**Nome do Projeto**

ChatBot

**Nome dos Alunos:**

* Arthur Trindade de Souza – RA: 24.00204-6
* Pedro Henrique de Paiva Bittencourt – RA: 24.00162-7
* Murilo Rodrigues – RA: 24.01780-9
* Henrique Impastaro – RA: 24.01777-9
* Thiago Arevolo De Azevedo – RA: 23.01294-3

Sumário

[1. Descrição/Resumo do Projeto 3](#_Toc190507292)

[2. Tecnologias Empregadas 3](#_Toc190507293)

[3. Definição de Papéis no Projeto 3](#_Toc190507294)

[4. Levantamento dos Requisitos do Sistema de Software – Product Backlog 3](#_Toc190507295)

[5.1. Planejamento da Sprint 1 – Sprint Backlog 4](#_Toc190507296)

[5.2. Implementação e Testes – Sprint 1 4](#_Toc190507297)

[5.3. Review da Sprint 1 4](#_Toc190507298)

[5.4. Retrospectiva da Sprint 1 4](#_Toc190507299)

[6. Sprint 2 4](#_Toc190507300)

[7. Registro da Apresentação ao Parceiro 4](#_Toc190507301)

[8. Referências 5](#_Toc190507302)

[9. Apêndice I 5](#_Toc190507303)

## 1. Descrição/Resumo do Projeto

* Apresentação do projeto.

Um chatbot de pizzaria interativo. O bot apresentará o cardápio ao cliente, anotará o pedido e entregará para a cozinha.

* Contexto do problema a ser resolvido.

Demora na extração e na requisição dos pedidos e demora no recebimento do pedido para a cozinha.

* Objetivo do sistema e benefícios esperados.

Automatizar e agilizar o processo de fazer o pedido e de recebimento pela cozinha.

* Público-alvo e stakeholders envolvidos.

Pizzarias com problema na extração de pedidos e demora na entrega do mesmo. Usuarios, que terão uma interação mais fácil e rápida na hora de fazer o pedido.

* Escopo do projeto (o que será e não será implementado).

## 2. Tecnologias Empregadas

* Linguagens de programação utilizadas.

Python

* Frameworks e bibliotecas adotadas.

Flask- Aplicação Web.

Rasa- Desenvolvimento Backend.

API do Twilio- Conexão com o Whatsapp.

* Banco de dados e sua justificativa.

MongoDB, Banco de dados robusto e versátil e em nuvem. Ideal para aplicação em uma empresa que precisa receber e armazenar pedidos.

Ferramentas auxiliares (versionamento, colaboração, gerenciamento de tarefas).

## 3. Definição de Papéis no Projeto

* Estrutura do time de acordo com o framework SCRUM.
* Nome e função de cada integrante:
  + Product Owner (PO).

Thiago

* + Scrum Master.

Arthur Trindade

* + Desenvolvedores.

Murilo Rodrigues

Pedro Henrique

Henrique

## 4. Levantamento dos Requisitos do Sistema de Software – Product Backlog

**4.1. Metodologia para levantamento de requisitos.**

O levantamento de requisitos é uma fase essencial no desenvolvimento de software, nele se identifica as necessidades dos usuários e como as transformar em funcionalidades reais. Abaixo estão as principais metodologias utilizadas:

1. Entrevista com Stakeholders

* **Descrição:** Conversas diretas com stakeholders (usuários, clientes, gerentes).
* **Vantagens:** Permite aprofundar informações, entender necessidades e obter feedback imediato.
* **Uso recomendado:** Quando é necessário entendimento detalhado ou quando há poucos stakeholders principais.

2. Questionários e Pesquisas

* **Descrição:** Formulários enviados para coletar informações de muitos usuários.
* **Vantagens:** Rápida coleta de dados em larga escala.
* **Uso recomendado:** Quando o público é grande e disperso (ex: estudantes de uma universidade).

3. Observação Direta

* **Descrição:** Análise do usuário executando tarefas reais no ambiente de trabalho.
* **Vantagens:** Captura comportamentos e necessidades que o usuário pode não mencionar.
* **Uso recomendado:** Útil para entender processos existentes ou identificar falhas não relatadas.

