Projeto Integrador 2º Semestre - DSM

**Disciplinas:**

Banco de Dados

Desenvolvimento Web II

Engenharia de Software II

**Professores:**

Bruno

Nilton

Orlando

**Grupo(02) / Datamind:**

**Sistema:** **QuickRotis**

|  |  |
| --- | --- |
| Integrante | **Papel Principal** |
| Gabriel Victorino | PO, Desenvolvedor Backend |
| Gustavo Francisco Habermann | Desenvolvedor Frontend/Backend |
| João Vitor de Camargo | Desenvolvedor Backend/Frontend |
| Rafael Botezelli | Documentação, Design |
|  |  |

**Fatec Araras**

**2024**

FICHA DE CONTROLE - PROJETO INTERDISCIPLINAR

DISCIPLINA CHAVE: Engenharia de Software II - PI II

PROFESSOR: Bruno Henrique de Paula Ferreira

GRUPO: 02 SEMESTRE: 2/2024

TÍTULO DO PROJETO: QuickRotis

DATA DA APRESENTAÇÃO: 03/12/2024

NOTA:

INTEGRANTES DO GRUPO: Grupo(02)

|  |  |
| --- | --- |
| Nome | Nota Individual |
| Gabriel Victorino |  |
| Gustavo Francisco Habermann |  |
| João Vitor de Camargo |  |
| Rafael Botezelli |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Araras, 03 de dezembro de 2024

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor Bruno Henrique de Paula Ferreira

**Sumário**

[1. Apresentação da Empresa 5](#_Toc184087501)

[1.1 Missão 6](#_Toc184087502)

[1.4 Link Repositório 6](#_Toc184087503)

[2. Escopo do sistema 7](#_Toc184087504)

[2.1 Objetivos do projeto 7](#_Toc184087505)

[2.2 Técnica de levantamento de requisitos 7](#_Toc184087506)

[2.3 Requisitos funcionais 7](#_Toc184087507)

[2.4 Requisitos Funcionais Futuros 9](#_Toc184087508)

[2.5 Requisitos não funcionais 11](#_Toc184087509)

[2.6 Comparativo entre sistemas 11](#_Toc184087510)

[2.7 Cronograma 12](#_Toc184087511)

[3. Documentação do Sistema 12](#_Toc184087512)

[3.1 Metodologia de Desenvolvimento 12](#_Toc184087513)

[3.2 Diagramas UML 14](#_Toc184087514)

[Diagrama de caso de uso (1º) 14](#_Toc184087515)

[DIAG-UC01 – Diagrama de Caso de Uso Principal 14](#_Toc184087516)

[Diagrama de classes (2º) 15](#_Toc184087517)

[Diagrama de sequência (2º) 16](#_Toc184087518)

[DIAG-SEQ01 – Login 16](#_Toc184087519)

[DIAG-SEQ02- Dashboard 16](#_Toc184087520)

[DIAG-SEQ03- Cadastrar Cliente 17](#_Toc184087521)

[DIAG-SEQ04- Cadastrar Usuario 17](#_Toc184087522)

[DIAG-SEQ05- Estoque 18](#_Toc184087523)

[DIAG-SEQ06 - Agendamento 19](#_Toc184087524)

[3.3 Modelo Conceitual (1º) 20](#_Toc184087525)

[3.4 Modelo lógico (1º) 20](#_Toc184087526)

[3.5 Modelo Físico (2º) 20](#_Toc184087527)

[Procedures 30](#_Toc184087528)

[Triggers 31](#_Toc184087529)

[Transactions 32](#_Toc184087530)

[3.6 Recursos e ferramentas (1º e 2º) 35](#_Toc184087531)

[3.7 Etapas / Sprints realizados (1º e 2º) 36](#_Toc184087532)

[3.8 Interface do usuário 37](#_Toc184087533)

[3.8.1 Esquema de Cores 37](#_Toc184087534)

[3.8.2 Telas finais 38](#_Toc184087535)

[4. Testes e Qualidade 40](#_Toc184087536)

[4.4 Estratégia de Testes: Descrever a estratégia de testes adotada 40](#_Toc184087537)

[4.2 Resultados dos Testes: Apresentar os resultados dos testes realizados 40](#_Toc184087538)

[4.3 Garantia da Qualidade: Descrever as práticas adotadas para garantia da qualidade 42](#_Toc184087539)

[4.4 Requisitos mínimos de hardware e software para o sistema 43](#_Toc184087540)

[4.5 Contrato para desenvolvimento de software 44](#_Toc184087541)

[5. Considerações Finais 44](#_Toc184087542)

[6 Referências 46](#_Toc184087543)

[Anexo I - Diário de bordo 47](#_Toc184087544)

[Anexo II – Cronograma efetivo 47](#_Toc184087545)

[Anexo III – Evidências 47](#_Toc184087546)

# Apresentação da Empresa

A empresa Datamind, com focos em pequenos negócios, busca juntamente a empresa Silmara Thomaz – Ateliê Gastronômico, a satisfação de seus clientes e demonstrar como tecnologias criam um grande impacto na vida de todos do cotidiano. Abaixo estão apresentados os logotipos de ambas:

**Logotipo

Descrição gerada automaticamente com confiança baixaLogotipo Datamind (Empresa dos alunos)**

**Logotipo** **empresa do cliente (antigo)**

**Texto, Carta

Descrição gerada automaticamente**

## 1.1 Missão

A empresa busca desenvolver soluções tecnológicas acessíveis e intuitivas para pequenos negócios, potencializando a organização, a gestão e experiência do pequeno empresário, aplicando conceitos de software apreendidos durante a execução do atual projeto.

1.2 Visão

Ser a principal escolha de software de gestão para microempreendedores, oferecendo ferramentas simples e eficientes que ajudam a automatizar suas operações e a crescer de forma sustentável, buscando em primeiro lugar, a satisfação do atual cliente deste projeto e a expansão do contato e influência.

1.3 Valores

Os valores da empresa Datamind, contemplam:

* **Inovação**: Buscamos sempre soluções criativas e inovadoras.
* **Qualidade**: Oferecemos produtos que agregam valor e confiança.
* **Acessibilidade**: Tornamos a tecnologia acessível para micro e pequenos empreendedores

## 1.4 Link Repositório

Abaixo segue o link do repositório principal:

<https://github.com/GabrielVictorino8266/pi_2_semestre>

# Escopo do sistema

O escopo de um sistema refere-se ao conjunto de funcionalidades, características e requisitos que definem o que o sistema irá realizar e como ele irá operar para atender às necessidades dos usuários e stakeholders. Ele delimita os limites do projeto, determinando o que está dentro e o que está fora do âmbito do sistema.

Necessário apresentar também o nome do sistema desenvolvido.

## Objetivos do projeto

Os principais objetos do projeto que a equipe busca alcançar são:

- Gerenciamento de Estoque

- Criar item, excluir item e atualizar item.

- Gerenciamento de Agendamento

- Criar, excluir e atualizar.

- Visualizar agendamentos na página de dashboard referente a atual semana.

- Sistema direcionado ao Desktop.

## 2.2 Técnica de levantamento de requisitos

As técnicas de levantamento de requisitos utilizadas para o projeto:

- **Observação**: Acompanhamento da rotina da cliente por um breve período de tempo.

- **Entrevistas**: Relato por escrito de entrevistas presenciais com a cliente, buscando levantar a maior quantidade de informações possíveis para o desenvolvimento do projeto. Esta técnica foi primordial, pois permitiu uma visualização espacial do local, permitindo a equipe de desenvolvimento considerar alguns aspectos durante a criação do sistema.

- **Formulários**: Em virtude da baixa disponibilidade do tempo, um alinhamento entre o time e a cliente nem sempre foi possível, por isso o uso de ferramentas como google forms possibilitou a equipe a acompanhar e se organizar melhor.

## Requisitos funcionais

Requisitos funcionais, são declarações dos serviços que o sistema deve fornecer, do modo como o sistema deve reagir a determinadas entradas e de como deve se comportar em determinadas situações. (Sommerville)

**Exemplo de descrição:**

|  |  |
| --- | --- |
| **[RF001]** | **Login de usuário** |
| **Descrição** | Possuir tela de autenticação para qualquer usuário com cadastro acessar o sistema. |
| **Prioridade** | Essencial |
| **Caso de uso relacionado** | -- |

|  |  |
| --- | --- |
| **[RF002]** | **Tela Inicial com atalhos para outras funções do sistema.** |
| **Descrição** | Possuir uma tela inicial onde seja possível haver atalhos para outras telas (funcionalidades) do sistema, com a exibição destes a depender do nível de permissão de cada ator do Diagrama de Caso de Uso. |
| **Prioridade** | Essencial |
| **Caso de uso relacionado** | -- |

|  |  |
| --- | --- |
| **[RF003]** | **Cadastro de Funcionário** |
| **Descrição** | A partir da tela inicial, redirecionar para uma tela onde somente o ADMIN possui permissão para cadastrar o usuário. |
| **Prioridade** | Essencial |
| **Caso de uso relacionado** | RF006 |

|  |  |
| --- | --- |
| **[RF004]** | **Gerenciar Agendamentos** |
| **Descrição** | Possuir uma tela dedicada para o agendamento de pedidos, onde deverá conter informações sobre os agendamentos existentes, mas que permite criar e atualizar agendamento, |
| **Prioridade** | Essencial |
| **Caso de uso relacionado** | -- |

|  |  |
| --- | --- |
| **[RF005]** | **Gerenciar Estoque** |
| **Descrição** | Possuir uma tela dedicada para o estoque de produtos, onde deverá contere informações sobre os agendamentos existentes, mas que permita criar e atualizar novos produtos, definir preço, etc. |
| **Prioridade** | Essencial |
| **Caso de uso relacionado** | -- |

|  |  |
| --- | --- |
| **[RF006]** | **Nível de permissão para cada usuário** |
| **Descrição** | O nível de acesso do usuário deve ser especificado no banco de dados. E, no momento do Login, o sistema deve ser capaz de lidar com quais telas e funcionalidades do sistema o usuário tem acesso. Assim sendo, o ADMINISTRADOR pode acessar todas as funções, incluindo o cadastro do usuário novo.  Qualquer usuário, sem ser ADMINISTRADOR, não pode cadastrar outros usuários, mas tem acessos a todas as outras funcionalidades do sistema. |
| **Prioridade** | Importante |
| **Caso de uso relacionado** | -- |

