29
Figura 19 - respostas da pergunta: Indique o quão você concorda ou discorda com a seguinte afirmação: "A navegação na plataforma para acessar e gerenciar os recursos do projeto é intuitiva e fácil de usar." 30

Lista de Tabelas

Tabela 1 - preço das entradas e saídas por 1 milhão de tokens em dólar. [19] 15

Tabela de Siglas

Sigla	Significado
LLMs	Large Language Models
SWEBOK	Software Engineering Body of Knowledge
SofIA	Software Engineering Planning Tool with IA
MVP	Minimum Viable Product
MVC	Model-View-Controller
BDD	Behaviour Driven Development
IA	Inteligência Artificial
NLP	Natural Language Processing
PLMs	Pre-trained language models
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
ORM	Object-Relational Mapping
EAP	Estrutura Analítica do Projeto
HTTP	Hypertext Transfer Protocol
CERN	Conseil Européen pour la Recherche Nucléaire
HTML	Hypertext Markup Language

1 Introdução

Requisitos, design, construção, testes e manutenção, essas são algumas das etapas fundamentais no desenvolvimento de software, como aponta o SWEBOK [1], base de conhecimento que é praticamente a "bíblia" da engenharia de software. Se colocarmos esses processos em formato de pirâmide, observamos que as etapas de requisitos e design (falando funcionalmente) representam a base dessa pirâmide e naturalmente, toda pirâmide sem uma base sólida apresenta uma tendência a desenvolver problemas por todo o seu ciclo de existência. Deixando a analogia da pirâmide de lado, a maioria das metodologias adotadas durante o ciclo de desenvolvimento de software adota uma etapa de planejamento antes das tarefas, podemos ter como exemplo o Agile, onde os requisitos sempre são a primeira etapa, como na imagem abaixo.

Figura 1 - ilustração do método Agile - https://jaenhosjsaon.medium.com/agile-and-scrum-methodology-32a14ba3746f

Em 2016, uma pesquisa realizada entrevistando 228 empresas que desenvolvem software apontou que requisitos incompletos ou não-documentados correspondem a 48% dos problemas mais citados, requisitos inconsistentes também aparece na lista com 19% [2].

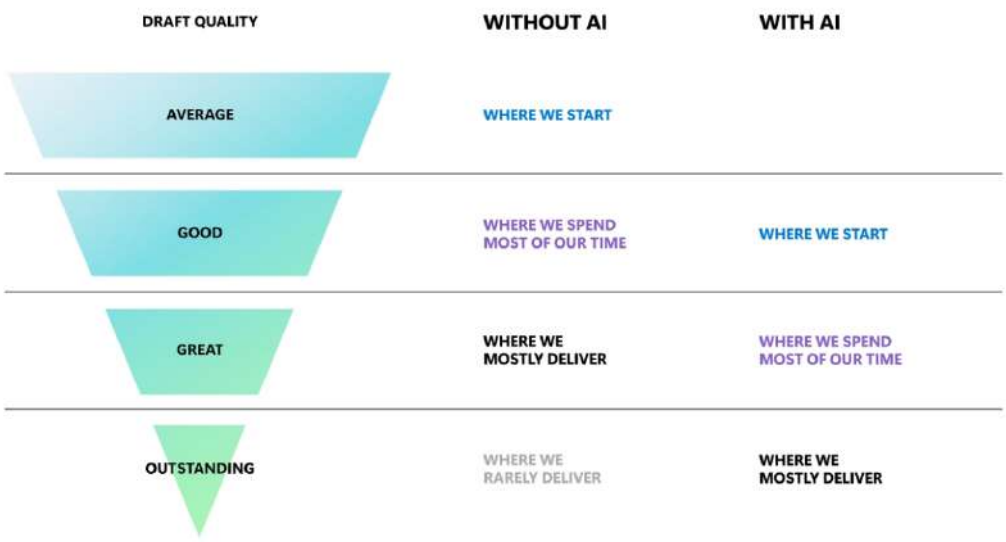
Nos últimos anos, vimos ferramentas promovendo o uso de inteligência artificial para acelerar os seus processos, assim como toda ruptura tecnológica nos últimos anos, naturalmente vem acompanhado de um certo medo uma grande parte das pessoas que temem ficar 100% dependentes da tecnologia ou ter seus empregos substituídos pelas inteligências artificiais futuramente [3]. Isso não é novidade, e boa parte desse medo é alimentado pela ficção como Skynet, Matrix, entre outros. [3], as inteligências artificiais ainda não conseguem pensar de

forma independente ou superar a inteligência humana, mas pode ser utilizada para aumentar a produtividade e poupar tempo. [3]

É fundamental reconhecer que a IA, pode e deve ser utilizada como uma ferramenta pelo ser humano. Seu uso pode trazer riscos, mas também pode trazer benefícios, a perspectiva fundamental a ser seguida é não encarar a relação "Humano-IA" como uma competição, e sim como uma colaboração, no qual a IA traz mais benefícios ao ser utilizada em conjunto com o ser humano. [4].

Seguindo na linha da colaboração, uma forma bastante comum de se utilizar a IA atualmente em ferramentas, é permitir que a inteligência artificial produza um escopo inicial da ideia, baseado no comando do ser humano, e assim as pessoas conseguem trabalhar apenas aprimorando o trabalho que a IA produziu. O infográfico da Microsoft sobre o uso de IA nas suas ferramentas ilustra bem a ideia [5].

Figura 2- [5] - A Whole New Way of Working



1.1 Objetivos

Visando apresentar um bom exemplo de como a colaboração entre a inteligência artificial e a humana podem funcionar, este estudo tem como propósito a apresentação do SofIA (Sofware Engineering Planning Tool With IA), um MVP (Produto Mínimo Viável) cuja finalidade é utilizar inteligência artificial para produzir artefatos baseados na ideia que o usuário informar, e que podem servir como um ponto de partida no processo de desenvolvimento de software. Adicionalmente, além da introdução ao MVP, este trabalho detalhará de forma sistemática as fases que foram conduzidas durante o desenvolvimento da solução e apresentará resultados de uma pesquisa de feedback dos usuários associada.

2 Estado da Arte

A linguagem executa um papel fundamental na comunicação e na autoexpressão dos seres humanos e em suas interações com máquinas. A necessidade de modelos generalizados surge da crescente demanda por máquinas que executem tarefas linguísticas complexas como tradução, resumo, busca, geração de texto e etc. [6]

O desenvolvimento e pesquisa desses modelos, trouxeram algo revolucionário, habilitando a criação de LLMs que possuem a capacidade de realizar várias tarefas com o nível de performance similar ao de um ser humano. [6]

2.1 LLMs

Os LLMs podem ser considerados um avanço no desenvolvimento de técnicas voltadas ao processamento de linguagem natural (NLP), esses modelos são treinados utilizando quantidades massivas de texto e são capazes de gerar conteúdo escrito como um ser humano, responder questões e realizar outras tarefas relacionadas com uma alta precisão. [7]

LLMs evoluem dos chamados PLMs (pre-trained language models), enquanto a modelagem de linguagem convencional (LM) treina modelos em uma configuração que o limita em uma tarefa específica, os PLMs são treinados utilizando uma configuração onde existe uma "auto-supervisão" do conteúdo, que separa partes que podem ser compartilhadas entre várias tarefas de processamento de linguagem natural. Após o "fine-tuning" para adaptar o PLM para uma tarefa específica, é aumentado consideravelmente os parâmetros do modelo e a quantidade de dados que será utilizado para treinamento, concluindo a transição de PLMs para LLMs. [6]

2.2 ChatGPT e similares

Em novembro de 2022, o ChatGPT, da OpenAI, foi lançado ao público inicialmente com o modelo GPT-3, o sucesso deu início a uma corrida pela adoção do público para o uso de IA no seu cotidiano e ao longo de todo o ano de 2023 surgiram diversas alternativas ao ChatGPT [8], como o Copilot [9], da Microsoft (que utiliza o GPT como base para a sua ferramenta), o LLama da Meta [11], Bard (e sua evolução, o Gemini[10]) do Google e diversas outras alternativas provenientes de iniciativas Open Source e de empresas menores.

3 Implementação do MVP e Funcionamento

3.1 Tecnologias

Para a construção do MVP, diversas tecnologias foram utilizadas, especialmente tecnologias de desenvolvimento de aplicações web modernas. Foi utilizado a plataforma da OpenAI [15] como serviço externo de LLM, um backend escrito em TypeScript [18] e Express [14], banco de dados MongoDB [17] e Nuxt 3 [14] como frontend. Uma parte da solução foi hospedada no Railway [16] e outra na Vercel [12]. Todas as ferramentas foram escolhidas por conta do conhecimento prévio, facilidade e velocidade na implementação do MVP.

3.1.1 OpenAI

A escolha de utilizar o serviço da OpenAI foi tomada pela praticidade de utilizar a plataforma, pela sua documentação e suporte oficial para as tecnologias escolhidas para a construção do restante da solução. O modelo escolhido para o uso na solução, foi o gpt-3.5-turbo-0125, a escolha desse modelo, no lugar do mais recente, o gpt-4-turbo, foi tomada por uma razão custo x benefício, já que nos primeiros testes o gpt-3.5-turbo-0125 obteve um resultado satisfatório e tem o custo por token de 20x a 120x mais barato que o modelo mais recente [19], pelo menos até o presente momento.

Modelo	**Input/1M tokens**	**Output/1M tokens**
gpt-4-0125-preview	$10.00	$30.00
gpt-4	$30.00	$60.00
gpt-4-32k	$60.00	$120.00
gpt-3.5-turbo-0125	$0.50	$1.50

Tabela 1 - preço das entradas e saídas por 1 milhão de tokens em dólar. [19]

O consumo do serviço é possível de ser realizado por chamadas HTTP, a escolha para a solução foi utilizar a biblioteca oficial da OpenAI para Node.js [20]. O modelo escolhido

aceita uma entrada de até 16.385 tokens por chamada, o que seria equivalente a um texto de aproximadamente 8000 palavras.

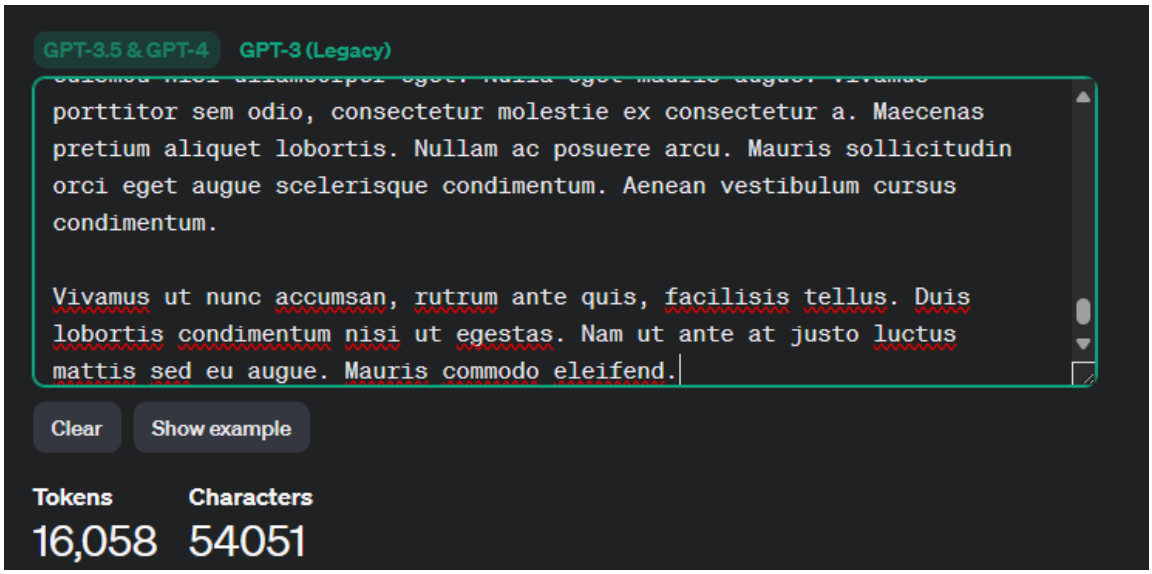


Figura 3 - Um texto Lorem Ipsum de 8000 palavras convertidas para tokens em https://platform.openai.com/tokenizer.