4. Brainstorming

* **Descrição:** Técnica em grupo usada para gerar o máximo de ideias possíveis sobre requisitos ou soluções, sem julgamentos iniciais.
* **Vantagens:** Estimula a criatividade, promove participação ativa e pode revelar necessidades não óbvias.
* **Uso recomendado:** Quando se busca inovação, novas ideias ou entender melhor as expectativas dos usuários de forma colaborativa.

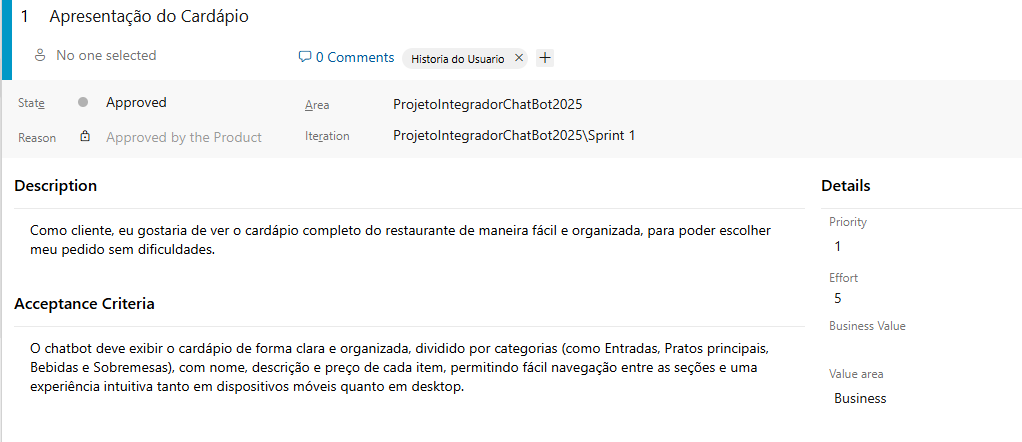
5. Prototipação

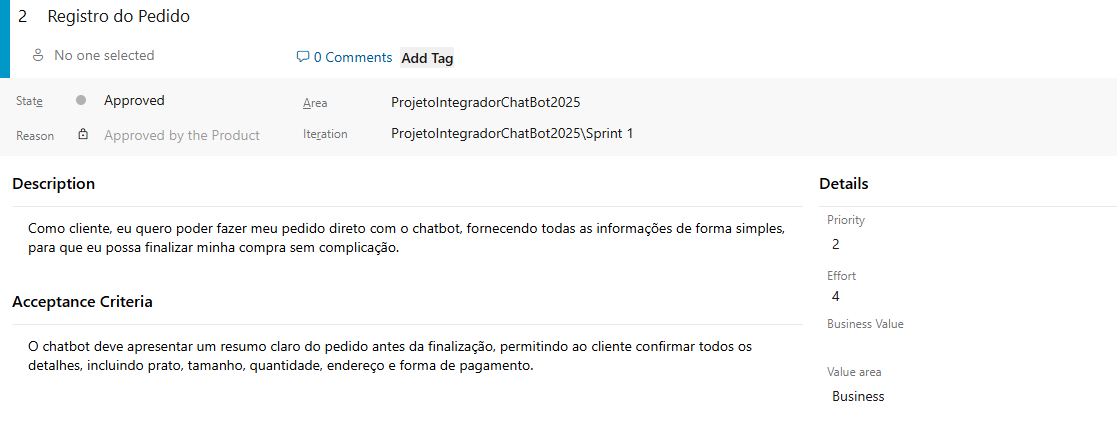
* **Descrição:** Criação de versões simplificadas da interface ou funcionalidades do sistema para visualização e validação.
* **Vantagens:** Facilita o entendimento dos requisitos, permite feedback rápido e corrige erros antes do desenvolvimento.
* **Uso recomendado:** Quando os usuários têm dificuldade em descrever o que precisam ou preferem ver exemplos.
* **Ferramenta utilizada:** Figma