## Requisitos Funcionais Futuros

Devido ao tempo de desenvolvimento do projeto e de seu tamanho e complexidade, tendo em vista que os integrantes descobriram e aprenderam novas tecnologias durante o desenvolvimento, os seguintes requisitos, para tornar o sistema mais completo e funcional, foram classificados como futuros:

|  |  |
| --- | --- |
| **[RF007]** | **Listagem de 3 Últimos Preços de Compra** |
| **Descrição** | O sistema deve permite a visualização de um histórico de preços, contabilizando as 3 últimas alterações de valor refente ao produto. |
| **Prioridade** | Desejável |
| **Caso de uso relacionado** | -- |
| **Justificativa** | Já foi implementado o código sql para registrar esta informação, porém, devidoa o tempo de desenvolvimento, foi combinado que isto se encaixaria como requisito funcional futuro. |

|  |  |
| --- | --- |
| **[RF008]** | **Cadastro por interface de categorias** |
| **Descrição** | O sistema deve permitir o cadastro de categorias por meio da interface gráfica, sendo realizado somente pelo ADMINISTRADOR. |
| **Prioridade** | Desejável |
| **Caso de uso relacionado** | [Caso de uso não existe nesta versão atual] |
| **Justificativa** | Conforme alinhado com a cliente, a necessidade deste cadastro não era necessário, pois no momento, as únicas opções com as quais ela trabalha são fixas e não tem alteração. |

|  |  |
| --- | --- |
| **[RF009]** | **Cadastro por interface de tipo de produto** |
| **Descrição** | O sistema deve permitir o cadastro de tipos de produtos por meio da interface gráfica, sendo realizado somente pelo ADMINISTRADOR. |
| **Prioridade** | Desejável |
| **Caso de uso relacionado** | [Caso de uso não existe nesta versão atual] |
| **Justificativa** | Conforme alinhado com a cliente, a necessidade deste cadastro não era necessário, pois no momento, as únicas opções com as quais ela trabalha são fixas e não tem alteração. |

|  |  |
| --- | --- |
| **[RF009]** | **Reset de senha por meio da Interface do usuário Administrador** |
| **Descrição** | O sistema deve permitir o reset de senha por meio da interface do administrador. Onde somente este usuário tem esta capacidade de alteração. |
| **Prioridade** | Desejável |
| **Caso de uso relacionado** | [Caso de uso não existe nesta versão atual] |
| **Justificativa** | Conforme alinhado com a cliente, ela é a única no momento, então não há outros usuário e um suporte, durante o período vigente no contrato, seria fornecido suporte do time de desenvolvimento. |

|  |  |
| --- | --- |
| **[RF010]** | **Apontamento de relação entre Produto Final e Ingrediente** |
| **Descrição** | O sistema deve permitir cadastrar em uma tabela receitas, todas as informações referentes ao produto final e a receita. |
| **Prioridade** | Desejável |
| **Caso de uso relacionado** | [Caso de uso não existe nesta versão atual] |
| **Justificativa** | Conforme alinhado com a cliente, a mesma não cria produtos diferentes e sempre segue o mesmo cardápio. Logo, a priorização desta funcionalidade foi alocada como desejável, uma vez que, o cadastro ficaria todo pelo time de desenvolvimento. |

|  |  |
| --- | --- |
| **[RF010]** | **Listagem de Usuários do Sistema** |
| **Descrição** | O sistema deve permitir realizar uma listagem de usuários cadastrados, bem como a edição e exclusão de suas informações. |
| **Prioridade** | Desejável |
| **Caso de uso relacionado** | [Caso de uso não existe nesta versão atual] |
| **Justificativa** | Conforme alinhado com a cliente, somente ela trabalha com o negócio no momento, então a prioridade e recursos para este requisito foram alocados como futuros. |

## 2.5 Requisitos não funcionais

Requisitos não funcionais são restrições sobre os serviços ou funções oferecidas pelo sistema. Eles incluem restrições de tempo, restrições sobre o processo de desenvolvimento e restrições impostas por padrões. Os requisitos não funcionais se aplicam, frequentemente, ao sistema como um todo, em vez de às características individuais ou aos serviços. (Sommerville)

|  |  |  |
| --- | --- | --- |
| **[RNF001]** | **Plataforma Desktop.** | **Versão: 1** |
| O sistema deve ser capaz de funcionar perfeitamente no desktop, onde a cliente terá o uso principal. Como opcional, considera-se a aplicação da **responsividade**. | | |

|  |  |  |
| --- | --- | --- |
| **[RNF002]** | **Garantir diferentes níveis de acesso a cada usuário.** | **Versão: 1** |
| O nível de acesso do usuário deve ser especificado no banco de dados. E, no momento do Login, o sistema deve ser capaz de lidar com quais telas e funcionalidades do sistema o usuário tem acesso. Assim sendo, o ADMINISTRADOR pode acessar todas as funções, incluindo o cadastro do usuário novo.  Qualquer usuário, sem ser ADMINISTRADOR, não pode cadastrar outros usuários, mas tem acessos a todas as outras funcionalidades do sistema. | | |

|  |  |  |
| --- | --- | --- |
| **[RNF003]** | **Linguagem de Programação PHP.** | **Versão: 1.5** |
| O Sistema deve ser obrigatoriamente desenvolvido com a linguagem PHP, utilizando uma versão de servidor mais recente, no mínimo 8. | | |

## 2.6 Comparativo entre sistemas

Conforme observado pela equipe do projeto, foi identificado um controle manual, sem uso de nenhum sistema digital para controle de agendameto e estoque, os quais eram mantidos por meios físicos, como cadernos. O desenvolvimento do sistema digital permitiu identificar as seguintes vantagens entre a solução antiga e a solução atual.

**Tabela 1 – Comparativo das funcionalidades da aplicação**

|  |  |  |
| --- | --- | --- |
| Funcionalidades | Controle manual (sem sistema) | SISTEMA DESENVOLVIDO |
| Gerenciar Agendamentos | **-** | **X** |
| Gerenciar Estoque | **-** | **X** |
| Backup de Dados | **-** | **X** |
| Multiplataforma | **-** | **X** |
| Controle de Acesso (Login) | **-** | **X** |
| Intuitivo e Organizado | **-** | **X** |

## 2.7 Cronograma

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DATAS** | **22/08/2024 - 31/08/2024** | **01/09/2024 - 15/09/2024** | **16/09/2024 - 30/09/2024** | **01/10/2024 - 15/10/2024** | **16/10/2024 - 30/10/2024** | **01/11/2024 - 15/11/2024** | **16/11/2024 - 31/11/2024** | **03/12/2024** |
| **Definição Grupos** | **finished** |  |  |  |  |  |  |  |
| **Criação Empresa** |  | **finished** |  |  |  |  |  |  |
| **Escopo Sistema** |  |  | **finished** |  |  |  |  |  |
| **Requisitos** |  |  |  | **finished** |  |  |  |  |
| **Diagramas** |  |  |  |  | **finished** | | |  |
| **Protótipo** |  |  |  |  | **finished** | | |  |
| **Documentação** |  |  |  |  | **finished** | | |  |
| **Entrega** |  |  |  |  |  |  | **finished** |  |
| **Apresentação** |  |  |  |  |  |  |  | **finished** |

# Documentação do Sistema

Neste capítulo, são apresentados os documentos técnicos que descrevem os aspectos fundamentais do sistema desenvolvido, fornecendo uma base sólida para compreensão e manutenção futura. A documentação é uma parte essencial do processo de desenvolvimento de software, pois oferece um registro detalhado das decisões tomadas e das características do sistema.

## Metodologia de Desenvolvimento

A principal metodologia que a equipe usou para o desenvolvimento do projeto é o Kanban, o qual se baseia em um sistema visual para gerenciamento de projetos e do fluxo de trabalho, buscando otimização e produtividade do time. Se originiou no Japão, por volta de 1940, como parte do Sistema de Produção Toyota, com a ideia principal da época de utilizar cartões visuais para gerenciar o progresso do time.