A conversão de palavras para tokens é uma medida aproximada e possui diversos fatores envolvidos no cálculo, a comparação foi utilizada apenas de modo figurativo com o objetivo de ter um parâmetro para exemplificar o tamanho do texto que pode ser inserido.

O recurso utilizado para gerar texto no serviço da OpenAI, é chamado "Chat Completions API", ou API de conclusão de bate-papo em uma tradução literal para o português. Esse recurso permite enviar uma lista com objetos que simulem um bate-papo com a API, facilitando o agrupamento de mensagens que serão enviadas ao serviço para obter um texto gerado, ou até mesmo retroalimentar a conversa com um texto já retornado previamente pelo serviço.

3.1.2 Backend

3.1.2.1 Arquitetura

O backend da solução foi escrito em Node.js, utilizando o TypeScript como linguagem de programação e a biblioteca ExpressJS como servidor web. A arquitetura escolhida inicialmente foi MVC, porém durante a etapa de desenvolvimento, alguns desafios foram encontrados, o que resultou na adição de algumas camadas extras, como casos de uso (comum na arquitetura limpa) e recursos, com o objetivo de padronizar e agilizar o processo. O estado atual representa uma mistura de algumas arquiteturas, o serviço de IA utilizado durante o desenvolvimento (OpenAI) passou por diversas mudanças e avanços nesse período, resultando na adição de alguns *"workarounds"* para garantir o funcionamento integral da solução. Em

consonância a escolha de uma arquitetura específica seria uma limitação e poderia potencializar problemas futuros.

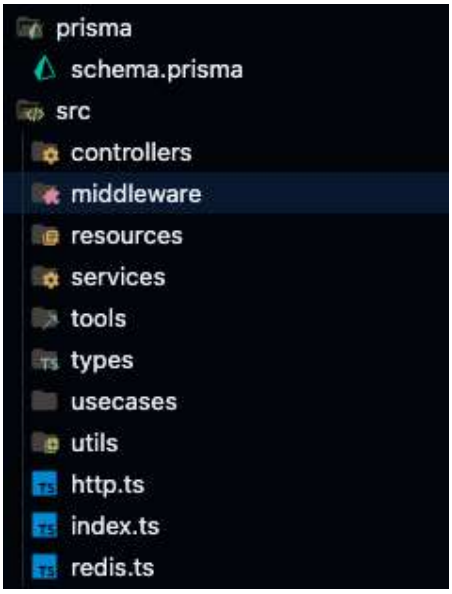


Figura 4 - Estrutura de pastas do backend

3.1.2.2 Banco de Dados e ORM

Para armazenar os dados da aplicação, foi escolhido o MongoDB, um banco de dados não relacional orientado a documentos, que permite uma grande flexibilidade na definição dos esquemas de dados e uma fácil escalabilidade. Para facilitar a interação com o MongoDB e garantir a consistência dos dados, foi utilizado o Prisma [21], um ORM (Object-Relational Mapping) que abstrai o acesso ao banco de dados e oferece uma interface simples e declarativa para definir os modelos de dados e as relações entre eles. Em algumas arquiteturas mais simples, o Prisma é capaz de substituir algumas camadas como model, repository e service.

O Prisma foi escolhido por diversas vantagens que ele apresenta em relação a outras bibliotecas construídas para gerenciar o MongoDB, como o Mongoose ou a biblioteca padrão do Mongo para JavaScript. Uma delas é a sua capacidade de gerar automaticamente o esquema do banco de dados a partir da definição dos modelos no Prisma Schema, um arquivo que especifica os atributos e tipos de cada entidade. Outra vantagem é a sua API de consulta, que permite realizar operações CRUD (Create, Read, Update, Delete) de forma simples e segura, evitando injeções de código malicioso. Além disso, o Prisma oferece recursos como geração de tipos automática baseada no schema, validação de dados, filtragem avançada, paginação e suporte a vários bancos de dados.

```
model User {
  id         String      @id @default(auto()) @map("_id") @db.ObjectId
  email      String      @unique
  userSub    String      @unique
  name       String
  avatarUrl  String?
  credits    Int         @default(10)
  createdAt  DateTime    @default(now())
  updatedAt  DateTime    @updatedAt
}
```

Figura 5 - exemplo do modelo User representado no schema.prisma

O uso do Prisma com o MongoDB contribuiu para o desenvolvimento ágil e eficiente da solução, pois permitiu uma maior produtividade, confiabilidade no acesso aos dados, melhor organização e manutenção do código, pois reduziu a complexidade e a repetição das operações com o banco de dados.

Figura 6 - operação Upsert realizada no banco de dados utilizando a API do Prisma

```
const user = await prisma.user.upsert({
  update: {},
  create: {
    email: email as string,
    name: name as string,
    userSub: userSub as string,
    credits: 10,
  },
  where: {
    userSub,
  },
});
```

3.1.2.3 Autenticação e Autorização

Para garantir a segurança e a privacidade dos dados dos clientes, a solução desenvolvida utiliza um mecanismo de autenticação utilizando e-mail e senha complementado por um mecanismo de login social baseado no OAuth 2.0. Esse mecanismo permite que os usuários se identifiquem usando suas contas sociais já existentes, sem necessidade de criar ou memorizar senhas adicionais. Apesar desse método de autenticação oferecer diversas informações do

usuário, as únicas que precisamos para não comprometer a privacidade do usuário se resumem a e-mail e nome do usuário.

A autorização é feita por meio de cookies, que mantêm o estado de autenticação do usuário entre as requisições. No momento que o usuário faz o login, é gerado um token que é enviado ao navegador do usuário como um cookie seguro e criptografado. Esse cookie é usado para verificar se o usuário está autenticado e autorizado a acessar determinadas rotas da aplicação. Se o cookie não for válido ou expirar, o usuário é redirecionado para a página de login novamente. Dessa forma, a solução garante que apenas os usuários autenticados e autorizados possam interagir com os dados da aplicação, e garantir que cada usuário acesse os seus dados especificamente.

3.1.3 Frontend

3.1.3.1 Monorepo

O Turborepo [22] é uma ferramenta que facilita a gestão de múltiplos projetos em um único repositório, técnica conhecida como monorepo [23]. Ele utiliza um sistema inteligente de cache e dependências para acelerar os processos de desenvolvimento, reduzindo o tempo e aumentando a produtividade. Além disso, ele facilita a organização e o compartilhamento de código entre os projetos que estão dentro do repositório, evitando código repetido entre as aplicações e pacotes.

3.1.3.2 Vue e Nuxt

O framework web escolhido para o desenvolvimento do frontend foi o Nuxt. Esta é uma versão atualizada do Nuxt.js, um framework baseado em Vue.js que facilita o desenvolvimento de aplicações universais, isto é, aplicações que podem ser renderizadas tanto no lado do cliente quanto no lado do servidor sem prejudicar a experiência no desenvolvimento ou adicionar camadas desnecessárias ao código.

Combinando com o tópico anterior, monorepos, o Nuxt possui uma arquitetura modular, composta por um sistema de plugins, que permitem adicionar funcionalidades extras agindo diretamente dentro do framework, um sistema de módulos, que são extensões do próprio framework, que adicionam recursos como roteamento automático, geração estática, internacionalização, entre outros, e um sistema de camadas, onde é possível unificar várias aplicações em apenas uma, o que facilita o isolamento das funcionalidades e reduz a quantidade de código desnecessária. Com isso, é possível configurar a aplicação de acordo com as necessidades específicas do projeto, sem perder as vantagens de uma estrutura padronizada.

3.2 A solução em funcionamento

Ao entrar na aplicação e fazer login, o usuário é apresentado a uma tela inicial onde é possível criar e gerenciar grupos ou projetos. A jornada de criação com o uso colaborativo da inteligência artificial é na criação de um projeto, onde o usuário define o título e vai informar a ideia que será a base da criação, esse procedimento tem inicio da tela de criação de projeto, apresentada na figura 7.

Figura 7 - Tela de criação de projeto

Ao confirmar, a informação inserida pelo usuário será enviada para o banco de dados, e a API começara o processamento junto ao serviço de inteligência artificial. O usuário tem a possibilidade de acompanhar o progresso do processamento enquanto navega no projeto, como na figura 8.

Figura 8 - Acompanhamento do progresso pelo usuário

O projeto utilizado como exemplo possui a seguinte descrição: “Preciso de um conjunto de aplicativos para o controle de um restaurante, um para a cozinha, outro para o caixa e um para o entregador.”

3.2.1 Prompt Expandido

O prompt expandido possui o objetivo de utilizar a inteligência artificial para adicionar um contexto ao que o usuário inseriu que pode servir como um guia de desenvolvimento ou a base para uma elucidação de requisitos, como funcionalidades, a previsão de um contexto de uso e mais detalhes que podem ser uteis ao que o usuário inseriu previamente. Na API o serviço de IA é chamado com uma mistura de um texto relativo ao recurso e a inserção do usuário, no recurso de prompt expandido, um dos prompts é o seguinte: *“Complemente e melhore e devolva em formato markdown a seguinte descrição utilizando o contexto do mercado e desenvolvendo melhor itens como funcionalidades, fluxos e sugestões: [[context]]”,* o *[[context]]* é substituído pelo texto inserido pelo usuário, e a resposta retornada do serviço de IA é armazenada no banco de dados e será utilizada para montar todos os recursos seguintes. O exemplo completo gerado a partir da nossa entrada de testes pode ser visto na figura 9.

Sistema Integrado para Controle de Restaurante

Contexto

No mercado de alimentos e bebidas, é essencial contar com ferramentas que auxiliem no gerenciamento eficiente das operações de um restaurante. Para isso, é fundamental ter aplicativos específicos para a cozinha, o caixa e o entregador, que sejam integrados para fornecer informações em tempo real e facilitar a tomada de decisões.

Funcionalidades do Aplicativo para a Cozinha

- **Gestão de Pedidos**: permite o acompanhamento dos pedidos em tempo real, com notificações de novos pedidos e status de preparo.
- **Controle de Estoque**: possibilita o controle do estoque de ingredientes e a geração automática de alertas de reposição.
- **Gerenciamento de Receitas**: oferece acesso a todas as receitas dos pratos do restaurante, com instruções detalhadas e registro de modificações.
- **Comanda Eletrônica**: permite a visualização das comandas dos clientes, facilitando a preparação e a entrega dos pedidos.

Funcionalidades do Aplicativo para o Caixa

- **Registro de Pedidos**: oferece uma interface amigável para a realização de pedidos, incluindo a seleção de mesas e a personalização dos pedidos.
- **Controle de Pagamentos**: possibilita o registro de pagamentos em dinheiro, cartão e outras formas, com opções de divisão de contas e emissão de recibos.
- **Relatórios de Vendas**: gera relatórios detalhados sobre as vendas, permitindo análises de desempenho e comparações de períodos.
- **Integração com Sistema de Reservas**: permite o acesso às reservas feitas no restaurante, possibilitando o gerenciamento das mesas e a organização dos atendimentos.

Funcionalidades do Aplicativo para o Entregador

- **Roteirização de Entregas**: calcula a melhor rota para as entregas, considerando a localização dos clientes e a otimização do tempo.
- **Acompanhamento de Pedidos**: fornece informações sobre os pedidos em andamento, incluindo os itens e as instruções de entrega.
- **Assinatura Eletrônica**: permite a coleta de assinaturas dos clientes no momento da entrega, para comprovação do recebimento.
- **Integração com GPS**: facilita a localização dos endereços de entrega, reduzindo o tempo de deslocamento e minimizando erros de rota.