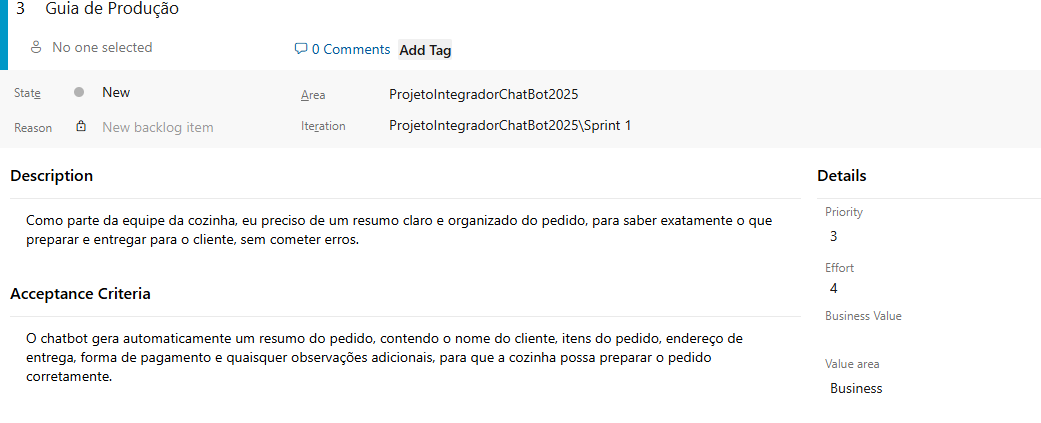
6. Histórias de Usuário

* **Descrição:** Declarações simples e curtas que descrevem funcionalidades do ponto de vista do usuário.
* **Vantagens:** Fácil de entender, priorizar e adaptar em metodologias ágeis.
* **Uso recomendado:** Em projetos ágeis como Scrum ou Kanban.

**4.2. Descrição detalhada do Product Backlog:**







**4.3 Ferramenta utilizada**

Utilizamos a ferramenta Azure Devops para a criação do product backlog, o mesmo pode ser encontrado detalhadamente no link:

[Link Backlog](https://dev.azure.com/ProjetoIntregrador/ProjetoIntegradorChatBot2025/_workitems/recentlyupdated/)

## 5.1. Planejamento da Sprint 1 – Sprint Backlog

* **Estratégia de desenvolvimento da Sprint.**

**Objetivo principal:** A equipe iniciou a Sprint com o objetivo de desenvolver o fluxo essencial de pedidos do chatbot. A prioridade foi garantir que os clientes consigam fornecer suas informações com facilidade e que a equipe da cozinha receba um resumo claro para preparo do pedido.

**Metodologia:**

**Spring length:** 2 semanas

**Code Reviews:** Pull request antes do merge sempre

**Daily Meetings:** Reuniões rapidas para acompanhamento do desenvolvimento

**Regras de Negócio**

* **Fluxo de Pedido Guiado:**

○ O chatbot deve conduzir o cliente por todas as etapas do pedido, de forma sequencial e intuitiva, minimizando dúvidas e erros durante a interação.

* **Coleta de Informações Obrigatórias:**

○ O cliente deve informar obrigatoriamente os seguintes dados:

  - Nome do prato ou produto desejado;  
  - Tamanho (se aplicável, como pequeno, médio, grande);  
  - Quantidade de itens;  
  - Endereço de entrega completo;  
  - Forma de pagamento (pix, cartão, dinheiro etc).

* **Apresentação do Resumo para Validação:**

○ Antes da finalização do pedido, o chatbot deve gerar um resumo contendo todas as informações fornecidas pelo cliente.

○ Esse resumo deve ser exibido para revisão, com a opção de confirmar ou corrigir os dados.

* **Confirmação do Cliente:**

○ O pedido só será enviado à cozinha após a confirmação explícita do cliente.

○ Caso o cliente identifique erros, ele deve ter a opção de editar as informações.

* **Resumo Enviado para a Cozinha:**

○ Após a confirmação, a cozinha recebe um resumo estruturado e claro com:  
  - Nome do cliente;  
  - Itens do pedido com quantidades e observações;  
  - Endereço de entrega;  
  - Forma de pagamento;  
  - Observações adicionais (se houver).

## **Arquitetura Proposta:**

A arquitetura do sistema proposto segue o modelo cliente-servidor com uma estrutura modular. O frontend é responsável por exibir informações e receber entradas via interface WhatsApp, enquanto o backend, desenvolvido em Python com Flask, trata os dados e gerencia o fluxo conversacional com o auxílio do Rasa. A comunicação entre frontend e backend é realizada por meio de requisições HTTP.  
  