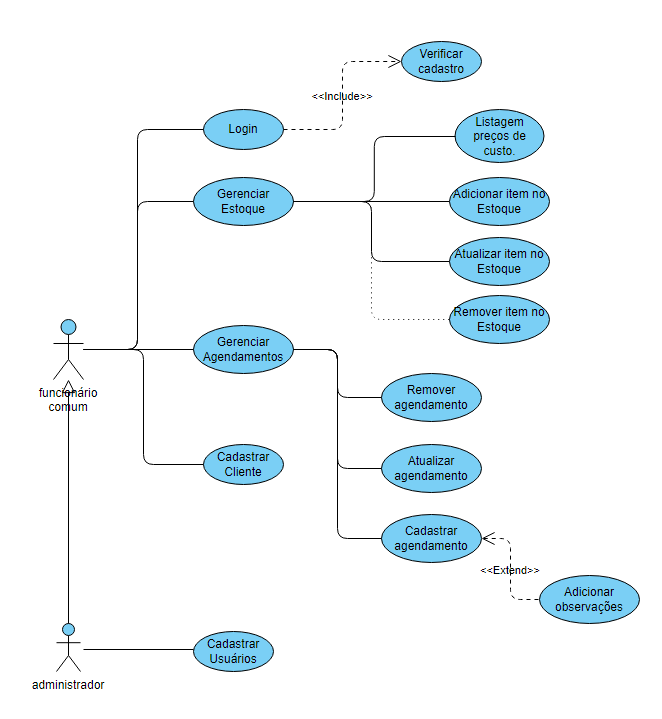
No caso deste projeto, faz-se o uso de um quadro digital, ferramenta online e gratuita, o Trello, segue abaixo uma foto atualizada referente ao quadro da equipe:



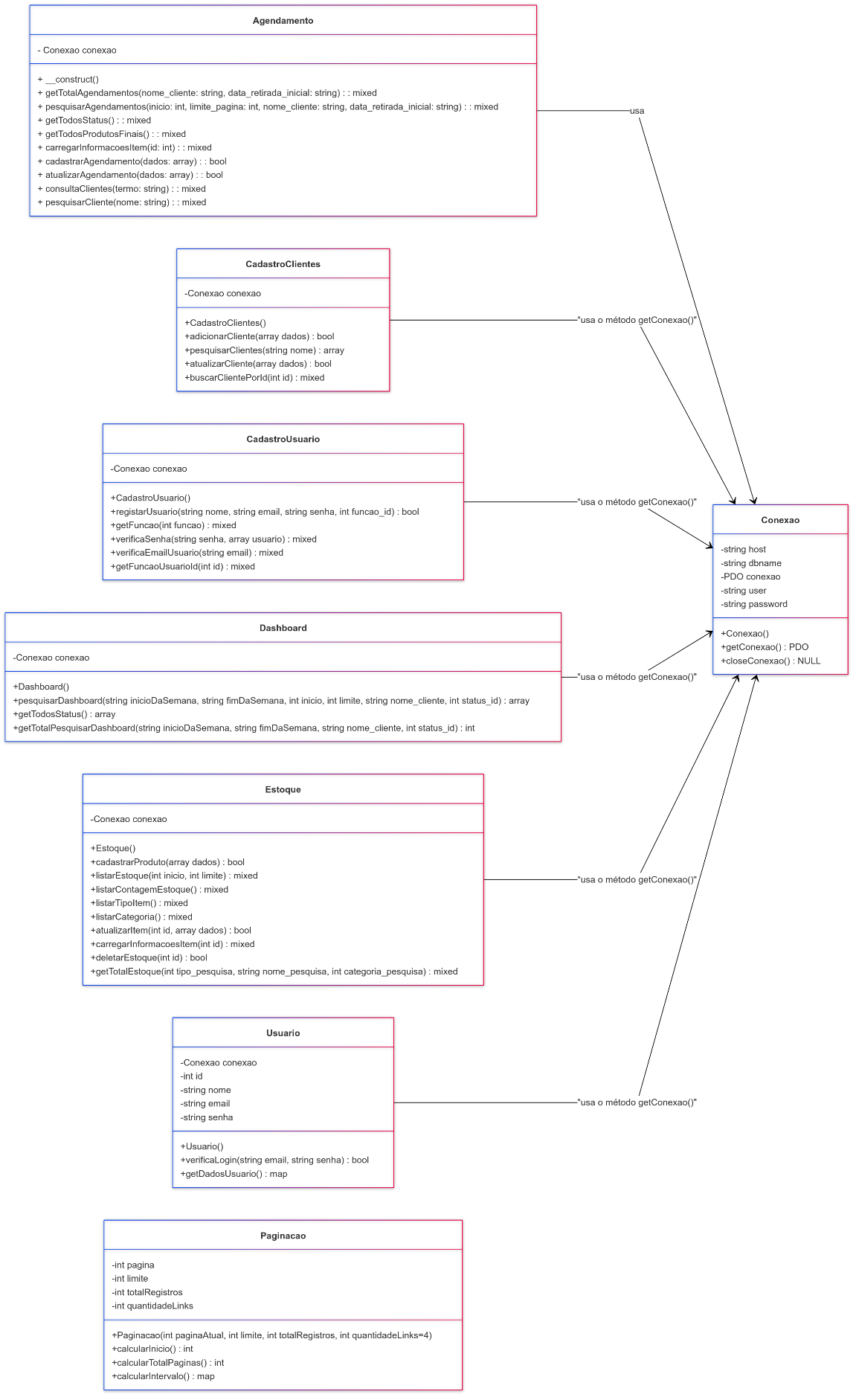
## 3.2 Diagramas UML

## Diagrama de caso de uso (1º)

### DIAG-UC01 – Diagrama de Caso de Uso Principal



## Diagrama de classes (2º)

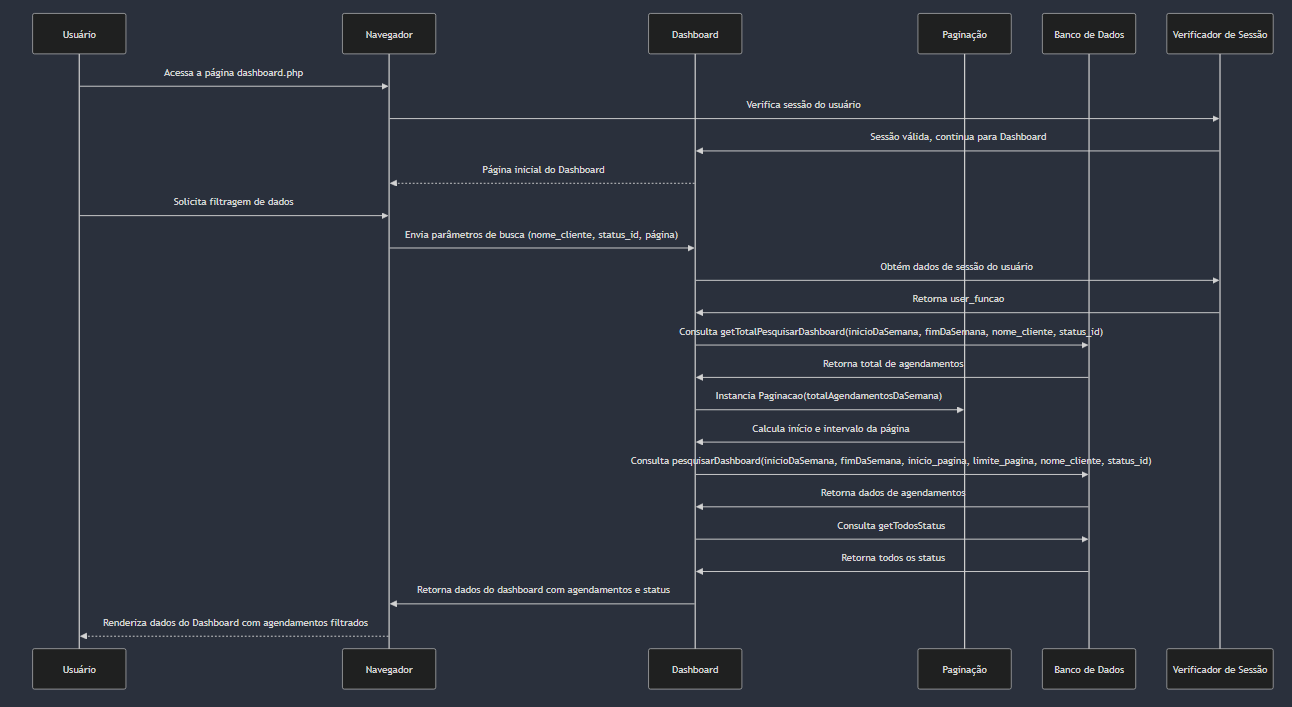


## Diagrama de sequência (2º)

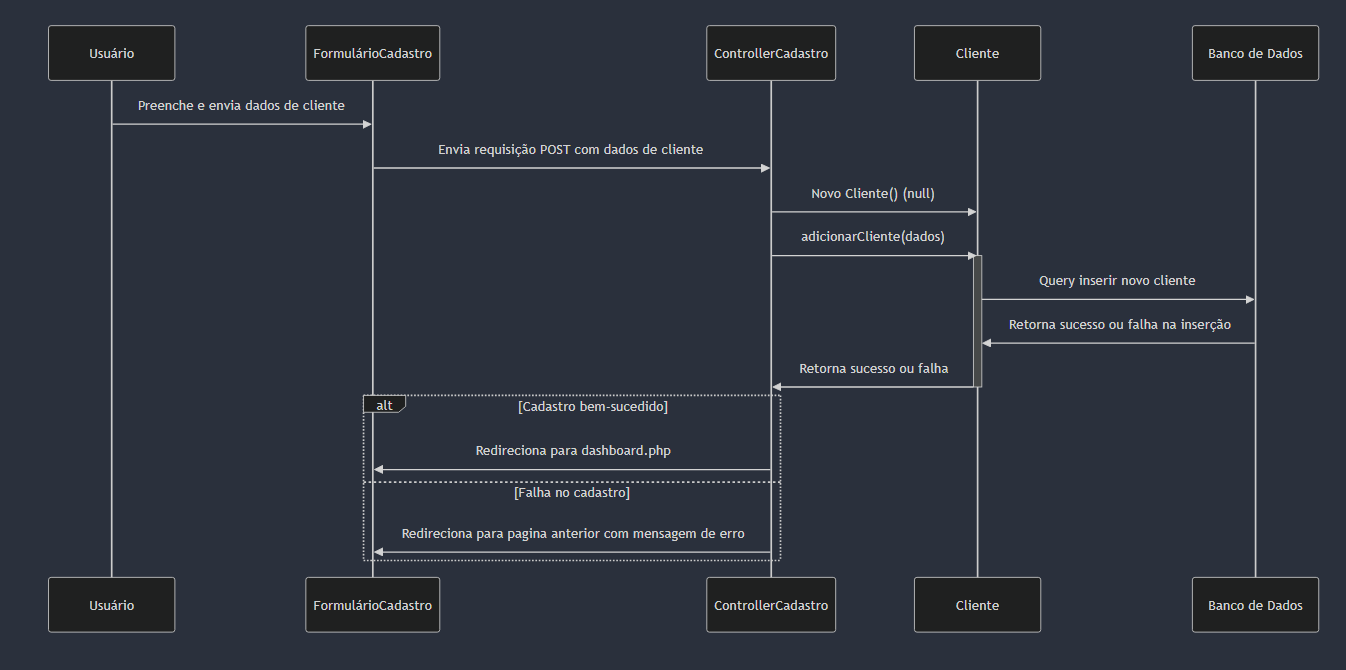
### DIAG-SEQ01 – Login



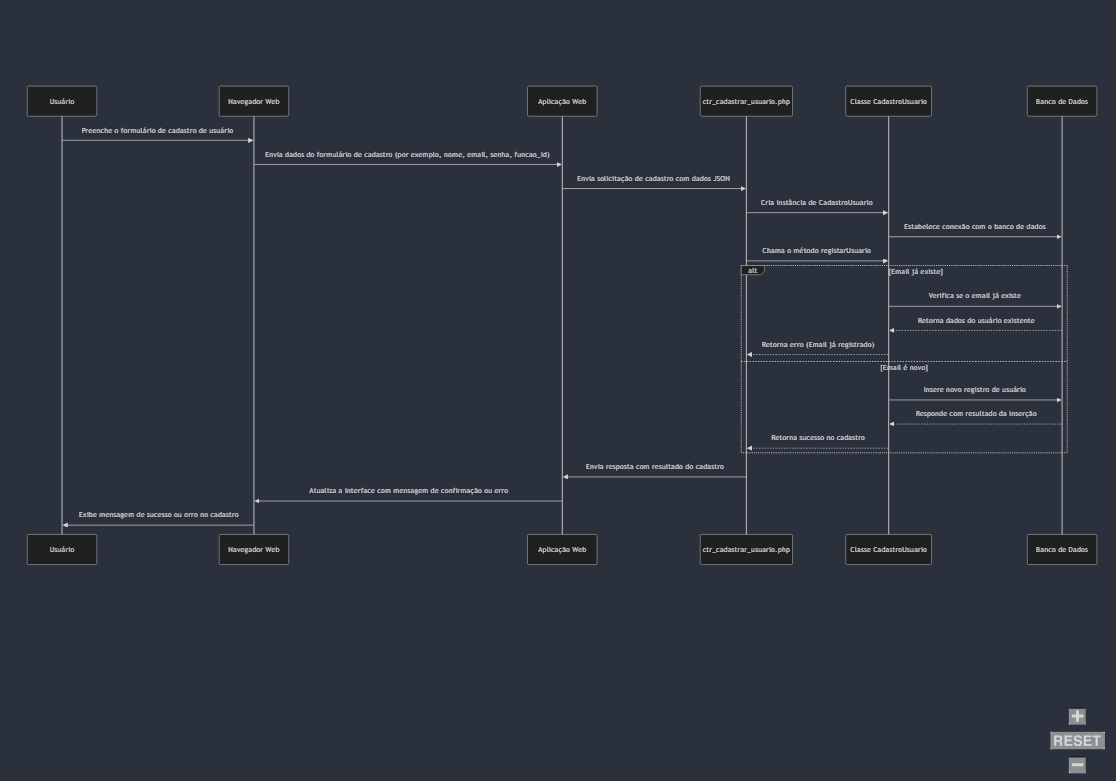
### DIAG-SEQ02- Dashboard



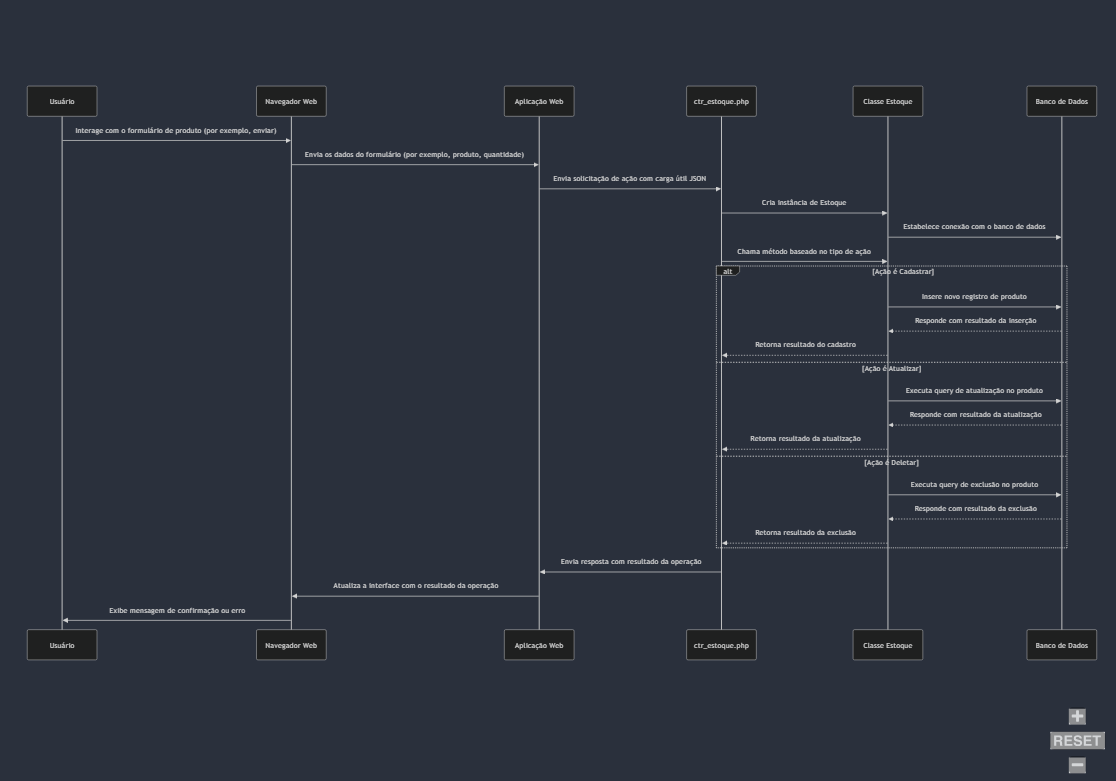
### DIAG-SEQ03- Cadastrar Cliente



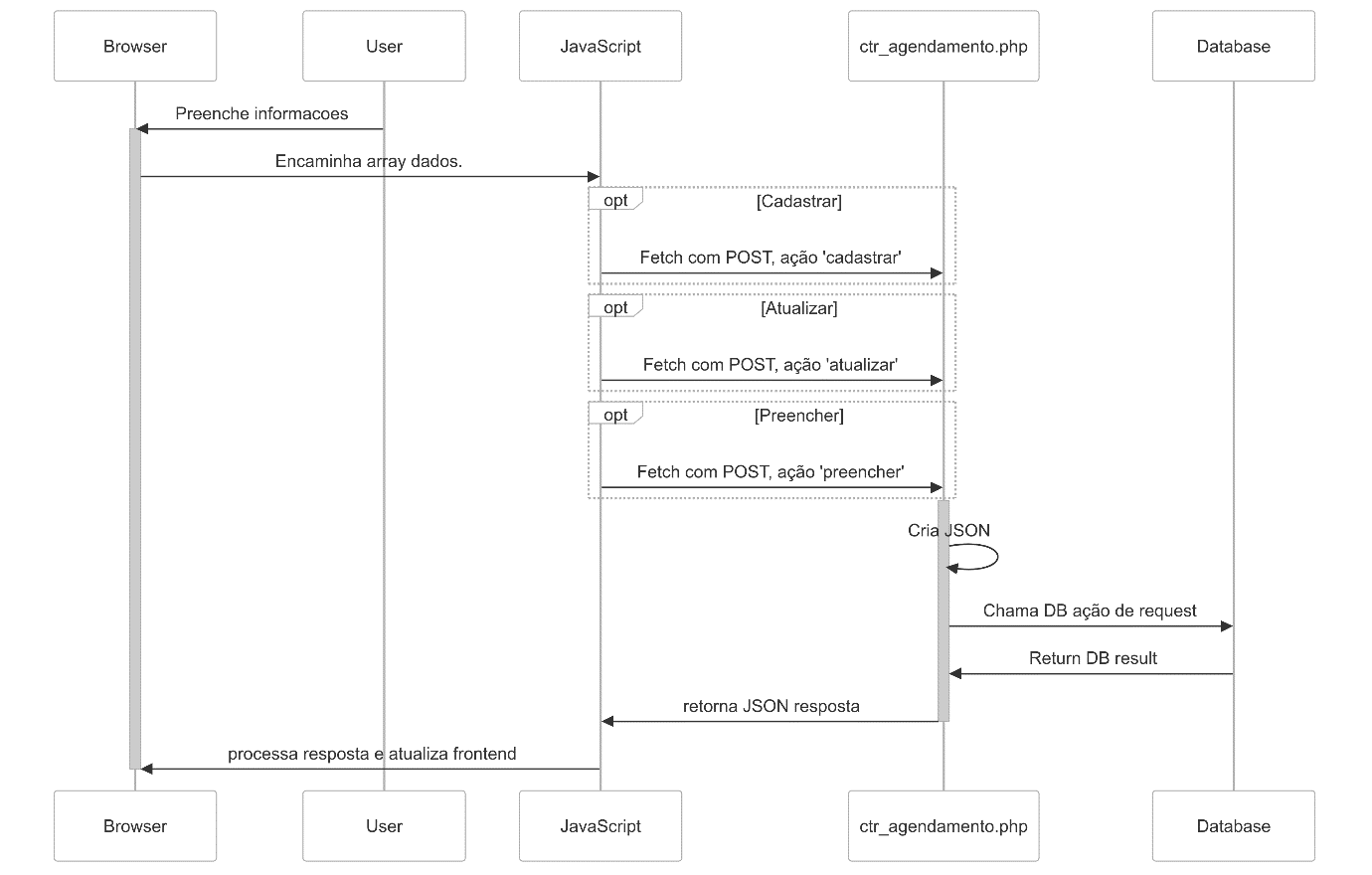
### DIAG-SEQ04- Cadastrar Usuario



### DIAG-SEQ05- Estoque

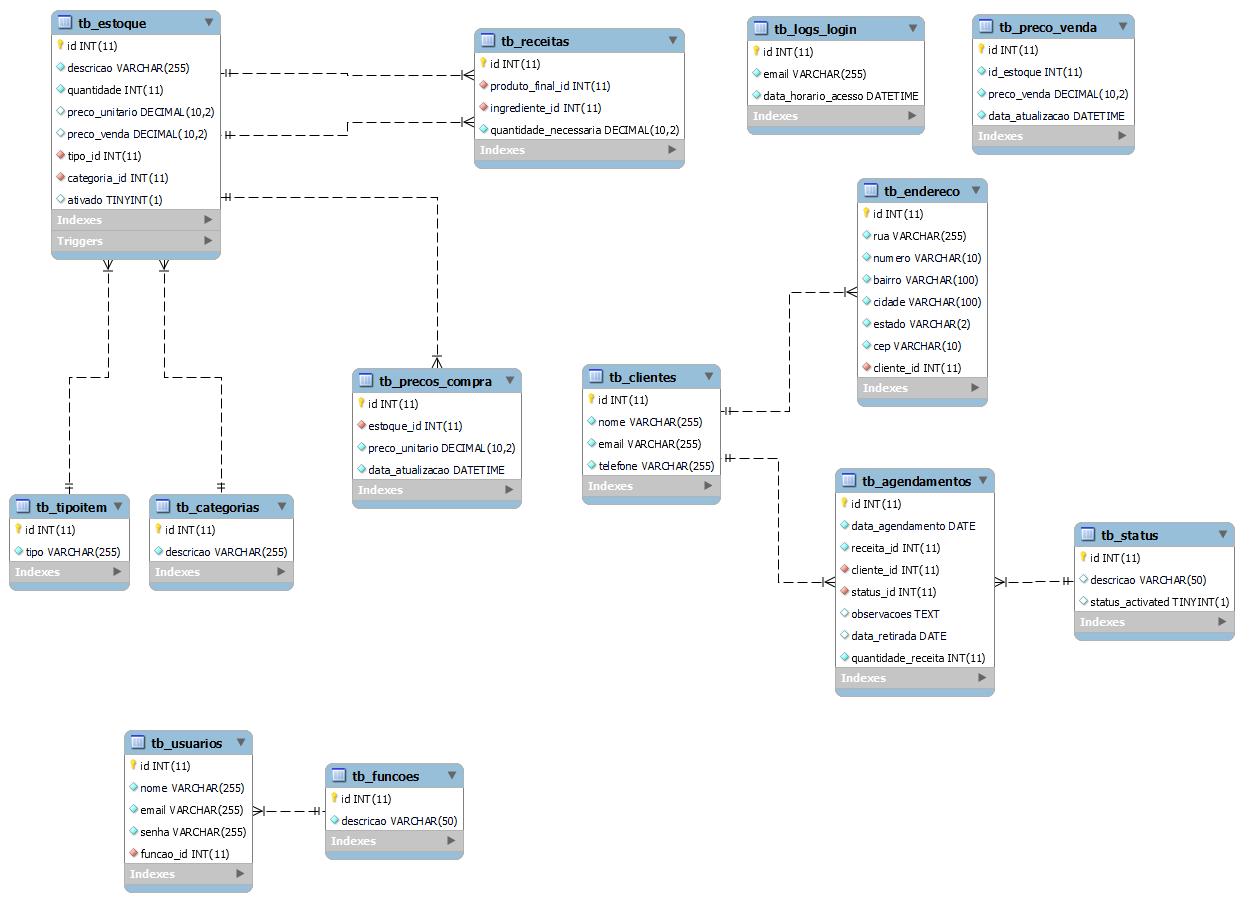


### DIAG-SEQ06 - Agendamento



## Modelo Conceitual (1º)

## Modelo lógico (1º)



## Modelo Físico (2º)

Abaixo segue o modelo físico exportado pelo MYSQL Workbench

-- MySQL Script generated by MySQL Workbench

-- Tue Dec  3 01:38:21 2024

-- Model: New Model    Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema rotisdb

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `rotisdb` ;

-- -----------------------------------------------------

-- Schema rotisdb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `rotisdb` DEFAULT CHARACTER SET utf8mb4 ;

USE `rotisdb` ;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_status`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_status` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_status` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `descricao` VARCHAR(50) NULL DEFAULT NULL,

  `status\_activated` TINYINT(1) NULL DEFAULT NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 4

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_clientes`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_clientes` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_clientes` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `nome` VARCHAR(255) NOT NULL,

  `email` VARCHAR(255) NOT NULL,

  `telefone` VARCHAR(255) NOT NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 19

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_agendamentos`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_agendamentos` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_agendamentos` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `data\_agendamento` DATE NOT NULL,