Além das funcionalidades mencionadas, sugere-se a implementação de uma interface de gestão centralizada, que permita a integração e a visualização das informações dos três aplicativos, proporcionando uma visão unificada do restaurante e facilitando a tomada de decisões estratégicas.

Figura 9 - Exemplo de prompt expandido

Notando que o usuário não inseriu nenhuma funcionalidade específica, seria um desafio construir uma documentação de requisitos sem funcionalidades definidas e a inteligência artificial contornou esse problema.

3.2.2 Descrição do Projeto

A descrição do projeto utiliza o artefato gerado anteriormente para definir conceitos importantes antes dos requisitos, como limitações, escopo, possíveis entregas e uma estrutura analítica. Abaixo um exemplo da estrutura utilizada nesse artefato:

"export const project_scope = ({ context }: { context: string }) => ({
header: `${context} \nVocê é um engenheiro de software encarregado de desenvolver um novo sistema. Sua tarefa é elaborar um Documento de Escopo do Projeto com base no contexto fornecido acima em markdown. Irei adicionar cada etapa em uma mensagem diferente, quero me responda em markdown.`,
steps: [
`1. Descrição do Projeto:
[...]`,
`2. Declaração do Escopo:
[...]`,
`3. Critérios de Aceitação:
[...]`,
`4. Limites do Projeto:
[...]`,
`5. Entregas:
[em tabelas]
[...]`,
`6. Restrições e Assunções:
[...]`,
`7. Estrutura Analítica do Projeto (EAP):
[...]`,
],
});"

A comunicação com o serviço de IA é feita utilizando o prompt principal misturado com o prompt expandido gerado na etapa anterior. Primeiramente é enviado ao chat com a IA um contexto de uma possível persona a ser assumida, no caso um engenheiro de software e a forma de retorno, e em seguida executa a consulta das etapas posteriores de forma iterativa, representando o título de cada sessão do documento, resultando no exemplo encontrado no Apêndice A.

3.2.3 Documento de Requisitos

O documento de requisitos é um artefato fundamental para o desenvolvimento de software, pois define o que o sistema deve fazer e como deve se comportar sob diferentes condições. De acordo com o SWEBOK, o documento de requisitos deve ser claro, consistente, completo, preciso, verificável e rastreável. Além disso, o documento de requisitos deve especificar os requisitos de forma modular, abstrata e concisa, evitando ambiguidades e detalhes desnecessários.

Com a implementação similar ao artefato anterior, o documento de requisitos tenta emular e criar uma base como descrito no SWEBOK, com requisitos funcionais, requisitos não funcionais, requisitos de interface, atributos de qualidade e restrições.

Em cada um dos tópicos é gerada uma lista com um título e uma descrição detalhada de cada requisito, a saída é um documento de texto utilizando o formato Markdown. Exemplo:

Requisitos Funcionais:
*1. **Gestão de Pedidos:***
- O sistema deve permitir o acompanhamento em tempo real dos pedidos, com notificações de novos pedidos e atualização do status de preparo.

O exemplo completo do documento de requisitos gerado utilizando o prompt de testes pode ser visto no Apêndice B.

3.2.4 Plano de Projeto

Um plano de projeto é um documento que descreve os objetivos, o escopo, a estrutura, as atividades, os recursos, os riscos e as entregas de um projeto de software. O plano de projeto serve como um guia para todo o ciclo de vida do projeto, desde o início até o encerramento, e ajuda a monitorar e controlar o progresso e a qualidade do projeto. Um bom plano de projeto

deve ser claro, completo, consistente, realista e atualizado conforme as mudanças ocorrem no projeto.

Novamente, sua implementação e estrutura são similares ao artefato anterior, sendo um documento bem maior, sendo composto pelos objetivos do projeto, o escopo do projeto, uma tabela representando o cronograma de implementação, uma área descrevendo os recursos necessários, o orçamento, os riscos, comunicação, qualidade, stakeholders e a gestão de mudanças. O exemplo completo do plano de projeto gerado utilizando o prompt de testes pode ser visto no Apêndice C.

3.2.5 Documento de Arquitetura

O documento de arquitetura possui uma visão geral do sistema, seus requisitos, suas restrições, seus componentes, suas interfaces, seus padrões, seus mecanismos e suas propriedades de qualidade. O documento de arquitetura define os objetivos e os princípios arquiteturais que norteiam o projeto, facilitando a comunicação entre os envolvidos, a manutenção e a evolução do sistema.

A implementação é similar aos anteriores, mas aqui dividida em duas partes: A primeira sendo o documento de arquitetura em si, com elementos como a visão geral da arquitetura, diagrama de componentes, visão da implementação, entre outros. E a segunda etapa é tentar montar um diagrama visual com o documento de arquitetura devolvido pela IA. O exemplo completo do documento de arquitetura gerado utilizando o prompt de testes pode ser visto no Apêndice D.

Complementando o documento de arquitetura, é gerado em paralelo a tentativa de uma representação visual da arquitetura, utilizando a biblioteca de diagramação Mermaid [24]. O código gerado para a representação visual no nosso projeto de testes pode ser visto no Apêndice E, podendo ser modificado pelo usuário para ser utilizado em outras ferramentas que suportem a biblioteca Mermaid.

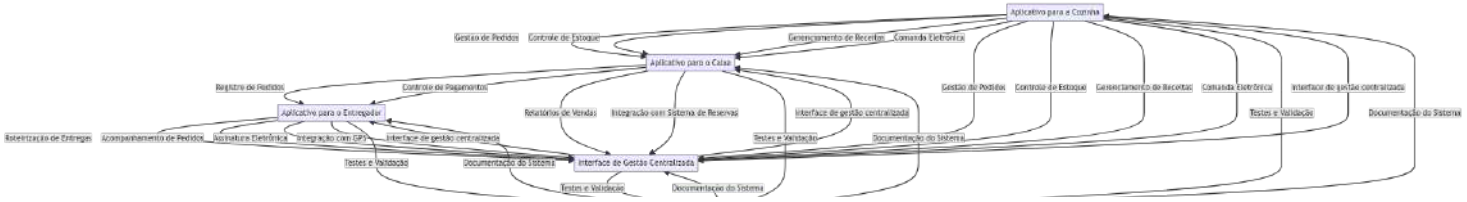


Figura 10 - Representação visual do documento de arquitetura gerado pela IA

3.2.6 Mapa de Telas

Os mapas de telas, são gerados paralelamente uns aos outros, e no ponto de vista da agilidade e engenharia de software são equivalentes aos épicos. Cada mapa de tela possui uma

descrição do que deve ser feito, uma lista com histórias de usuário e um BDD com alguns casos de teste. Os mapas de telas são apresentados ao usuário como botões organizados e de fácil visualização, como na figura 11.

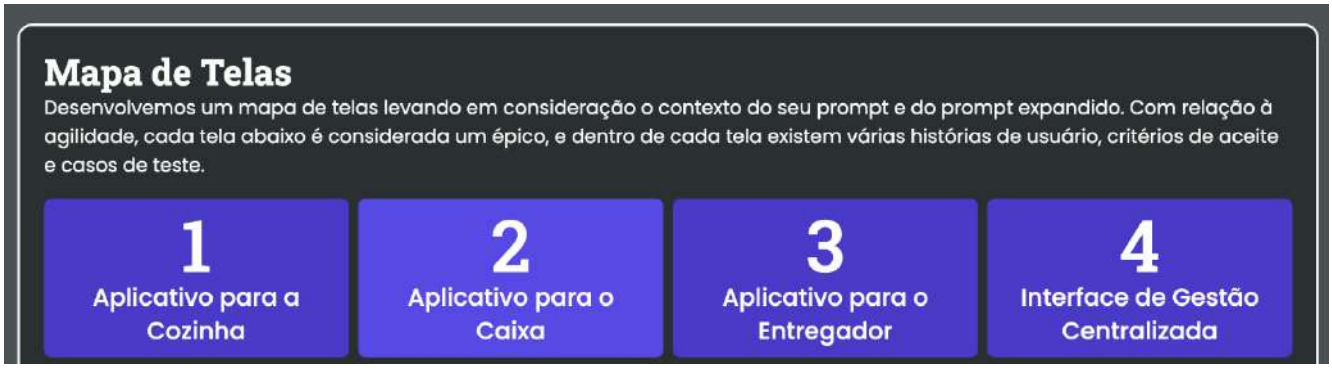


Figura 11 - Mapas de tela, cada item equivale a um épico

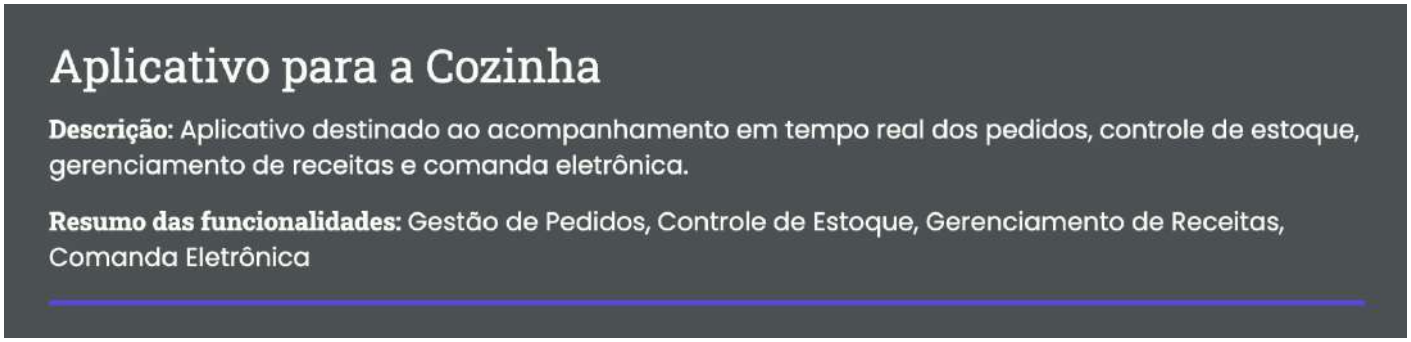


Figura 12 - Cabeçalho de um mapa de tela com descrição e resumo das funcionalidades.

Cada história de usuário é composta por um título, descrição, critério de aceite e uma lista de subtarefas, existe a possibilidade para o usuário efetuar o controle dessas tarefas, sinalizando como em andamento ou concluído. O título e descrição são apresentados como o cabeçalho do mapa, como na figura 12. As histórias de usuário vem logo em seguida, como na figura 13.

Figura 13 - História de usuário renderizada na tela.

BDD, ou Behaviour Driven Development (Desenvolvimento Orientado por Comportamento), é um artefato que se concentra em obter uma compreensão clara dos comportamentos desejados do software através de discussões com as partes interessadas e expressando esses comportamentos em uma linguagem natural. O BDD é mostrado na tela como um documento de texto em markdown, como na figura 14. Exemplo de como o BDD foi montado utilizando o contexto da IA:

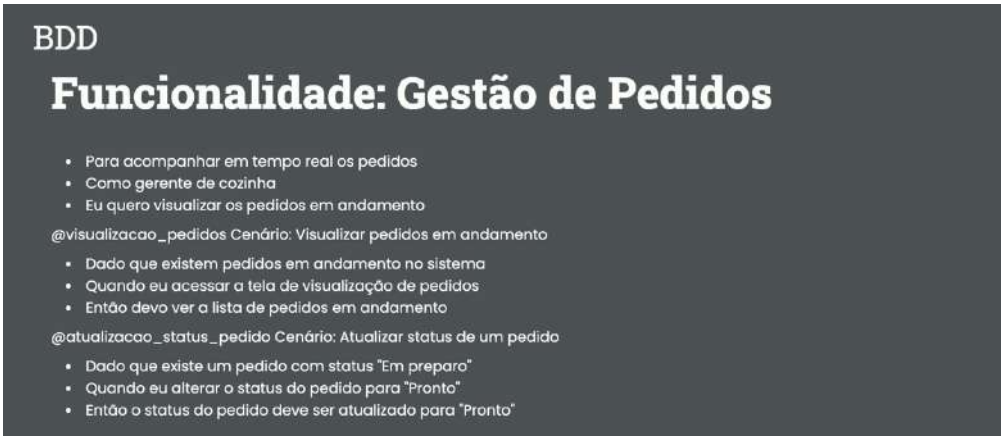


Figura 14 - BDD montado por IA

3.3 Por que artefatos em Markdown?

A World Wide Web, criada por Tim Berners-Lee no CERN em 1989, projetada para facilitar a colaboração entre físicos de todo o mundo, em conjunto, o HTML foi desenvolvido inicialmente como linguagem de publicação da web com a ideia de dar a possibilidade de vincular as páginas entre si utilizando a marcação de texto [25]. Desde então, o HTML evoluiu de uma linguagem simples para um sistema complexo de marcação, permitindo a criação de páginas ricas em multimídia e se tornando a base da web que conhecemos hoje.