O banco de dados MongoDB armazena informações dos pedidos e dos clientes, garantindo acesso rápido e flexível. A API do Twilio é utilizada para a integração entre o WhatsApp e o backend, possibilitando o envio e recebimento de mensagens em tempo real.  
  
A arquitetura contempla também um módulo de testes, utilizado para verificar os fluxos de conversação e validar o correto funcionamento do sistema antes da implantação.

**Solução Técnica Adotada:**

O desenvolvimento do chatbot foi realizado utilizando a linguagem Python, com o framework Flask para a interface web e o Rasa para o controle do fluxo de conversas. A integração com o WhatsApp foi feita via API do Twilio, permitindo comunicação direta com os usuários. O banco de dados MongoDB foi utilizado para armazenar os dados dos pedidos, garantindo agilidade no acesso e flexibilidade no armazenamento.

**Quadro de Tarefas Detalhado:**  
O gerenciamento das tarefas da Sprint foi feito com a ferramenta Azure DevOps. O Sprint Backlog foi organizado em cartões de tarefas que descrevem funcionalidades específicas, como:

* Criar fluxo de saudação do chatbot
* Implementar coleta de dados do cliente (nome, pedido, endereço, pagamento)
* Validar dados com o usuário antes de confirmar pedido
* Enviar resumo do pedido para a cozinha
* Implementar testes de fluxo básico
* Integração com WhatsApp via Twilio

Cada cartão foi atribuído a um membro da equipe, com prazos definidos e critérios de aceite descritos de acordo com as histórias de usuário definidas no backlog.

**Ferramenta Utilizada para Gerenciamento:**  
Azure DevOps foi a principal ferramenta de gerenciamento adotada pela equipe. Nela foram criadas as sprints, histórias de usuário, tarefas e bugs, além de possibilitar a visualização do progresso através de painéis e gráficos burndown. O uso dessa ferramenta proporcionou maior organização, rastreabilidade e colaboração entre os membros da equipe.

## 5.2. Implementação e Testes – Sprint 1

* **Link do repositório contendo a implementação.**

[Github](https://github.com/IMT-PII-3-Semestre/chatbot-poliedro)

**Algoritmos ou estruturas de dados obrigatórias implementadas**

Durante o desenvolvimento da Sprint 1, foram utilizadas algumas estruturas de dados e algoritmos fundamentais para garantir o bom funcionamento do chatbot:

**1. Dicionário para armazenamento temporário de dados do usuário (estrutura de dados):**

* Foi utilizado um dicionário em Python para armazenar temporariamente os dados coletados do usuário durante o fluxo da conversa. Isso permitiu uma estrutura organizada e acessível para manter o estado da sessão do cliente.

**2. Validação de dados com expressões regulares**

Para garantir a integridade dos dados inseridos pelo cliente, especialmente endereço e formas de pagamento, foram utilizadas expressões regulares para validação de formatos

**3. Serialização de dados**

Antes de enviar os dados para a cozinha ou gravar no banco de dados, os dados do pedido foram estruturados em formato JSON. Isso permitiu padronizar a comunicação entre módulos e facilitar a integração com o banco MongoDB

**Metodologia de Testes: TDD e BDD**

Durante o desenvolvimento do sistema, foram adotadas duas metodologias de teste amplamente reconhecidas no desenvolvimento ágil de software: **TDD** e **BDD**. Ambas visam melhorar a qualidade do código, facilitar manutenções futuras e garantir que o software atenda corretamente aos requisitos do usuário.

**TDD – (Desenvolvimento Orientado a Testes)**

O TDD é uma prática de desenvolvimento onde os testes são escritos antes do código de produção. O foco é garantir que cada unidade (função, método, componente) funcione corretamente desde o início.

**Etapas do TDD:**

1. Escrever um teste automatizado que falha (porque o código ainda não existe).
2. Escrever o código mínimo necessário para passar no teste.
3. Executar o teste para confirmar que agora passa.
4. Refatorar o código, se necessário, mantendo os testes bem-sucedidos.

**Benefícios:**

* Garante código mais confiável e com menos bugs;
* Facilita refatorações e manutenções;
* Promove desenvolvimento incremental.