  `receita\_id` INT(11) NOT NULL,

  `cliente\_id` INT(11) NOT NULL,

  `status\_id` INT(11) NOT NULL,

  `observacoes` TEXT NULL DEFAULT NULL,

  `data\_retirada` DATE NULL DEFAULT NULL,

  `quantidade\_receita` INT(11) NOT NULL,

  PRIMARY KEY (`id`),

  INDEX `status\_id` (`status\_id` ASC) VISIBLE,

  INDEX `cliente\_id` (`cliente\_id` ASC) VISIBLE,

  CONSTRAINT `tb\_agendamentos\_ibfk\_1`

    FOREIGN KEY (`status\_id`)

    REFERENCES `rotisdb`.`tb\_status` (`id`),

  CONSTRAINT `tb\_agendamentos\_ibfk\_2`

    FOREIGN KEY (`cliente\_id`)

    REFERENCES `rotisdb`.`tb\_clientes` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 89

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_categorias`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_categorias` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_categorias` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `descricao` VARCHAR(255) NOT NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 8

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_endereco`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_endereco` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_endereco` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `rua` VARCHAR(255) NOT NULL,

  `numero` VARCHAR(10) NOT NULL,

  `bairro` VARCHAR(100) NOT NULL,

  `cidade` VARCHAR(100) NOT NULL,

  `estado` VARCHAR(2) NOT NULL,

  `cep` VARCHAR(10) NOT NULL,

  `cliente\_id` INT(11) NOT NULL,

  PRIMARY KEY (`id`),

  INDEX `cliente\_id` (`cliente\_id` ASC) VISIBLE,

  CONSTRAINT `tb\_endereco\_ibfk\_1`

    FOREIGN KEY (`cliente\_id`)

    REFERENCES `rotisdb`.`tb\_clientes` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 9

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_tipoitem`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_tipoitem` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_tipoitem` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `tipo` VARCHAR(255) NOT NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 4

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_estoque`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_estoque` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_estoque` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `descricao` VARCHAR(255) NOT NULL,