Em 2004, Aaron Swartz e John Grubber apresentaram ao mundo o Markdown [26][27]. O projeto consiste em simplificar a escrita de páginas da web e posts de blog, deixando tão fácil como a escrita de um e-mail. A sintaxe é simples e o texto é convertido para HTML automaticamente [26][27], existindo também ferramentas que realizam o processo contrário, convertendo de HTML para Markdown.

A sintaxe em Markdown é adotada em diversas ferramentas como Notion, Obsidian, Dropbox Paper, entre outros. O formato é adotado também em frameworks que documentam tomadas de decisão, como o MADR [28], e na construção de documentação de ferramentas, podendo ser transformadas em páginas da web e editadas de forma simples. A escolha do Markdown foi fundamentada para facilitar o reuso dos artefatos gerados com outras ferramentas pertencentes ao fluxo de trabalho do usuário.

4 Pesquisa, resultados e futuro

Ao finalizar a etapa de desenvolvimento, para avaliar a eficácia do projeto, foi realizada uma pesquisa com um formulário enviado a pessoas de diversas áreas da tecnologia da informação. O objetivo era que as pessoas utilizassem a ferramenta por um tempo e responder à pesquisa em seguida, com suas opiniões acerca do projeto e como ele pode ser eficaz no suporte ao desenvolvimento de software. Em paralelo, outro objetivo a ser cumprido pelo formulário seria a coleta de feedback sobre a solução desenvolvida, a fim de identificar pontos fortes e áreas de melhoria. A qualidade das respostas foi priorizada em vez da quantidade, a fim de obter informações mais precisas.

Os resultados da pesquisa mostraram que a solução foi bem recebida pelos participantes, que elogiaram sua facilidade de uso e eficácia na resolução de problemas. O feedback coletado será utilizado para aprimorar ainda mais a solução, garantindo que ela atenda às necessidades dos usuários de maneira eficiente e eficaz.

4.1 Pesquisa e feedback

O formulário enviado é composto por 32 perguntas, alternando entre perguntas objetivas e discursivas abordando aspectos como perfil do usuário, experiências anteriores com desenvolvimento de software e sobre a usabilidade e eficácia da plataforma desenvolvida. Até o presente momento da escrita desse trabalho, 14 pessoas testaram a ferramenta e responderam todo o formulário da pesquisa. A plataforma recebeu 52 usuários novos a partir da data em que a pesquisa foi enviada, desses, 38 criaram um projeto com artefatos gerados por IA.

Realizando a análise das respostas, alguns insights interessantes puderam ser analisados com o perfil dos usuários.

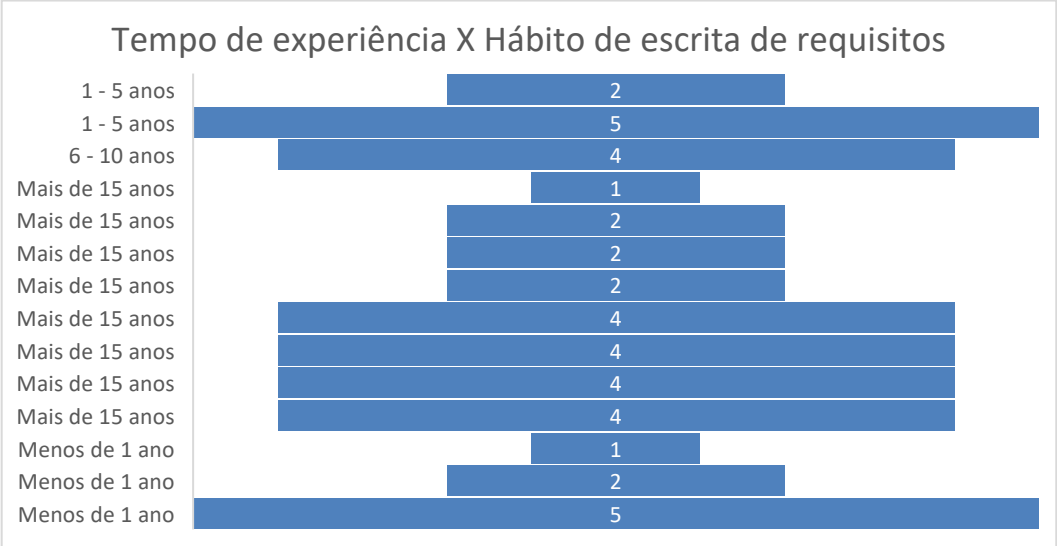


Figura 15 - Gráfico que compara o tempo de experiência na área com o hábito de documentar requisitos.

O gráfico da figura 15 é originado do cruzamento de respostas entre duas perguntas objetivas, uma para obter o tempo de experiência do usuário na área em que atua, e a segunda com o seguinte enunciado: "Indique o quão você concorda com a seguinte afirmação: ""Costumo escrever requisitos funcionais e não funcionais no meu cotidiano de trabalho."" (onde 1 significa discordo totalmente e 5 concordo totalmente)". A maioria dos usuários (com uma margem bem pequena) que concorda com a afirmação da pergunta, ou seja, possuem o hábito de escrever requisitos funcionais e não funcionais, são usuários com mais de 5 anos na área. Os usuários que possuem menos de um ano na área, apenas um concorda com a afirmação da pergunta. Os que afirmam ter mais de 15 anos, metade concordou com a afirmação. Para um estudo mais aprofundado, mais respostas seriam o ideal, porém é possível enxergar uma relação, quanto maior o tempo de experiência, mais comum é o hábito de escrita da documentação, o que pode ser identificado como falta de exemplos ou a burocracia na etapa de desenvolvimento. Em relação a quantidade de respostas no geral, foi equilibrado, como mostra na figura 16.

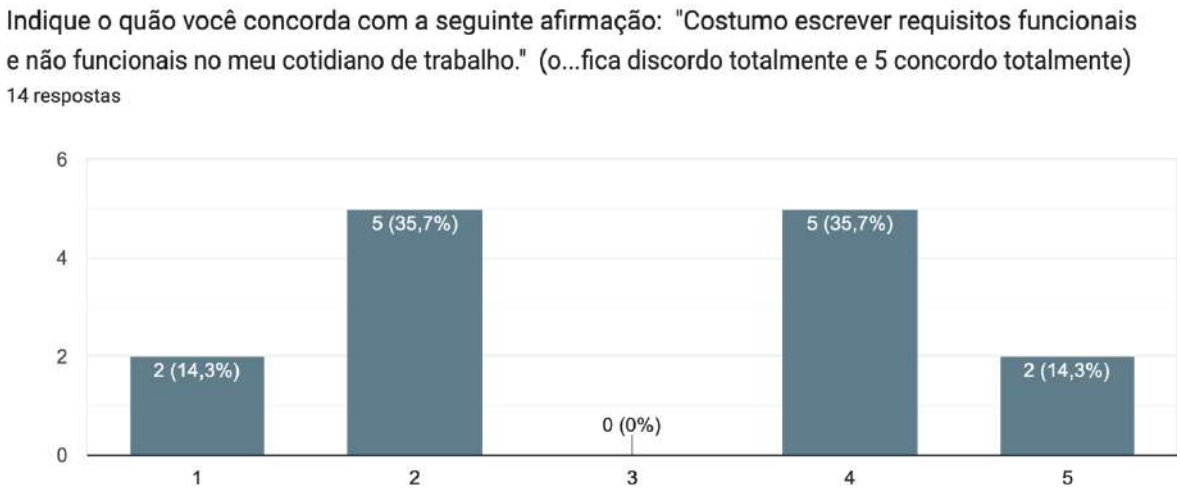


Figura 16 - Quantidade total de respostas da pergunta sobre o hábito de escrita de requisitos

Sobre a percepção da importância da etapa de desenvolvimento, apenas 14% dos usuários discordam que a etapa de desenvolvimento é importante, pergunta objetiva realizada com o seguinte enunciado: Indique o quanto você concorda com a seguinte afirmação: “"A etapa de planejamento é de grande importância antes de iniciar o desenvolvimento das aplicações em meu trabalho atual."(onde 1 significa discordo totalmente e 5 concordo totalmente)”. A representação das respostas em geral pode ser vista na figura 17.

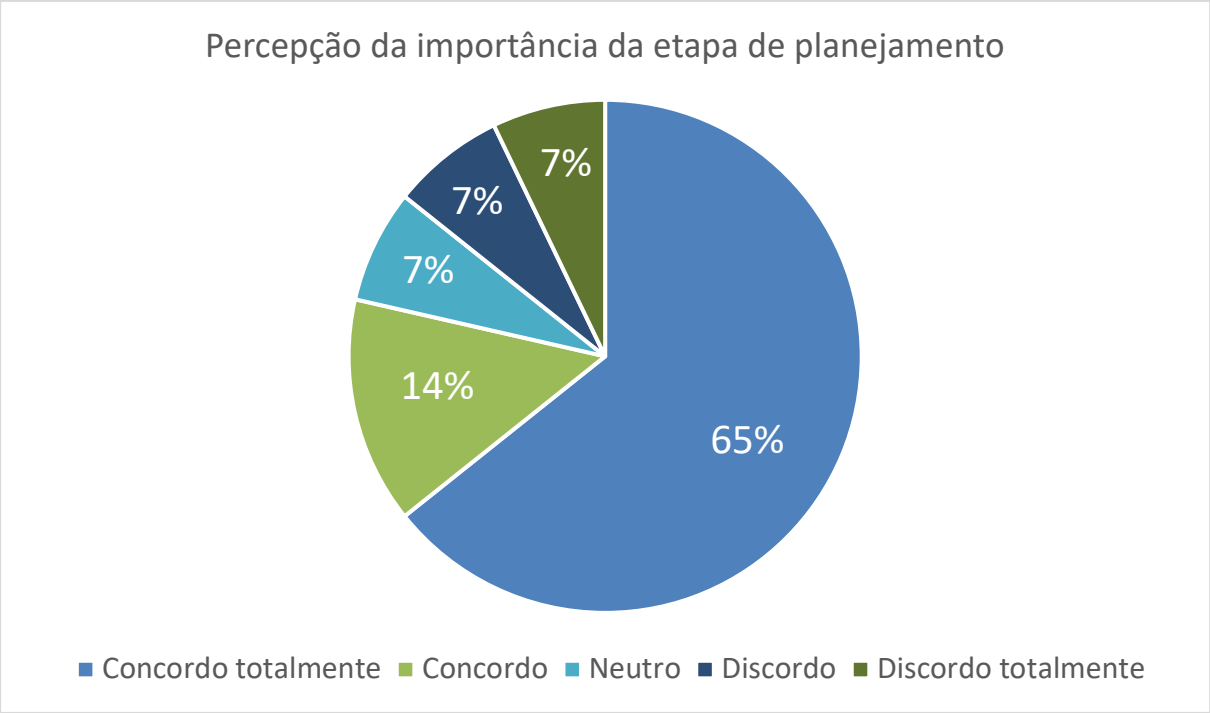


Figura 17 - gráfico de percepção da importância da etapa de planejamento

Falando sobre a usabilidade do SofIA, os usuários apresentaram facilidade ao utilizar a ferramenta, o que indica que a curva de aprendizado é relativamente baixa. Dados obtidos de perguntas objetivas utilizando o mesmo sistema de escala das anteriores.