**Aplicação no projeto:**  
O TDD foi utilizado principalmente para validar as funções de:

* Validação de dados do usuário;
* Tratamento de mensagens e entrada de dados;
* Persistência de informações no banco de dados.

**BDD – (Desenvolvimento Orientado a Comportamento)**

O BDD é uma evolução do TDD que foca no comportamento do sistema doponto de vista do usuário. Os testes são escritos em linguagem natural, descrevendo o que o sistema deve fazer em determinadas situações.

**Benefícios:**

* Facilita a comunicação entre desenvolvedores, testadores e stakeholders;
* Garante que o sistema atenda às expectativas do usuário final;
* Cria uma documentação viva do comportamento da aplicação.

**Aplicação no projeto:**

O BDD foi empregado na criação de testes automatizados para:

* Verificar o comportamento do bot durante a interação com o usuário;
* Garantir o fluxo correto desde a saudação até a finalização do pedido;
* Confirmar mensagens de erro e tratativas para entradas inesperadas.

**Integração das abordagens**

* O **TDD** garantiu a qualidade do código-fonte e funcionalidades internas.
* O **BDD** assegurou que o comportamento do chatbot fosse o esperado pelo usuário.

A integração dessas duas abordagens permitiu um desenvolvimento seguro, com testes bem definidos e alinhados tanto com os requisitos técnicos quanto com os funcionais.

# Testes de BDD para os Cartões de História

Os cenários a seguir representam os testes de comportamento do chatbot com base nos cartões de história definidos para o projeto. Esses testes asseguram que o sistema responda corretamente às ações esperadas dos usuários.

## **Cartão de História 1: Informar o nome**

Funcionalidade: Receber o nome do cliente  
  
Cenário: Cliente informa o nome corretamente  
Dado que o chatbot iniciou a conversa  
Quando o cliente informa seu nome como "João"  
Então o chatbot deve responder com "Olá João! Qual a forma de pagamento desejada?"

## **Cartão de História 2: Selecionar a forma de pagamento**

Funcionalidade: Escolher forma de pagamento  
  
 Cenário: Cliente escolhe a forma de pagamento "PIX"  
 Dado que o nome do cliente foi informado  
 Quando o cliente informa a forma de pagamento "PIX"  
 Então o chatbot deve responder com "Pagamento via PIX selecionado. Deseja enviar o comprovante agora?"

## **Cartão de História 3: Enviar comprovante de pagamento**

Funcionalidade: Envio de comprovante  
  
 Cenário: Cliente envia o comprovante após selecionar PIX  
 Dado que o cliente escolheu a forma de pagamento "PIX"  
 E o chatbot solicitou o comprovante  
 Quando o cliente envia uma imagem como comprovante  
 Então o chatbot deve responder com "Comprovante recebido com sucesso! Aguarde a confirmação do pedido."

## Cartão de História 4: Interação com mensagem inválida

Funcionalidade: Entrada inválida do usuário  
  
 Cenário: Cliente envia uma mensagem que não faz parte do fluxo  
 Dado que o cliente está em um fluxo de pedido  
 Quando o cliente envia "qual é o seu nome?"  
 Então o chatbot deve responder com "Desculpe, não entendi. Você pode repetir ou escolher uma das opções informadas?"

## Cartão de História 5: Finalização do pedido

Funcionalidade: Conclusão do pedido

Cenário: Cliente conclui todas as etapas do pedido  
Dado que o nome, a forma de pagamento e o comprovante foram enviados  
Quando o chatbot confirmar os dados recebidos  
Então o chatbot deve responder com "Pedido finalizado com sucesso! Em breve você receberá a confirmação pelo WhatsApp."

Esses testes comportamentais simulam interações reais com o chatbot, garantindo que o sistema funcione como esperado do ponto de vista do usuário.

**5.3 – Review da Sprint 1**

**Relato dos Resultados Obtidos**

Durante a Sprint 1, os seguintes resultados foram alcançados:

* Implementação inicial do chatbot com fluxo básico de interação com o usuário.
* Captação do nome do cliente e solicitação da forma de pagamento.
* Primeiros testes de integração com plataforma de mensagens.
* Criação dos primeiros cenários de teste em BDD baseados nos cartões de história.
* Estruturação do repositório e organização do pipeline de desenvolvimento.