  `quantidade` INT(11) NOT NULL,

  `preco\_unitario` DECIMAL(10,2) NULL DEFAULT NULL,

  `preco\_venda` DECIMAL(10,2) NULL DEFAULT NULL,

  `tipo\_id` INT(11) NOT NULL,

  `categoria\_id` INT(11) NOT NULL,

  `ativado` TINYINT(1) NULL DEFAULT NULL,

  PRIMARY KEY (`id`),

  INDEX `categoria\_id` (`categoria\_id` ASC) VISIBLE,

  INDEX `tipo\_id` (`tipo\_id` ASC) VISIBLE,

  CONSTRAINT `tb\_estoque\_ibfk\_1`

    FOREIGN KEY (`categoria\_id`)

    REFERENCES `rotisdb`.`tb\_categorias` (`id`),

  CONSTRAINT `tb\_estoque\_ibfk\_2`

    FOREIGN KEY (`tipo\_id`)

    REFERENCES `rotisdb`.`tb\_tipoitem` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 36

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_funcoes`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_funcoes` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_funcoes` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `descricao` VARCHAR(50) NOT NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 3

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_logs\_login`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_logs\_login` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_logs\_login` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `email` VARCHAR(255) NOT NULL,

  `data\_horario\_acesso` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP(),

  PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 30

DEFAULT CHARACTER SET = utf8mb4;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_preco\_venda`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_preco\_venda` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_preco\_venda` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `id\_estoque` INT(11) NOT NULL,

  `preco\_venda` DECIMAL(10,2) NOT NULL,

  `data\_atualizacao` DATETIME NOT NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 51

DEFAULT CHARACTER SET = utf8mb4;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_precos\_compra`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_precos\_compra` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_precos\_compra` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `estoque\_id` INT(11) NOT NULL,

  `preco\_unitario` DECIMAL(10,2) NOT NULL,

  `data\_atualizacao` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP(),

  PRIMARY KEY (`id`),

  INDEX `estoque\_id` (`estoque\_id` ASC) VISIBLE,

  CONSTRAINT `tb\_precos\_compra\_ibfk\_1`

    FOREIGN KEY (`estoque\_id`)

    REFERENCES `rotisdb`.`tb\_estoque` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 52

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_receitas`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_receitas` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_receitas` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `produto\_final\_id` INT(11) NOT NULL,

  `ingrediente\_id` INT(11) NOT NULL,

  `quantidade\_necessaria` DECIMAL(10,2) NOT NULL,

  PRIMARY KEY (`id`),

  INDEX `produto\_final\_id` (`produto\_final\_id` ASC) VISIBLE,

  INDEX `ingrediente\_id` (`ingrediente\_id` ASC) VISIBLE,

  CONSTRAINT `tb\_receitas\_ibfk\_1`

    FOREIGN KEY (`produto\_final\_id`)

    REFERENCES `rotisdb`.`tb\_estoque` (`id`),

  CONSTRAINT `tb\_receitas\_ibfk\_2`

    FOREIGN KEY (`ingrediente\_id`)

    REFERENCES `rotisdb`.`tb\_estoque` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 9

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

-- -----------------------------------------------------

-- Table `rotisdb`.`tb\_usuarios`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `rotisdb`.`tb\_usuarios` ;

CREATE TABLE IF NOT EXISTS `rotisdb`.`tb\_usuarios` (

  `id` INT(11) NOT NULL AUTO\_INCREMENT,

  `nome` VARCHAR(255) NOT NULL,

  `email` VARCHAR(255) NOT NULL,

  `senha` VARCHAR(255) NOT NULL,

  `funcao\_id` INT(11) NOT NULL,

  PRIMARY KEY (`id`),

  UNIQUE INDEX `uc\_email` (`email` ASC) VISIBLE,

  INDEX `funcao\_id` (`funcao\_id` ASC) VISIBLE,

  CONSTRAINT `tb\_usuarios\_ibfk\_1`

    FOREIGN KEY (`funcao\_id`)

    REFERENCES `rotisdb`.`tb\_funcoes` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 22

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_unicode\_ci;

USE `rotisdb` ;

-- -----------------------------------------------------

-- procedure SP\_Registra\_Acesso

-- -----------------------------------------------------

USE `rotisdb`;

DROP procedure IF EXISTS `rotisdb`.`SP\_Registra\_Acesso`;

DELIMITER $$

USE `rotisdb`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_Registra\_Acesso`(IN email VARCHAR(255))

BEGIN

  INSERT INTO tb\_logs\_login (email, data\_horario\_acesso)

    VALUES (email, NOW());

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure SP\_Registra\_Alteracao\_Preco\_Custo

-- -----------------------------------------------------

USE `rotisdb`;

DROP procedure IF EXISTS `rotisdb`.`SP\_Registra\_Alteracao\_Preco\_Custo`;

DELIMITER $$

USE `rotisdb`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_Registra\_Alteracao\_Preco\_Custo`(IN id\_estoque INT,IN preco\_antigo DECIMAL(10,2))

BEGIN

    INSERT INTO tb\_precos\_compra (estoque\_id, preco\_unitario, data\_atualizacao)

    VALUES (id\_estoque, preco\_antigo, NOW());

END$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure SP\_Registra\_Preco\_Venda

-- -----------------------------------------------------

USE `rotisdb`;

DROP procedure IF EXISTS `rotisdb`.`SP\_Registra\_Preco\_Venda`;

DELIMITER $$

USE `rotisdb`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_Registra\_Preco\_Venda`(IN id\_estoque INT, IN preco\_venda DECIMAL(10,2))

BEGIN

  INSERT INTO tb\_preco\_venda (id\_estoque, preco\_venda, data\_atualizacao)

    VALUES (id\_estoque, preco\_venda, NOW());

END$$

DELIMITER ;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

USE `rotisdb`;

DELIMITER $$

USE `rotisdb`$$

DROP TRIGGER IF EXISTS `rotisdb`.`TRG\_Registra\_Preco\_Unitario` $$

USE `rotisdb`$$

CREATE

DEFINER=`root`@`localhost`

TRIGGER `rotisdb`.`TRG\_Registra\_Preco\_Unitario`

BEFORE UPDATE ON `rotisdb`.`tb\_estoque`

FOR EACH ROW

BEGIN

  IF OLD.preco\_unitario <> NEW.preco\_unitario THEN

      CALL SP\_Registra\_Alteracao\_Preco\_Custo(OLD.id, OLD.preco\_unitario);

    END IF;

END$$

USE `rotisdb`$$

DROP TRIGGER IF EXISTS `rotisdb`.`TRG\_Registra\_Preco\_Venda` $$

USE `rotisdb`$$

CREATE

DEFINER=`root`@`localhost`

TRIGGER `rotisdb`.`TRG\_Registra\_Preco\_Venda`

BEFORE UPDATE ON `rotisdb`.`tb\_estoque`

FOR EACH ROW

BEGIN

  IF OLD.preco\_venda <> NEW.preco\_venda THEN

      CALL SP\_Registra\_Preco\_Venda(OLD.id, OLD.preco\_venda);

    END IF;

END$$

USE `rotisdb`$$

DROP TRIGGER IF EXISTS `rotisdb`.`before\_update\_estoque` $$

USE `rotisdb`$$

CREATE

DEFINER=`root`@`localhost`

TRIGGER `rotisdb`.`before\_update\_estoque`

BEFORE UPDATE ON `rotisdb`.`tb\_estoque`

FOR EACH ROW

BEGIN

    -- Evitar alteração do campo 'ativado' quando outros campos estão sendo atualizados

    IF OLD.preco\_unitario = NEW.preco\_unitario AND OLD.preco\_venda = NEW.preco\_venda THEN

        -- Verificar se o item de estoque está relacionado a agendamentos ativos

        IF EXISTS (

            SELECT 1

            FROM tb\_agendamentos AS a

            JOIN tb\_receitas AS r ON a.receita\_id = r.id

            WHERE (r.produto\_final\_id = OLD.id OR r.ingrediente\_id = OLD.id)

              AND a.status\_id IN (SELECT id FROM tb\_status WHERE descricao IN ('Em Andamento', 'Finalizado'))

        ) THEN

            -- Atualizar o atributo ativado para 0 (desativado) em vez de excluir

            SET NEW.ativado = 0;

            -- Definir uma mensagem de erro para o controle da aplicação

            SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = 'Nao foi possivel desativar. Possivelmente existem agendamentos com este item de estoque.';

        END IF;

    END IF;

END$$

DELIMITER ;

### Procedures

-- Procedure para registrar tempo de acesso de usuario.

DELIMITER $$

CREATE PROCEDURE SP\_Registra\_Acesso(IN email VARCHAR(255))

BEGIN

    INSERT INTO tb\_logs\_login (email, data\_horario\_acesso)

    VALUES (email, NOW());

END $$

DELIMITER ;

```

```sql

-- Procedure para registrar atualizacao de estoque no atributo de preco\_unitario.