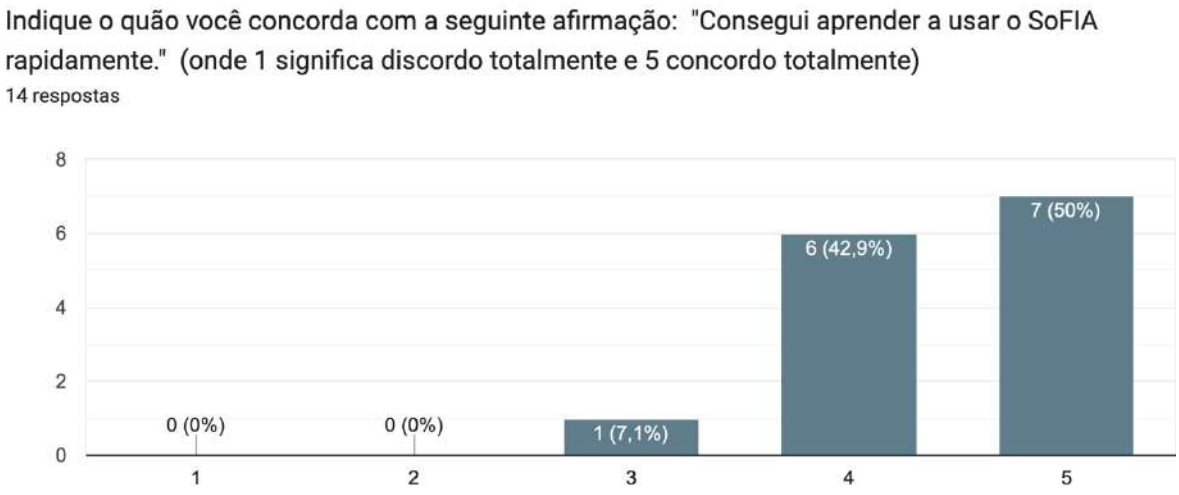


Figura 18 – Respostas da pergunta: Indique o quão você concorda com a seguinte afirmação: "Consegui aprender a usar o SoFIA rapidamente."

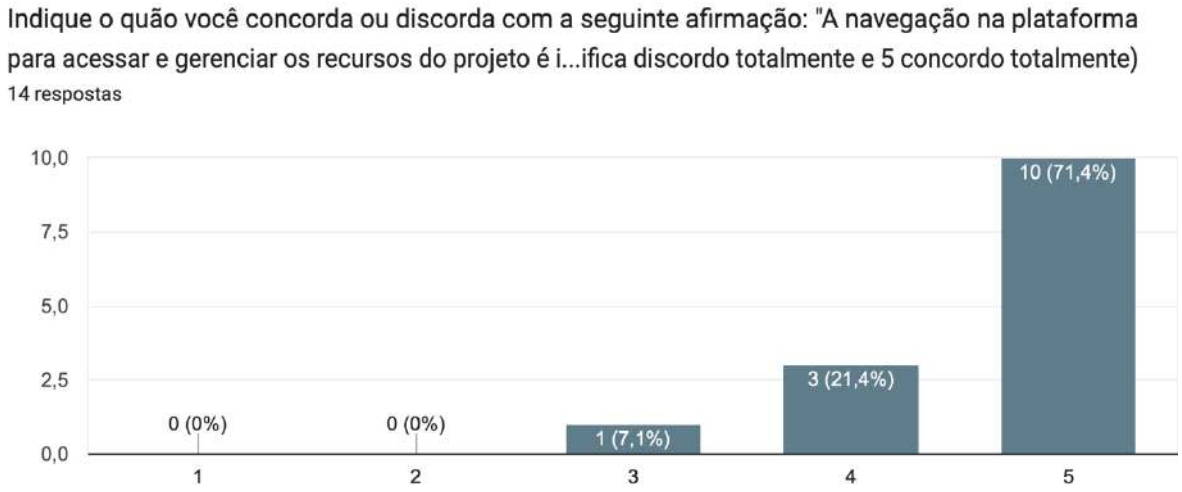


Figura 19 - respostas da pergunta: Indique o quão você concorda ou discorda com a seguinte afirmação: "A navegação na plataforma para acessar e gerenciar os recursos do projeto é intuitiva e fácil de usar."

Condensando algumas perguntas discursivas sobre a experiência e percepção da utilidade do software, foi destacado pelos usuários a capacidade da plataforma em fornecer artefatos úteis que auxiliam e orientam o desenvolvimento de software e sua capacidade de antecipar as necessidades dos usuários e fornecer soluções proativas como funcionalidades extras.

Além disso, o uso da inteligência artificial pela plataforma foi um ponto de destaque. Em destaque a capacidade do sistema de organizar informações de maneira eficiente e fornecer insights valiosos. Eles notaram que a IA muitas vezes oferecia funcionalidades e perspectivas que eles não haviam considerado inicialmente. Isso não apenas economiza tempo, mas também abre novas possibilidades no planejamento de cada software que utilizar a plataforma como ponto de partida.

A plataforma provou ser uma ferramenta valiosa para o desenvolvimento de software. Sua capacidade de fornecer artefatos úteis, organizar informações de maneira eficiente e oferecer insights inovadores torna-a uma excelente escolha para qualquer pessoa interessada. Com base no feedback dos usuários, é seguro dizer que a plataforma pode servir como um excelente ponto de partida para o desenvolvimento de softwares. Seu uso pode acelerar o processo de desenvolvimento e melhorar a experiência dos desenvolvedores e gestores na etapa de planejamento, especialmente no levantamento de requisitos.

Em conclusão, o SofIA e plataformas similares apresentam um valor inestimável no mundo do desenvolvimento de software. Apresentando ao mundo um excelente exemplo de como a inteligência artificial pode e deve ser utilizada para apoiar desenvolvedores e gestores no auxílio ao desenvolvimento de software, seja no planejamento ou na execução.

4.2 Trabalhos futuros

A plataforma, alimentada pelo feedback dos usuários, vai continuar a ser desenvolvida utilizando o feedback dos usuários da pesquisa e deve retornar para mais experimentos no. O uso de IA em colaboração com o ser humano em suas atividades deve continuar a ser explorado não apenas na etapa de planejamento, mas em todas as outras etapas do desenvolvimento de software.

Neste trabalho, a colaboração entre humanos e IA provou ser extraordinária, especialmente na etapa de planejamento. No entanto, o potencial da IA vai além do planejamento, existe um gigantesco potencial de revolucionar processos na engenharia de software, tornando os processos mais eficientes e menos propensos a erros. O uso de IA deve ser explorado em outras etapas do desenvolvimento de software, como testes, correção de código, DevOps e etc.

5 Bibliografia

[1] Guide to the Software Engineering Body of Knowledge SWEBOK ® A Project of the IEEE Computer Society. [s.l: s.n.]. Disponível em: <https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf>. Acesso em: 13 de dezembro de 2023.

[2] FERNÁNDEZ, D. M. et al. Naming the Pain in Requirements Engineering: Contemporary Problems, Causes, and Effects in Practice. Empirical Software Engineering, v. 22, n. 5, p. 2298–2338, 24 out. 2016. Disponivel em: <https://arxiv.org/abs/1611.10288>. Acesso em: 13 de dezembro de 2023.

[3] LEWIS, J. A. Waiting for Skynet. www.csis.org, 18 jan. 2018. Disponível em: < https://www.csis.org/analysis/waiting-skynet>. Acesso em: 13 de novembro de 2023.

[4] Neves, André; Calegário, Filipe; Belfort, Rui; Meira, Silvio; Garcia, Vinicius. "inteligências individual, social e artificial [um novo espaço estratégico para criar, colaborar e agir]". Disponível em: <https://tds.company/wp-content/uploads/2023/05/ebook-inteligencia-individual-social-artificial-tds-v2.pdf>. Acesso em 13 de dezembro de 2023.

[5] "A Whole New Way of Working". Disponível em: <https://www.microsoft.com/en-us/worklab/ai-a-whole-new-way-of-working>. Acesso em 13 de dezembro de 2023.

[6] NAVEED, H. et al. A comprehensive overview of large language models. Disponível em: <http://arxiv.org/abs/2307.06435>. Acesso em: 1 de fevereiro de 2024.

[7] KASNECI, E. et al. ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, v. 103, n. 102274, 1 abr. 2023. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1041608023000195>. Acesso em: 1 de fevereiro de 2024.

[8] GORICHANAZ, T. ChatGPT turns 1: AI chatbot's success says as much about humans as technology. Disponível em: <https://theconversation.com/chatgpt-turns-1-ai-chatbots-success-says-as-much-about-humans-as-technology-218704>. Acesso em: 27 de fevereiro de 2024.

[9] Microsoft Copilot: Your everyday AI companion. Disponível em: <https://copilot.microsoft.com>. Acesso em: 27 de fevereiro de 2024.

[10] Gemini - chat to supercharge your ideas. Disponível em: <https://gemini.google.com>. Acesso em: 27 de fevereiro de 2024.

[11] Llama. Disponível em: <https://llama.meta.com>. Acesso em: 27 de fevereiro de 2024.

[12] Develop. Preview. Ship. For the best frontend teams – Vercel. Disponível em: <https://vercel.com>. Acesso em: 27 de fevereiro de 2024.

[13] Express - Node.js web application framework. Disponível em: <https://expressjs.com>. Acesso em: 27 de fevereiro de 2024.

[14] Nuxt: The Intuitive Web Framework. Disponível em: <https://nuxt.com>. Acesso em: 27 de fevereiro de 2024.

[15] OpenAI API. Disponível em: <https://platform.openai.com>. Acesso em: 27 de fevereiro de 2024.

[16] Railway. Disponível em: <https://railway.app>. Acesso em: 27 de fevereiro de 2024.

[17] MongoDB - The most popular database for modern apps. Disponível em: <https://www.mongodb.com>. Acesso em: 27 de fevereiro de 2024.

[18] TypeScript - JavaScript that scales. Disponível em: <https://www.typescriptlang.org>. Acesso em: 27 de fevereiro de 2024.

[19] OPENAI. Pricing. Disponível em: <https://openai.com/pricing>. Acesso em: 27 de fevereiro de 2024.

[20] OpenAI Node API Library. Disponível em: <https://github.com/openai/openai-node>. Acesso em: 27 de fevereiro de 2024.

[21] PRISMA - DATABASE TOOLS FOR MODERN APPLICATION DEVELOPMENT. Prisma. Disponível em: <https://www.prisma.io>. Acesso em: 27 de fevereiro de 2024.

[22] Turborepo. Disponível em: <https://turbo.build/repo>. Acesso em: 27 de fevereiro de 2024.

[23] BROUSSE, N. The issue of monorepo and polyrepo in large enterprises. Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming, abr. 2019. Disponível em: < https://dl.acm.org/doi/pdf/10.1145/3328433.3328435>. Acesso em: 27 de fevereiro de 2024.

[24] Mermaid | Diagramming and charting tool. Disponível em: <https://mermaid.js.org>. Acesso em: 27 de fevereiro de 2024.

[25] LONGMAN, A. Chapter 2. A History of HTML. Disponível em: <https://www.w3.org/People/Raggett/book4/ch02.html>. Acesso em: 15 de março de 2024.

[26] Daring Fireball: Markdown. Disponível em: <https://daringfireball.net/projects/markdown/>. Acesso em: 15 de março de 2024.

[27] Markdown (Aaron Swartz: The Weblog). Disponível em: <http://www.aaronsw.com/weblog/001189>. Acesso em: 15 de março de 2024.