**O Objetivo foi Cumprido?**

Sim. O objetivo principal da Sprint 1 era estabelecer uma base funcional mínima para o chatbot, cobrindo o início da jornada do usuário (desde a saudação até a escolha da forma de pagamento). A equipe conseguiu entregar uma versão funcional conforme planejado.

**Planejamento para a Próxima Sprint**

Para a Sprint 2, estão planejadas as seguintes entregas:

* Implementar a funcionalidade de envio e verificação de comprovante de pagamento.
* Desenvolver a lógica de conclusão de pedido e confirmação.
* Refinar a máquina de estados do chatbot para incluir exceções e entradas inválidas.
* Coletar métricas de desempenho da aplicação em ambiente de teste.

**Feedback do Parceiro Testando esta Sprint**

O parceiro responsável pelos testes da Sprint 1 forneceu feedback positivo quanto à clareza do fluxo e respostas do chatbot. No entanto, foram destacadas melhorias:

* Incluir tratamento de mensagens fora do contexto (mensagens aleatórias).
* Tornar mais claras as mensagens de confirmação após cada etapa.

Esse feedback será incorporado ao planejamento da Sprint 2 com foco em melhorar a usabilidade e resiliência da aplicação.

**5.4 – Retrospectiva da Sprint 1**

**Avaliação do Desempenho do Time**

O desempenho da equipe na Sprint 1 foi satisfatório. Todos os integrantes colaboraram de forma ativa nas tarefas propostas, mantendo uma boa comunicação e ritmo de entrega. As reuniões diárias ajudaram a identificar rapidamente obstáculos e garantir que todos estivessem alinhados com os objetivos da Sprint.

**Pontos Fortes e Fracos da Sprint**

**Pontos Fortes:**

* Boa organização das tarefas no board de acompanhamento.
* Entregas dentro do prazo estipulado.
* Cooperação entre os membros durante o desenvolvimento e testes.

**Pontos Fracos:**

* Dificuldade inicial na configuração de ferramentas de testes automatizados.
* Algumas dúvidas em relação ao escopo funcional dos cartões de história.
* Necessidade de mais detalhamento nos critérios de aceitação.
* Dificuldade na comunicação com o parceiro.

**O que Manter e o que Melhorar para a Próxima Sprint**

**Manter:**

* Rotina de reuniões diárias curtas e objetivas.
* Registro de tarefas com descrições claras.
* Práticas de versionamento e integração contínua.

**Melhorar:**

* Investir mais tempo no refinamento dos cartões de história antes do início da Sprint.
* Adotar revisão de código entre pares para aumentar a qualidade técnica.

**6 Planejamento da Sprint 2 – Sprint Backlog**

**Objetivo principal:**

* Integrar a persistência de dados com MongoDB (usuários e pedidos).
* Implementar versão mobile do chatbot no front‑end.
* Fazer melhorias visuais na interface de conversa.
* Incluir cadastro integrado ao início da conversa, gravando perfil do usuário no banco.

**Estratégia de desenvolvimento da Sprint:**

* **Duração:** 3 semanas
* **Cerimônias:** Reuniao semanal de 1 hora e atualizações diarias de progresso.

**Regras de negócio:**

1. **Fluxo de Cadastro Inicial:**
   * Ao primeiro contato, o chatbot pergunta “Como devo chamar você?” e coleta nome, endereço e telefone.
   * Se o usuário já existir no banco, esse fluxo é pulado.
2. **Cardápio e Pedido:**
   * Após cadastro ou identificação, o bot exibe opções de itens, quantidades e complementos.
   * Validação de entradas no front‑end (menus e botões responsivos).
3. **Persistência de Sessão:**
   * Cada interação grava estado parcial no MongoDB, permitindo retomada de conversas.
4. **Interface Mobile**

Funcionalidade total das telas e interface.