DELIMITER $$

CREATE PROCEDURE SP\_Registra\_Alteracao\_Preco\_Custo(IN id\_estoque INT,IN preco\_antigo DECIMAL(10,2))

BEGIN

    INSERT INTO tb\_precos\_compra (estoque\_id, preco\_unitario, data\_atualizacao)

    VALUES (id\_estoque, preco\_antigo, NOW());

END $$

DELIMITER ;

-- Procedure para registrar preco de venda ao atualizar de estoque.

delimiter $$

CREATE PROCEDURE SP\_Registra\_Preco\_Venda(IN id\_estoque INT, IN preco\_venda DECIMAL(10,2))

BEGIN

    INSERT INTO tb\_preco\_venda (id\_estoque, preco\_venda, data\_atualizacao)

    VALUES (id\_estoque, preco\_venda, NOW());

END $$

DELIMITER ;

### Triggers

-- Chama trigger para registrar preco de compra na tb\_compras

DELIMITER $$

CREATE TRIGGER TRG\_Registra\_Preco\_Unitario BEFORE UPDATE

ON tb\_estoque

FOR EACH ROW

BEGIN

    IF OLD.preco\_unitario <> NEW.preco\_unitario THEN

        CALL SP\_Registra\_Alteracao\_Preco\_Custo(OLD.id, OLD.preco\_unitario);

    END IF;

END $$

DELIMITER ;

-- Chama trigger para registrar preco\_venda na tabela precos\_venda

DELIMITER $$

CREATE trigger TRG\_Registra\_Preco\_Venda BEFORE UPDATE

ON tb\_estoque

FOR EACH ROW

BEGIN

    IF OLD.preco\_venda <> NEW.preco\_venda THEN

        CALL SP\_Registra\_Preco\_Venda(OLD.id, OLD.preco\_venda);

    END IF;

END $$

DELIMITER ;

-- Trigger acionada para verificar se ira deletar estoque

DROP TRIGGER IF EXISTS before\_update\_estoque;

DELIMITER $$

CREATE TRIGGER before\_update\_estoque

BEFORE UPDATE ON tb\_estoque

FOR EACH ROW

BEGIN

    -- Evitar alteração do campo 'ativado' quando outros campos estão sendo atualizados

    IF OLD.preco\_unitario = NEW.preco\_unitario AND OLD.preco\_venda = NEW.preco\_venda THEN

        -- Verificar se o item de estoque está relacionado a agendamentos ativos

        IF EXISTS (

            SELECT 1

            FROM tb\_agendamentos AS a

            JOIN tb\_receitas AS r ON a.receita\_id = r.id

            WHERE (r.produto\_final\_id = OLD.id OR r.ingrediente\_id = OLD.id)

              AND a.status\_id IN (SELECT id FROM tb\_status WHERE descricao IN ('Em Andamento', 'Finalizado'))

        ) THEN

            -- Atualizar o atributo ativado para 0 (desativado) em vez de excluir

            SET NEW.ativado = 0;

            -- Definir uma mensagem de erro para o controle da aplicação

            SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = 'Nao foi possivel desativar. Possivelmente existem agendamentos com este item de estoque.';

        END IF;

    END IF;

END$$

DELIMITER ;

### Transactions

Foi utilizado transactions nas tabelas de clientes e usuario, pois nelas, ao terminar o cadastro de usuario, se captura o id dele (último cadastrado) e insere na tabela de endereços, que corresponde a uma tentativa de normalização dos dados do banco.

Abaixo seguem as transactions em PHP, que se tratam do tipo implícitas:

--### Transactions

    public function adicionarCliente($dados) {

    try{

        // Definindo a query de inserção

        $this->conexao->beginTransaction();

        $query = "INSERT INTO tb\_clientes (nome, email, telefone) VALUES (:nome, :email, :telefone);";

        $stmt = $this->conexao->prepare($query);

        $stmt->bindParam(":nome", $dados['nome'], PDO::PARAM\_STR);

        $stmt->bindParam(":email", $dados['email'], PDO::PARAM\_STR);

        $stmt->bindParam(":telefone", $dados['telefone'], PDO::PARAM\_STR);

        $stmt->execute();

        $cliente\_id = $this->conexao->lastInsertId();

        $queryEndereco = "INSERT INTO tb\_endereco (cep, rua, numero, cidade, estado, bairro, cliente\_id) VALUES (:cep, :rua, :numero, :cidade, :estado, :bairro, :cliente\_id);";

        $stmt = $this->conexao->prepare($queryEndereco);

        // Associando os parâmetros

        $stmt->bindParam(":cep", $dados['cep'], PDO::PARAM\_STR);

        $stmt->bindParam(":rua", $dados['rua'], PDO::PARAM\_STR);

        $stmt->bindParam(":numero", $dados['numero'], PDO::PARAM\_STR);

        $stmt->bindParam(":cidade", $dados['cidade'], PDO::PARAM\_STR);

        $stmt->bindParam(":estado", $dados['estado'], PDO::PARAM\_STR);

        $stmt->bindParam(":bairro", $dados['bairro'], PDO::PARAM\_STR);

        $stmt->bindParam(":cliente\_id", $cliente\_id, PDO::PARAM\_INT);

        // Executando a consulta

        if($stmt->execute()){

            $this->conexao->commit();

            return true;

        }

    }catch (Exception $e) {

            // Em caso de erro, reverte a transação

            $this->conexao->rollBack();

            throw new Exception("Erro ao cadastrar cliente e endereço: " . $e->getMessage());

        }

    }

     public function atualizarCliente($dados){

        // Iniciar transação

        $this->conexao->beginTransaction();

        try{

            $query = "UPDATE tb\_clientes SET nome = :nome, email = :email, telefone = :telefone WHERE id = :id\_cliente;";

            $stmt\_cliente = $this->conexao->prepare($query);

            $stmt\_cliente->bindParam(":nome", $dados['nome'], PDO::PARAM\_STR);

            $stmt\_cliente->bindParam(":email", $dados['email'], PDO::PARAM\_STR);

            $stmt\_cliente->bindParam(":telefone", $dados['telefone'], PDO::PARAM\_STR);

            $stmt\_cliente->bindParam(":id\_cliente", $dados['id\_cliente'], PDO::PARAM\_INT);

            $stmt\_cliente->execute();

            // Atualizar a tabela tb\_endereco

            $queryendereco = " UPDATE tb\_endereco SET cep = :cep, rua = :rua, numero = :numero, cidade = :cidade, estado = :estado, bairro = :bairro WHERE cliente\_id = :id\_cliente";

            $stmtendereco = $this->conexao->prepare($queryendereco);

            $stmtendereco->bindParam(":cep", $dados['cep'], PDO::PARAM\_STR);

            $stmtendereco->bindParam(":rua", $dados['rua'], PDO::PARAM\_STR);

            $stmtendereco->bindParam(":numero", $dados['numero'], PDO::PARAM\_STR);

            $stmtendereco->bindParam(":cidade", $dados['cidade'], PDO::PARAM\_STR);

            $stmtendereco->bindParam(":estado", $dados['estado'], PDO::PARAM\_STR);

            $stmtendereco->bindParam(":bairro", $dados['bairro'], PDO::PARAM\_STR);

            $stmtendereco->bindParam(":id\_cliente", $dados['id\_cliente'], PDO::PARAM\_INT);

            $stmtendereco->execute();

            // Commit da transação

            $this->conexao->commit();

            // Retorna true se as duas instruções executaram corretamente

            return true;

        }catch (Exception $e) {

            // Em caso de erro, realiza o rollback

            $this->conexao->rollBack();

            // Exibe a mensagem de erro para depuração (pode remover ou registrar em produção)

            echo "Erro: " . $e->getMessage();

            return false;

        }

    }

## Recursos e ferramentas (1º e 2º)

Os recursos e ferramentas utilizados no projeto possui diversas versões e especificações, distribuídos entre frontend, backend e ferramentas complementares.

Para o desenvolvimento Backend, os seguintes recursos com suas respectivas versões:

|  |  |  |
| --- | --- | --- |
| **Recurso** | **Versão** | **Observação** |
| Linguagem de Programação PHP | 8.1.4 | Mínimo utilizado no projeto contempla a versão 8. |
| Banco de Dados | 10.4.24-MariaDB | Usamos o MariaDB, semelhante ao MySQL. |
| Versão do Servidor | Apache/2.4.53 (Win64) OpenSSL/1.1.1n | - |
| Desenvolvimento e codificação: Visual Studio Code | 1.95.3 | Considera-se apenas a versão final, sendo que o programa passou por atualizações no decorrer do desenvolvimento. |

Já, para o desenvolvimento frontend, os seguintes recursos foram utilizados:

|  |  |  |
| --- | --- | --- |
| **Recurso** | **Versão** | **Observação** |
| HTML | Versão 5(HTML5) | - |
| CSS | Versão 3 (CSS3) | - |
| JavaScript | Versão 2024 | - |
| Figma | Web mais recente. 2024 | - |

Por fim, para a realização de testes do projeto, foram utilizadas as seguintes ferramentas:

|  |  |  |
| --- | --- | --- |
| **Recurso** | **Versão** | **Observação** |
| SonarLint | 4.12.0 | Uma extensão gratuita no VSCode que aponta erros e melhorias em códigos de diversas linguagens, incluindo o PHP. |
| DevTools -> Aba Performance | Brave 1.73.91 | O próprio navegador possui suas abas de teste e performance para a aplicação. |

E, por último, recursos e softwares relacionados a documentação estão apresentados abaixo:

|  |  |  |
| --- | --- | --- |
| **Recurso** | **Versão** | **Observação** |
| Visual Paradigm | Online (mais recente) – não verificável. | Sem restrição de versão. |
| Mermaid Editor | Online (mais recente) – não verificável. | Sem restrição de versão. |
| Draw.io | Online (mais recente) – não verificável | Sem restrição de versão. |
| Word | 2024 – 2408 | Sem restrição de versão. |
| Excel | 2024 – 2408 | Sem restrição de versão. |

## Etapas / Sprints realizados (1º e 2º)

**1º Sprint Mês 09 - Mês 10)**

* **Objetivos do Sprint:**
  + Definir requisitos iniciais do sistema.
  + Criar a estrutura inicial do banco de dados (tabelas, relacionamentos).
  + Alinhamento sobre Design
* **Tarefas concluídas:**
  + Cores, marcas, códigos básicos do banco de dados, códigos básicos do PHP.
* **Desafios enfrentados:**
  + Modelagem do Banco para atender necessidades do cliente
  + Requisitos de agendamento
  + Tentativa de estruturação do sistema em MVC.