[28] About Markdown Any Decision Records. Disponível em: <https://adr.github.io/madr/>. Acesso em: 15 de março de 2024.

6 Apêndices

6.1 Apêndice A – Descrição do projeto gerada pela IA

```
# Descrição do Projeto

O projeto consiste no desenvolvimento de um Sistema Integrado para Controle de Restaurante, que engloba três aplicativos distintos: para a cozinha, o caixa e o entregador. Cada aplicativo possui funcionalidades específicas que visam otimizar e facilitar as operações do restaurante, proporcionando uma visão unificada das informações. Além disso, a implementação de uma interface de gestão centralizada permitirá a integração e visualização das informações dos três aplicativos, contribuindo para a tomada de decisões estratégicas.

# Declaração do Escopo

## Objetivo
Desenvolver um sistema integrado para controle de restaurante, composto por três aplicativos distintos: para a cozinha, o caixa e o entregador, que permitam a gestão eficiente das operações do restaurante.

## Funcionalidades do Aplicativo para a Cozinha
1. Gestão de Pedidos
2. Controle de Estoque
3. Gerenciamento de Receitas
4. Comanda Eletrônica

## Funcionalidades do Aplicativo para o Caixa
1. Registro de Pedidos
2. Controle de Pagamentos
3. Relatórios de Vendas
4. Integração com Sistema de Reservas
```

Funcionalidades do Aplicativo para o Entregador

1. Roteirização de Entregas
2. Acompanhamento de Pedidos
3. Assinatura Eletrônica
4. Integração com GPS

Interface de Gestão Centralizada

Integração e visualização das informações dos três aplicativos, proporcionando uma visão unificada do restaurante e facilitando a tomada de decisões estratégicas.

Critérios de Aceitação

1. Os aplicativos para a cozinha, o caixa e o entregador devem ser desenvolvidos de forma a atender às funcionalidades especificadas.
2. A interface de gestão centralizada deve permitir a integração e visualização das informações dos três aplicativos, oferecendo uma visão unificada do restaurante.
3. O sistema deve fornecer notificações em tempo real para acompanhamento de pedidos e recebimento de novos pedidos.
4. A roteirização de entregas deve considerar a localização dos clientes e otimizar o tempo de deslocamento.
5. O sistema deve oferecer controle de estoque automático e gerar alertas de reposição de ingredientes conforme a necessidade.
6. Os relatórios de vendas devem ser detalhados e permitir análises de desempenho e comparações de períodos.
7. A coleta de assinaturas eletrônicas deve ser implementada para comprovação do recebimento dos pedidos.
8. A interface dos aplicativos deve ser amigável e intuitiva para os usuários.

```
# Limites do Projeto

1. O projeto se limita ao desenvolvimento dos aplicativos para
a cozinha, o caixa e o entregador, bem como da interface de
gestão centralizada.
2. Não está prevista a integração com outros sistemas de gestão
de restaurantes.
3. As funcionalidades dos aplicativos estarão restritas às
especificações fornecidas e não incluirão outros módulos ou
extensões não mencionados.
4. O sistema integrado não contemplará a gestão financeira e
contábil do restaurante.

# Entregas

| Entregas                                              | Data
Prevista |
|-------------------------------------------------------|-----
----------|
| Aplicativo para a Cozinha                                 |
DD/MM/AAAA    |
| Aplicativo para o Caixa                                   |
DD/MM/AAAA    |
| Aplicativo para o Entregador                              |
DD/MM/AAAA    |
| Interface de Gestão Centralizada                          |
DD/MM/AAAA    |
| Documentação do Sistema                                   |
DD/MM/AAAA    |
| Testes e Validação                                        |
DD/MM/AAAA    |
```

```
| Implantação e Treinamento                                    |
DD/MM/AAAA    |
| Suporte pós-implantação                                      |
DD/MM/AAAA    |

# Restrições e Assunções

## Restrições
1.  O sistema integrado estará sujeito às restrições de
infraestrutura de tecnologia disponível no restaurante.
2. O escopo do projeto está limitado ao que foi definido na
Declaração do Escopo, não sendo previstas outras funcionalidades
além das especificadas.

## Assunções
1. Assume-se que a equipe responsável pelo projeto terá acesso
às informações e recursos necessários para o desenvolvimento dos
aplicativos e da interface de gestão centralizada.
2.  Assume-se que os usuários finais dos aplicativos serão
treinados para utilização do sistema integrado.
3. Assume-se que as integrações com dispositivos externos, como
GPS e sistema de reservas, serão viáveis e seguirão as
especificações técnicas exigidas.

# Estrutura Analítica do Projeto (EAP)

## 1. Aplicativo para a Cozinha
   - 1.1 Levantamento de Requisitos
   - 1.2 Projeto e Desenvolvimento
   - 1.3 Testes e Validação

## 2. Aplicativo para o Caixa
   - 2.1 Levantamento de Requisitos
```

```
   - 2.2 Projeto e Desenvolvimento
   - 2.3 Testes e Validação

## 3. Aplicativo para o Entregador
   - 3.1 Levantamento de Requisitos
   - 3.2 Projeto e Desenvolvimento
   - 3.3 Testes e Validação

## 4. Interface de Gestão Centralizada
   - 4.1 Levantamento de Requisitos
   - 4.2 Projeto e Desenvolvimento
   - 4.3 Testes e Validação

## 5. Documentação do Sistema
   - 5.1 Levantamento de Requisitos
   - 5.2 Elaboração da Documentação

## 6. Testes e Validação
   - 6.1 Testes dos Aplicativos
   - 6.2 Testes da Interface de Gestão
   - 6.3 Validação do Sistema Integrado

## 7. Implantação e Treinamento
   - 7.1 Preparação para Implantação
   - 7.2 Treinamento dos Usuários
   - 7.3 Implantação do Sistema

## 8. Suporte pós-implantação
   - 8.1 Acompanhamento pós-implantação
   - 8.2 Suporte Técnico
```

6.2 Apêndice B - Documento de requisitos gerado por IA

```
### Requisitos Funcionais:

1. **Gestão de Pedidos:**
   - O sistema deve permitir o acompanhamento em tempo real dos pedidos, com notificações de novos pedidos e atualização do status de preparo.

2. **Controle de Estoque:**
   - Deve ser possível controlar o estoque de ingredientes, com alertas automáticos de reposição quando necessário.

3. **Gerenciamento de Receitas:**
   - O sistema deve disponibilizar acesso a todas as receitas dos pratos do restaurante, com instruções detalhadas e histórico de modificações.

4. **Comanda Eletrônica:**
   - Deve possibilitar a visualização das comandas dos clientes, facilitando a preparação e a entrega dos pedidos.

5. **Registro de Pedidos:**
   - Deve oferecer uma interface amigável para registro de pedidos, incluindo seleção de mesas e personalização dos pedidos.

6. **Controle de Pagamentos:**
   - Deve permitir o registro de pagamentos em dinheiro, cartão e outras formas, com opções de divisão de contas e emissão de recibos.

7. **Relatórios de Vendas:**
   - Deve gerar relatórios detalhados sobre as vendas, permitindo análises de desempenho e comparações de períodos.
```

8. **Roteirização de Entregas:**
   - Deve calcular a melhor rota para as entregas, considerando a localização dos clientes e otimização do tempo.

9. **Acompanhamento de Pedidos:**
   - Deve fornecer informações sobre os pedidos em andamento, incluindo os itens e as instruções de entrega.

10. **Assinatura Eletrônica:**
   - Deve permitir a coleta de assinaturas dos clientes no momento da entrega, para comprovação do recebimento.

11. **Integração com GPS:**
   - Deve facilitar a localização dos endereços de entrega, reduzindo o tempo de deslocamento e minimizando erros de rota.

12. **Interface de Gestão Centralizada:**
   - Sugere-se a implementação de uma interface de gestão centralizada, que permita a integração e a visualização das informações dos três aplicativos, proporcionando uma visão unificada do restaurante.

Requisitos Não Funcionais:

1. **Usabilidade:**
   - O sistema deve possuir uma interface intuitiva e de fácil utilização, visando a rápida adaptação dos usuários.

2. **Desempenho:**
   - O tempo de resposta do sistema deve ser rápido, mesmo em momentos de pico de utilização.

3. **Segurança:**
   - Todas as informações dos clientes, pedidos e pagamentos devem ser protegidas e acessíveis somente por usuários autorizados.

4. **Confiabilidade:**
   - O sistema deve ser robusto e confiável, evitando falhas que possam comprometer as operações do restaurante.

5. **Manutenção:**
   - Deve ser de fácil manutenção e atualização, visando a correção de possíveis falhas e a introdução de novas funcionalidades.

6. **Integração:**
   - O sistema deve possibilitar a integração com outros sistemas utilizados pelo restaurante, como sistemas de gestão financeira e de reservas.

7. **Disponibilidade:**
   - Deve estar disponível para uso durante o horário de funcionamento do restaurante, com um tempo mínimo de inatividade para manutenção programada.

8. **Eficiência:**
   - Deve utilizar os recursos de hardware e software de forma eficiente, evitando problemas de lentidão e congestionamento do sistema.

9. **Documentação:**

   - Todo o sistema deve possuir uma documentação clara, abrangendo as funcionalidades, interfaces e procedimentos de uso e manutenção.

10. **Localização:**
   - Deve estar disponível em idioma português e ser capaz de suportar outras localizações conforme necessário.

11. **Acessibilidade:**
   - Deve ser acessível para pessoas com deficiências, cumprindo as diretrizes de acessibilidade da web.

12. **Escalabilidade:**
   - O sistema deve ser capaz de lidar com um aumento significativo no número de usuários e operações, sem comprometer o desempenho.

13. **Compatibilidade:**
   - Deve ser compatível com diferentes dispositivos (desktop, tablets, smartphones) e sistemas operacionais (Windows, iOS, Android).

Requisitos de Interface:

1. **Aplicativo para a Cozinha:**
   - Deve possuir uma interface que permita o acompanhamento em tempo real dos pedidos, com notificações visuais de novos pedidos e status de preparo.

2. **Aplicativo para o Caixa:**

- Deve oferecer uma interface intuitiva para registro de pedidos, seleção de mesas e opções de pagamento, com destaque para a personalização dos pedidos.

3. **Aplicativo para o Entregador:**
   - Deve apresentar uma interface que permita o cálculo da rota de entrega, acompanhamento dos pedidos em andamento e coleta de assinaturas eletrônicas.

4. **Interface de Gestão Centralizada:**
   - Deve proporcionar uma visão unificada das informações dos três aplicativos, facilitando a tomada de decisões estratégicas e o gerenciamento geral do restaurante.

5. **Acessibilidade:**
   - Todas as interfaces devem ser projetadas considerando as diretrizes de acessibilidade, garantindo a utilização por usuários com necessidades especiais.

6. **Localização:**
   - As interfaces devem estar disponíveis em idioma português e serem adaptáveis para suportar outras localizações conforme necessário.

Atributos de Qualidade:

1. **Desempenho:**
   - Garantir alta performance e baixo tempo de resposta, mesmo em momentos de pico de uso.

2. **Segurança:**
   - Assegurar a proteção das informações sensíveis dos clientes, pedidos e pagamentos, utilizando práticas e tecnologias adequadas para garantir a segurança.

3. **Confiabilidade:**
   - Minimizar ocorrências de falhas e garantir a estabilidade do sistema para suportar as operações do restaurante.

4. **Usabilidade:**
   - Priorizar interfaces intuitivas e de fácil utilização, reduzindo a curva de aprendizado para os usuários.

5. **Manutenção:**
   - Facilitar a manutenção do sistema, permitindo correções ágeis, atualizações e introdução de novas funcionalidades.

6. **Disponibilidade:**
   - Garantir que o sistema esteja disponível durante o horário de funcionamento do restaurante, com um tempo mínimo de inatividade para manutenção.

7. **Eficiência:**
   - Utilização eficiente dos recursos de hardware e software para priorizar a eficiência operacional.