**Solução técnica adotada:**

* **Banco:** MongoDB
* **Driver:** PyMongo no Flask
* **Front‑end:** React PWA
* **Comunicação:** HTTP/JSON

**Quadro de tarefas detalhado:** Esta no link do DevOps

**6.2 Implementação e Testes – Sprint 2**

* **Link do repositório:**  
  https://github.com/IMT-PII-3-Semestre
* **Algoritmos e estruturas de dados obrigatórias:**
  + **Dicionário Python** para manter estado de sessão antes de persistir.
  + **Expressões regulares** para validação de nome, telefone e CEP.
  + **Serialização JSON** de objetos Mongo antes do armazenamento.
* **Abordagem de testes:**
  + **TDD:** testes unitários criados antes do código.
  + **BDD:** cenários em Gherkin
* **Principais testes de BDD:**

Cartão de História “Cadastro de Usuário”

Feature: Cadastro de usuário no primeiro contato

Scenario: Novo usuário informa dados corretamente

Given a conversa foi iniciada

When o usuário responde com nome, endereço e telefone válidos

Then o sistema salva o perfil no MongoDB

And responde "Cadastro concluído, [Nome]!"

Cartão de História “Reuso de Perfil”

Feature: Reuso de perfil em novo pedido

Scenario: Usuário já cadastrado inicia pedido

Given que o usuário existe na coleção `users`

When ele inicia um novo pedido

Then o chatbot pula a etapa de cadastro

And exibe o cardápio diretamente

* **Evidências de testes:**
  + Testes estão no Github

**6.3 Review da Sprint 2**

Esse review esta no link do DevOps a seguir: [link](https://dev.azure.com/ProjetoIntregrador/ProjetoIntegradorChatBot2025/_wiki/wikis/ProjetoIntegradorChatBot2025.wiki/2/Review-sprint-2)

**Resultados Obtidos:**

* Melhoria no front end
* Conexão com o banco de dados
* versão mobile funcional

**6.4 Retrospectiva da Sprint 2**

* **Avaliação do desempenho do time:**
  + Alta colaboração e cumprimento de prazos; comunicação diária eficiente.
* **Pontos fortes:**
  + Integração com MongoDB sem causar regressões.
  + Pair programming facilitou solução rápida de bugs críticos.
* **Pontos fracos:**
  + Documentação do PWA ficou superficial em relação ao cache offline.
  + Pipeline CI lento ao subir container MongoDB.

[Link do DevOps](https://dev.azure.com/ProjetoIntregrador/ProjetoIntegradorChatBot2025/_apps/hub/ms-devlabs.team-retrospectives.home)

**7. Registro da Apresentação ao Parceiro**  
**Local:** Sala de reuniões virtual via Teams

**Participantes:**

* + Pedro Henrique de Paiva Bittencourt
  + Thiago Arevolo De Azevedo
* Parceiro avaliador: Poliedro
* **Feedback recebido:**
* O parceiro elogiou a fluidez do fluxo conversacional, principalmente a inclusão do cadastro automático.
* A versão mobile foi destacada como um diferencial importante para o uso em campo.
* Foram sugeridas melhorias na exibição dos pedidos antigos no PWA, principalmente em relação à clareza das informações.

**Ajustes sugeridos:**

* Melhorar a responsividade dos cards em dispositivos menores.
* Incluir status de pedido (em produção, pronto, entregue) com indicadores visuais no PWA.

## 8. Referências

GITHUB. Repositório de código. Disponível em: <https://github.com/IMT-PII-3-Semestre/chatbot-poliedro>.

MONGODB, Inc. *MongoDB Manual*. Disponível em: <https://www.mongodb.com/docs/>.

PYTHON SOFTWARE FOUNDATION. *The Python Language Reference*. Disponível em: <https://docs.python.org/3/>.

FLASK. *Flask Documentation* . Disponível em: <https://flask.palletsprojects.com/en/2.3.x/.>

AZURE DEVOPS. Controle do backlog e sprints Disponível em: [Link](https://dev.azure.com/ProjetoIntregrador/ProjetoIntegradorChatBot2025/_backlogs/backlog/ProjetoIntegradorChatBot2025%20Team/Backlog%20items)

*PWA Overview*. Disponível em: <https://www.w3.org/TR/appmanifest/>.

OPENAI. *OpenAI API Documentation*. Disponível em: <https://platform.openai.com/docs/>.

*JavaScript Guide*. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>.

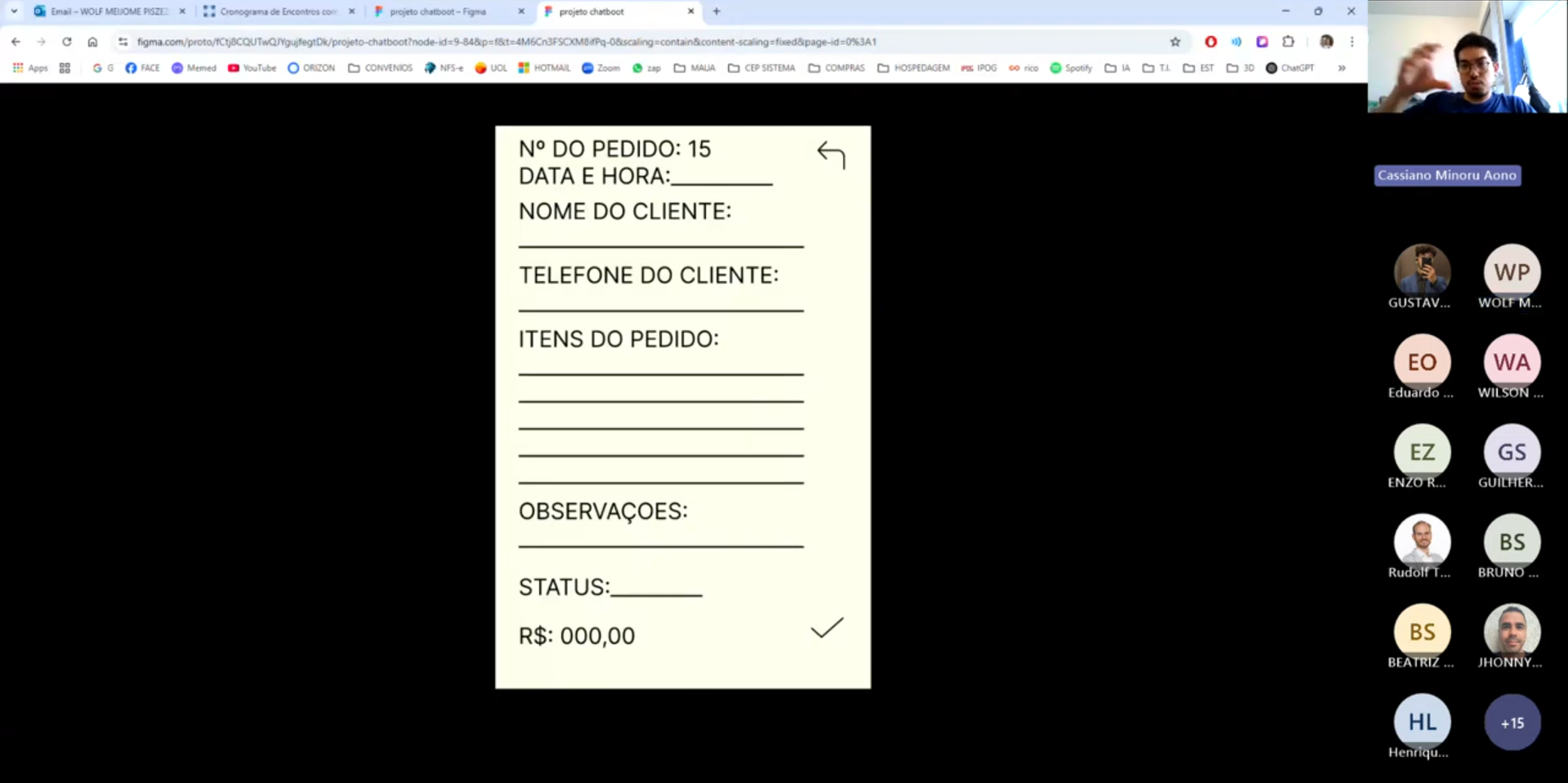
9. Apêndice I

* Registro das reuniões com o parceiro.
* Fotos, documentos de apoio e materiais relevantes.

Encontro 1 (28/03): Nessa reunião foram discutidas as primeiras informações do sistema proposto, deixando claro seus objetivos e requisitos.



Encontro 2 (25/04)



Encontro 3 (16/05): Nessa última reunião foram tiradas as últimas dúvidas referentes ao chatbot, encaminhando o projeto ao seu estágio final.