8. **Escalabilidade:**
   - Permitir que o sistema possa lidar com um aumento significativo no número de usuários e operações sem perda de desempenho.

9. **Compatibilidade:**
   - Garantir que o sistema seja compatível com diferentes dispositivos e sistemas operacionais, proporcionando uma experiência consistente para os usuários.

10. **Localização:**

- Oferecer suporte para diferentes localizações, permitindo a adaptação do sistema para atender a diversas regiões e idiomas.

Restrições:

1. **Recursos Financeiros:**
   - O desenvolvimento do sistema deve respeitar um orçamento predefinido para investimento no projeto.

2. **Prazos:**
   - O projeto deve ser concluído dentro de um prazo específico, alinhado com as necessidades do restaurante.

3. **Tecnologias Utilizadas:**
   - O uso de tecnologias para o desenvolvimento do sistema deve estar alinhado com as políticas e recursos disponíveis no restaurante.

4. **Requisitos Legais e Regulatórios:**
   - O sistema deve estar em conformidade com todas as leis e regulamentações aplicáveis, como proteção de dados e normas de segurança.

5. **Disponibilidade de Mão de Obra:**
   - Deve-se considerar a disponibilidade de profissionais capacitados para implementar, manter e oferecer suporte ao sistema.

6. **Integração com Sistemas Existentes:**
   - O sistema deve ser capaz de integrar-se a sistemas de gestão financeira e de reservas já em uso pelo restaurante.

6.3 Apêndice C – Plano de projeto gerado por IA

```
## Objetivos do Projeto:
- Desenvolver um sistema integrado para controle de restaurante, com aplicativos específicos para a cozinha, o caixa e o entregador.
- Oferecer ferramentas que permitam o acompanhamento em tempo real dos pedidos, estoque, receitas, comandas, registro de pedidos, controle de pagamentos, relatórios de vendas, roteirização de entregas, acompanhamento de pedidos, e integração com GPS.
- Facilitar a gestão centralizada do restaurante, fornecendo uma visão unificada das informações dos três aplicativos.
- Contribuir para a tomada de decisões estratégicas por meio da disponibilização de dados e relatórios detalhados sobre as operações do restaurante.
## Escopo do Projeto:
- Desenvolvimento de três aplicativos distintos para a cozinha, o caixa e o entregador, com funcionalidades específicas para cada área.
- Implementação de um sistema de gestão centralizado que permita a integração e visualização das informações dos três aplicativos.
- Integração com GPS para otimização de rotas de entrega.
- Geração de relatórios detalhados sobre vendas, estoque, e desempenho do restaurante.
- Registro de pedidos, controle de pagamentos, e emissão de recibos.
- Gestão de pedidos em tempo real, com notificações de novos pedidos e status de preparo.
- Controle de estoque com alertas de reposição automática.
- Roteirização de entregas considerando a localização dos clientes e otimização do tempo.
- Acompanhamento de pedidos em andamento, com informações sobre os itens e instruções de entrega.
```

```
- Facilitação da coleta de assinaturas eletrônicas no momento da entrega para comprovação do recebimento.
## Cronograma:

| Atividade                                          | Duração Estimada | Início Previsto | Término Previsto |
|----------------------------------------------------|------------------|-----------------|------------------|
| Levantamento de Requisitos                         | 2 semanas        | 01/08/20        | 14/08/20         |
| Design e Prototipagem dos Aplicativos              | 4 semanas        | 15/08/20        | 11/09/20         |
| Desenvolvimento dos Aplicativos                    | 12 semanas       | 12/09/20        | 04/12/20         |
| Testes e Correções                                 | 4 semanas        | 05/12/20        | 01/01/21         |
| Implementação do Sistema de Gestão Centralizada    | 6 semanas        | 02/01/21        | 13/02/21         |
| Integração com GPS e Testes de Roteirização        | 3 semanas        | 14/02/21        | 06/03/21         |
| Geração de Relatórios e Testes de Desempenho       | 5 semanas        | 07/03/21        | 10/04/21         |
| Treinamento de Usuários e Implantação              | 3 semanas        | 11/04/21        | 01/05/21         |
| Monitoramento Pós-Implantação e Ajustes            | Indeterminado    | 02/05/21        | Em diante        |
## Recursos Necessários:

### Recursos Humanos:
- Equipe de desenvolvimento de software (programadores, designers, testadores)
- Especialistas em gestão de restaurante para levantamento de requisitos
```

```
- Equipe de suporte e treinamento para usuários
- Gerente de projeto

### Recursos Materiais:
- Computadores e equipamentos de desenvolvimento
- Servidores para hospedagem dos aplicativos
- Dispositivos móveis para testes de aplicativos
- Acesso a serviços de GPS e integração

### Recursos Financeiros:
- Orçamento para contratação de equipe e especialistas
- Custos com servidores e hospedagem
- Investimento em licenças de software e ferramentas de desenvolvimento
- Recursos para treinamento de usuários e implantação do sistema

### Outros Recursos:
- Parcerias com empresas de entrega para integração dos serviços de entrega
- Acesso a APIs de geolocalização e GPS
- Acesso a bases de dados de restaurantes para obtenção de informações sobre cardápios e pratos
## Orçamento:

### Custos Previstos:

- Desenvolvimento de Software: estimativa de R$ 200.000,00
- Hospedagem de Servidores: previsto R$ 15.000,00 anualmente
- Licenças de Software: estimativa de R$ 20.000,00
- Treinamento de Usuários: previsto R$ 10.000,00
- Outros Custos Operacionais: previsto R$ 5.000,00

### Fontes de Financiamento:
```

- Recursos Próprios da Empresa
- Linhas de Crédito Específicas para Desenvolvimento Tecnológico
- Investidores ou Parcerias Estratégicas

Observações:
Os custos podem variar conforme a duração do projeto, alterações nos requisitos, custos adicionais de hardware, software ou serviços. É importante realizar um acompanhamento e controle financeiro ao longo do projeto para gerenciar possíveis variações orçamentárias.
Riscos:

1. **Risco de Alteração de Requisitos:**
   - Possibilidade de mudanças frequentes nos requisitos do sistema, impactando o cronograma e o orçamento.

2. **Risco de Falhas no Desenvolvimento:**
   - Problemas técnicos ou de comunicação que podem levar a atrasos na entrega e comprometer a qualidade do sistema.

3. **Risco de Segurança de Dados:**
   - Possibilidade de vazamento de dados sensíveis dos clientes ou do restaurante, causando prejuízos financeiros e reputacionais.

4. **Risco de Integração com Serviços Externos:**
   - Dificuldades na integração com serviços de GPS, empresas de entrega ou sistemas de reservas, impactando a eficiência operacional do restaurante.

5. **Risco de Aceitação de Usuários:**

   - Possibilidade do sistema não ser bem recebido pelos usuários finais, levando a resistência na sua utilização e necessidade de ajustes pós-implantação.

6. **Risco de Escassez de Recursos:**
   - Possibilidade de escassez de recursos humanos, equipamentos, ou financeiros, impactando o desenvolvimento e a implementação do sistema.

7. **Risco de Mudanças Legislativas:**
   - Alterações na legislação relacionada ao setor de alimentação e tecnologia, impactando os requisitos do sistema e necessitando de adaptações.

Cada um dos riscos identificados deve ser acompanhado de um plano de ação para mitigação e/ou resolução, bem como um responsável pela sua gestão.
Comunicação:

Stakeholders:
- Equipe de Desenvolvimento e Gerência
- Especialistas em Gestão de Restaurante
- Usuários Finais (Cozinheiros, Caixa, Entregadores)
- Investidores ou Patrocinadores do Projeto

Canais de Comunicação:
- Reuniões Presenciais ou Virtuais
- E-mails e Comunicados Internos
- Ferramentas de Gestão de Projetos (ex: Jira, Trello)
- Grupos de Mensagens Instantâneas (ex: Slack, Microsoft Teams)

Frequência de Comunicação:
- Reuniões semanais de acompanhamento do projeto

- Relatórios de Andamento e Status Updates
- Comunicação ad-hoc conforme necessidade do projeto

Responsável pela Comunicação:
- Gerente de Projeto ou Líder da Equipe de Desenvolvimento

A comunicação eficaz é fundamental para o sucesso do projeto, garantindo que todas as partes interessadas estejam alinhadas e informadas sobre o andamento, desafios e conquistas relacionadas ao desenvolvimento do sistema integrado para controle de restaurante.
Qualidade:

Estratégia de Garantia da Qualidade:
- Estabelecimento de critérios de qualidade para o desenvolvimento de software, com revisões e testes regulares durante o ciclo de vida do projeto.
- Implementação de práticas de desenvolvimento ágil, permitindo adaptação contínua e feedback rápido.
- Utilização de padrões de codificação, boas práticas de desenvolvimento e design.
- Realização de testes de unidade, integração e aceitação.
- Garantir a usabilidade e experiência do usuário dos aplicativos.

Monitoramento e Ajustes:
- Estabelecimento de métricas de qualidade e acompanhamento regular do desempenho do sistema.
- Coleta de feedback dos usuários e stakeholders para possíveis melhorias.
- Identificação, correção e prevenção de defeitos ao longo do desenvolvimento e após a implantação.

Garantia da Qualidade de Produto:
- Cumprimento de padrões e normas técnicas para o desenvolvimento de software.
- Utilização de ferramentas de controle de qualidade e testes automatizados.
- Acompanhamento de indicadores de desempenho e disponibilidade do sistema após a implantação.

Responsabilidades:
- A equipe de desenvolvimento é responsável por garantir a qualidade do código e dos aplicativos.
- Os especialistas em gestão de restaurante são responsáveis por validar os requisitos e funcionalidades relacionadas ao negócio.

O foco em qualidade desde o início do projeto é crucial para garantir a satisfação dos usuários finais e o sucesso do sistema integrado para controle de restaurante.
Stakeholders (Partes Interessadas):

1. **Equipe de Desenvolvimento:**
   - Desenvolvedores, Designers, Testadores, Gerente de Projeto

2. **Especialistas em Gestão de Restaurante:**
   - Proprietários, Gerentes, Chefes de Cozinha, Equipe de Atendimento, Equipe de Entrega

3. **Usuários Finais:**
   - Cozinheiros, Caixa, Entregadores

4. **Investidores ou Patrocinadores do Projeto:**
   - Responsáveis pelo financiamento e apoio estratégico do projeto.

5. **Clientes ou Consumidores Finais:**
   - Clientes do restaurante que serão impactados pela qualidade e eficiência do sistema.

6. **Fornecedores de Serviços Externos:**
   - Empresas de entrega, serviços de GPS, empresas de tecnologia relacionadas ao projeto.

7. **Órgãos reguladores:**
   - Agências governamentais ou entidades reguladoras do setor de alimentação e tecnologia, cujas normas podem impactar o projeto.

A comunicação e o envolvimento efetivo de todas as partes interessadas são fundamentais para o sucesso do projeto, garantindo que as necessidades e expectativas de cada grupo sejam consideradas ao longo do desenvolvimento do sistema integrado para controle de restaurante.
Gerenciamento de Mudanças:

Processo de Controle de Mudanças:
- Estabelecimento de um processo formal para solicitações e avaliação de mudanças no escopo, prazos, orçamento ou qualidade do projeto.
- Identificação dos impactos de cada mudança proposta no projeto e nas áreas relacionadas.
- Documentação e comunicação clara sobre as mudanças aprovadas e as razões por trás delas.

Comitê de Controle de Mudanças:
- Designação de um comitê que será responsável por avaliar e aprovar as mudanças propostas.

- O comitê incluirá membros da equipe de desenvolvimento, gestão de restaurante e patrocinadores do projeto.

Ferramentas de Suporte:
- Utilização de ferramentas de controle de versão, gestão de requisitos e gestão de projetos para registro e acompanhamento das mudanças.

Avaliação de Impacto:
- Realização de análise de impacto para cada mudança proposta, considerando cronograma, orçamento, recursos, riscos e qualidade.

Comunicação e Aprovação:
- Comunicação clara e transparente sobre as mudanças propostas, impactos identificados e decisões tomadas.
- Aprovação formal das mudanças pelo comitê de controle de mudanças, com documentação adequada.

O gerenciamento eficaz de mudanças é essencial para assegurar que o sistema integrado para controle de restaurante seja capaz de se adaptar às necessidades e demandas do negócio, dos usuários e do mercado ao longo do ciclo de vida do projeto.

6.4 Apêndice D – Documento de arquitetura gerado por IA

```
# Documento de Arquitetura de Software

## Introdução

Este documento tem como objetivo fornecer uma visão geral da arquitetura de software do Sistema Integrado para Controle de Restaurante, apresentando a estrutura e os componentes do sistema, bem como as principais decisões arquiteturais adotadas. O sistema é composto por três aplicativos distintos (cozinha, caixa e entregador) e uma interface de gestão centralizada, que visam otimizar e facilitar as operações do restaurante. A arquitetura de software é essencial para garantir a integração, desempenho, segurança e confiabilidade do sistema, atendendo aos requisitos funcionais e não funcionais estabelecidos.
# Visão Geral da Arquitetura

O sistema é projetado de forma a ser composto por quatro componentes principais:

1. **Aplicativo para a Cozinha**: Responsável pelo acompanhamento em tempo real dos pedidos, controle de estoque, gerenciamento de receitas e comanda eletrônica.

2. **Aplicativo para o Caixa**: Voltado para o registro de pedidos, controle de pagamentos, relatórios de vendas e integração com o sistema de reservas.

3. **Aplicativo para o Entregador**: Desenvolvido para a roteirização de entregas, acompanhamento de pedidos, coleta de assinaturas eletrônicas e integração com GPS.

4. **Interface de Gestão Centralizada**: Destinada à integração e visualização das informações dos três aplicativos,
```

proporcionando uma visão unificada do restaurante e facilitando a tomada de decisões estratégicas.

A interação entre estes componentes se dará através de uma arquitetura baseada em microsserviços, permitindo a escalabilidade, a manutenção e a evolução do sistema de forma independente. A comunicação entre os aplicativos será realizada de forma síncrona e assíncrona, de acordo com a necessidade de cada funcionalidade.

```plaintext
# Diagrama de Componentes

## Aplicativo para a Cozinha
- **Gestão de Pedidos**: Responsável pelo acompanhamento em tempo real dos pedidos e notificações de status.
- **Controle de Estoque**: Realiza o controle de estoque de ingredientes e gera alertas automáticos de reposição.
- **Gerenciamento de Receitas**: Disponibiliza acesso a todas as receitas dos pratos, com instruções detalhadas e histórico de modificações.
- **Comanda Eletrônica**: Possibilita a visualização das comandas dos clientes e facilita a preparação e a entrega dos pedidos.

## Aplicativo para o Caixa
- **Registro de Pedidos**: Oferece uma interface amigável para registro de pedidos, seleção de mesas e personalização dos pedidos.
- **Controle de Pagamentos**: Permite o registro de pagamentos em dinheiro, cartão e outras formas, com opções de divisão de contas e emissão de recibos.
```

```
- **Relatórios de Vendas**: Gera relatórios detalhados sobre as vendas, permitindo análises de desempenho e comparações de períodos.
- **Integração com Sistema de Reservas**: Integrado com o sistema de reservas para gerenciar as mesas e reservas dos clientes.

## Aplicativo para o Entregador
- **Roteirização de Entregas**: Calcula a melhor rota para as entregas, considerando a localização dos clientes e otimização do tempo.
- **Acompanhamento de Pedidos**: Fornece informações sobre os pedidos em andamento, incluindo os itens e as instruções de entrega.
- **Assinatura Eletrônica**: Permite a coleta de assinaturas dos clientes no momento da entrega, para comprovação do recebimento.
- **Integração com GPS**: Facilita a localização dos endereços de entrega, reduzindo o tempo de deslocamento.

## Interface de Gestão Centralizada
- **Integração e Visualização**: Permite a integração e a visualização das informações dos três aplicativos, proporcionando uma visão unificada do restaurante.
- **Análise e Tomada de Decisões**: Facilita a análise de dados e tomada de decisões estratégicas para o gerenciamento geral do restaurante.
```

```plaintext
# Interface do Usuário (User Interface - UI)

## Descrição
A camada de apresentação do sistema é projetada para oferecer uma interface intuitiva e de fácil utilização para os usuários
```

````
dos aplicativos. A experiência do usuário é priorizada, garantindo que as funcionalidades estejam acessíveis e compreensíveis.

## Tecnologias
As tecnologias utilizadas na interface do usuário incluem:
- HTML5/CSS3 para estruturação e estilização das páginas web.
- Frameworks de frontend como React, Angular ou Vue.js para o desenvolvimento das interfaces dos aplicativos.
- Bibliotecas de design responsivo, para garantir uma experiência consistente em diferentes dispositivos.

## Funcionalidades
As principais funcionalidades e interações da interface do usuário incluem:
- Registro de pedidos, seleção de mesas e personalização dos pedidos (Aplicativo para o Caixa).
- Acompanhamento em tempo real dos pedidos, notificações visuais de novos pedidos e status de preparo (Aplicativo para a Cozinha).
- Cálculo da rota de entrega, acompanhamento dos pedidos em andamento e coleta de assinaturas eletrônicas (Aplicativo para o Entregador).
- Exibição de informações integradas dos três aplicativos para facilitar a tomada de decisões estratégicas (Interface de Gestão Centralizada).
```

```plaintext
# Controladores (Controllers)

## Descrição
A camada de controle e gerenciamento das requisições é responsável por processar e direcionar as informações recebidas pela camada de apresentação para a lógica de negócios do sistema.
````

````
Os controladores desempenham um papel fundamental na coordenação das operações e interações entre a interface do usuário e o restante do sistema.

## Tecnologias
As tecnologias utilizadas nos controladores incluem:
- Node.js para a construção de controladores no lado do servidor, caso a aplicação utilize uma arquitetura de microsserviços.
- Frameworks específicos para web, como Express.js, para lidar com as rotas e requisições HTTP.
- Linguagens como JavaScript ou TypeScript, dependendo da escolha arquitetural e tecnológica adotada.

## Funcionalidades
As principais funcionalidades dos controladores incluem:
- Recebimento e validação de dados provenientes da interface do usuário.
- Roteamento das requisições para as respectivas funcionalidades de negócio.
- Gerenciamento de fluxo de dados e comunicação com os serviços e microsserviços do sistema.
- Preparação e envio de respostas para a camada de apresentação, de acordo com as ações solicitadas pelos usuários.
```

```plaintext
# Serviços (Services)

## Descrição
A camada de serviços e lógica de negócio é responsável por conter a lógica central do sistema, implementando as funcionalidades específicas e as regras de negócio do restaurante. Os serviços encapsulam as operações essenciais do sistema, implementando a funcionalidade principal oferecida pela aplicação.
````

```
## Tecnologias
As tecnologias utilizadas nos serviços podem incluir:
- Frameworks específicos para a implementação de lógica de negócio, como Spring Framework (Java) ou Django (Python).
- Bibliotecas para operações assíncronas e processamento de dados, dependendo das necessidades do sistema.

## Funcionalidades
As principais funcionalidades dos serviços incluem:
- Gestão de pedidos, incluindo acompanhamento em tempo real, status de preparo e notificações.
- Controle de estoque, com alertas automáticos de reposição e gestão de ingredientes.
- Cálculo da melhor rota para as entregas, considerando a localização dos clientes e otimização do tempo.
- Integrações com sistemas de reservas, GPS e outras ferramentas externas.
```

```plaintext
# Banco de Dados (Database)

## Descrição
A camada de persistência de dados é responsável por armazenar, acessar e gerenciar as informações do sistema de forma segura e eficiente. O banco de dados desse sistema integrado deve suportar as operações relacionadas aos pedidos, gerenciamento de estoque, receitas, pagamentos, clientes e demais dados relevantes para o restaurante.

## Tecnologias
As tecnologias utilizadas no banco de dados podem incluir:
```

```
- Banco de dados relacional, como PostgreSQL ou MySQL, para armazenamento estruturado de dados como pedidos, estoque e clientes.
- Banco de dados NoSQL, como MongoDB ou Cassandra, para armazenamento de dados não estruturados, como logs, informações de entrega e dados menos padronizados.

## Funcionalidades
As principais funcionalidades de armazenamento e consulta de dados incluem:
- Registro e recuperação de pedidos, com acompanhamento de status e histórico.
- Controle de estoque, com gerenciamento de ingredientes, alertas de reposição e controle de quantidade.
- Armazenamento e acesso às receitas dos pratos do restaurante, com histórico de modificações e detalhes das instruções.
- Gerenciamento de informações de clientes, pagamentos e reservas.
```
```
# Visão de Implantação

A estratégia de implantação do sistema seguirá um processo gradual e planejado, visando minimizar impactos no funcionamento do restaurante e garantir uma transição suave para a nova solução. As etapas de implantação incluirão:

1. **Planejamento**: Análise detalhada dos requisitos de hardware e software necessários, definição das configurações de segurança, backup e recuperação de dados, bem como definição de um plano de contingência.

2. **Configuração da Infraestrutura**: Provisionamento e configuração dos servidores e banco de dados necessários para
```

suportar a aplicação. Garantir a disponibilidade de recursos compatíveis com os requisitos do sistema.

3. **Testes e Validação**: Realização de testes de integração, performance, segurança e usabilidade para validar o sistema em um ambiente controlado antes da implantação.

4. **Treinamento dos Usuários**: Preparação e treinamento da equipe do restaurante para utilização do novo sistema, abordando aspectos técnicos e operacionais.

5. **Implantação em Ambiente Controlado**: Lançamento do sistema em um ambiente controlado, com monitoramento constante para identificar e corrigir possíveis problemas operacionais.

6. **Implantação Gradativa**: Implantação gradual em diferentes áreas do restaurante, começando por setores específicos e expandindo para outras áreas à medida que a equipe adquire familiaridade com a nova solução.

7. **Acompanhamento pós-implantação**: Monitoramento contínuo do desempenho, funcionalidades e aceitação do sistema após a implantação. Oferta de suporte técnico e ajustes conforme necessário.

A estratégia de implantação será conduzida considerando os horários de menor movimento no restaurante, evitando interferências no atendimento aos clientes e oferecendo suporte técnico dedicado durante o processo de transição.

6.5 Apêndice E – Código da representação visual gerada por IA

```mermaid
flowchart TD
A[Aplicativo para a Cozinha]
B[Aplicativo para o Caixa]
C[Aplicativo para o Entregador]
D[Interface de Gestão Centralizada]
E(Documentação do Sistema)

A -->|Gestão de Pedidos| B
A -->|Controle de Estoque| B
A -->|Gerenciamento de Receitas| B
A -->|Comanda Eletrônica| B
B -->|Registro de Pedidos| C
B -->|Controle de Pagamentos| C
B -->|Relatórios de Vendas| D
B -->|Integração com Sistema de Reservas| D
C -->|Roteirização de Entregas| D
C -->|Acompanhamento de Pedidos| D
C -->|Assinatura Eletrônica| D
C -->|Integração com GPS| D
D -->|Interface de gestão centralizada| A
D -->|Interface de gestão centralizada| B
D -->|Interface de gestão centralizada| C
A -->|Gestão de Pedidos| D
A -->|Controle de Estoque| D
A -->|Gerenciamento de Receitas| D
A -->|Comanda Eletrônica| D
E -->|Documentação do Sistema| A
E -->|Documentação do Sistema| B
E -->|Documentação do Sistema| C
E -->|Documentação do Sistema| D
A-->|Testes e Validação| E
B-->|Testes e Validação| E
C-->|Testes e Validação| E
D-->|Testes e Validação| E
```