**1º Sprint Mês 11 - Mês 12)**

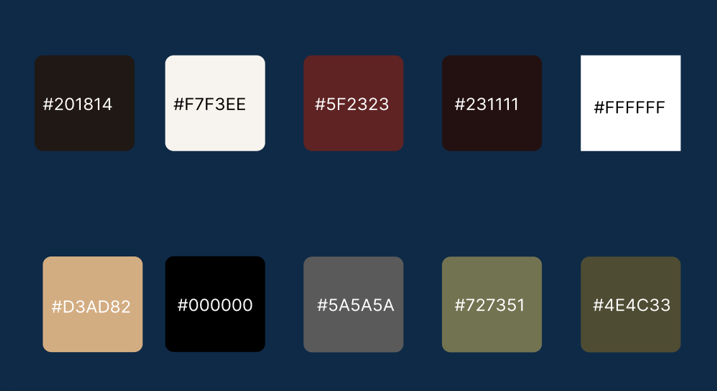
* **Objetivos do Sprint:**
  + Finalizar requisitos iniciais do sistema.
  + Finalizar a estrutura do banco de dados (tabelas, relacionamentos).
  + Finalizar integração entre Frontend E Backend
* **Tarefas concluídas:**
  + Integração, testes,responsividade e lógica completa do backend.
* **Desafios enfrentados:**
  + Adaptação do MVC ao projeto
  + Adaptação do banco ao longo do desenvolvimento
  + Mudanças de interface e refatoração do backend para lógica dos requisitos funcionais.

## 3.8 Interface do usuário

A interface do usuário foi elaborada, principalmente, considerando o aspecto visual do cartão fornecida pelo cliente, onde não temos a necessidade entrar em contato com o cliente dela, já que este é um sistema de gerenciamento dela, de uso restrito e interno.

## 3.8.1 Esquema de Cores

As cores do sistema foram elaboradas, seguindo a seguinte paleta:

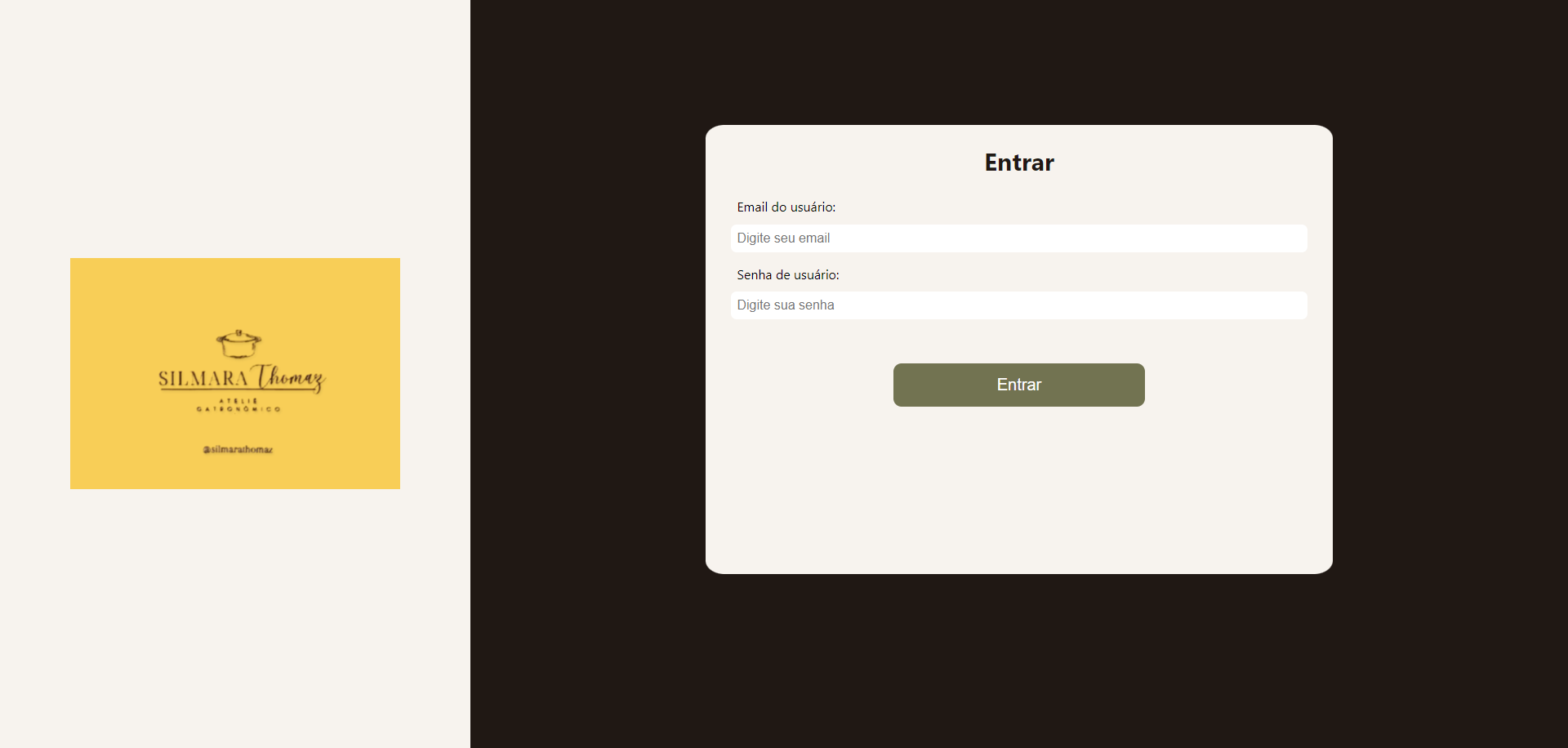


As cores escolhidas para o sistema de rotisseria foram selecionadas com base em sua capacidade de criar uma atmosfera acolhedora e transmitir qualidade. O preto escuro sugere sofisticação e elegância, enquanto o bege claro e o branco trazem uma sensação de limpeza e simplicidade. O vermelho escuro é associado ao apetite e ao calor, ideal para um ambiente alimentar. O cinza e o verde oliva são neutros e equilibram a paleta, com o verde simbolizando frescor e alimentos naturais. Essas cores, em conjunto, ajudam a criar um design que transmite confiança, qualidade e acolhimento, essenciais para o ambiente de uma rotisseria.

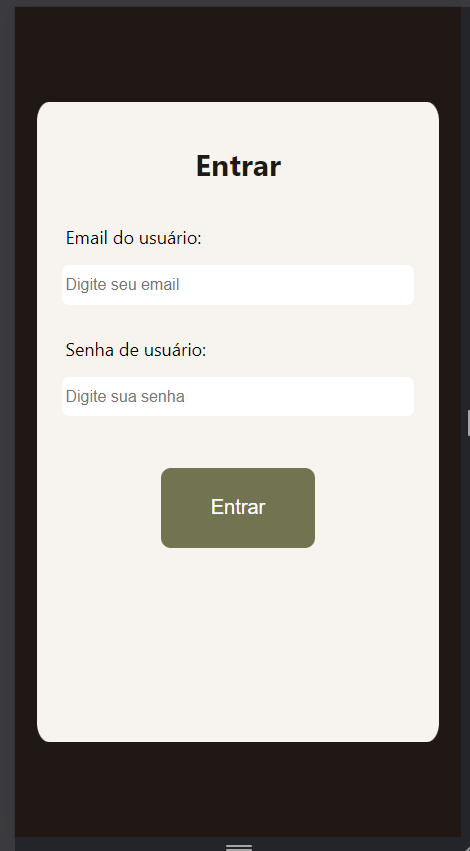
## 3.8.2 Telas finais

Seguem abaixo as telas desenvolvidas para o sistema considerando todos os pontos levantados anteriormente para sua elaboração:

**Tela 1 – Login (desktop)**



**Tela 1 – Login (mobile)**

****

**Tela 2 – Dashboard (desktop)**

**Tela 2 – Dashboard (mobile)**

**Tela 3 – Estoque (desktop)**

**Tela 3 – Estoque (mobile)**

**Tela 4 – Agendamentos (desktop)**

**Tela 4 – Agendamentos (mobile)**

**Tela 5 – Cadastrar Cliente (desktop)**

**Tela 5 – Cadastrar Cliente (mobile)**

**Tela 6 – Cadastrar Usuário (desktop)**

**Tela 6 – Cadastrar Usuário (mobile)**

# Testes e Qualidade

## Estratégia de Testes: Descrever a estratégia de testes adotada

A estratégia de testes do projeto consiste em utilizar como gerenciamento de projeto o GitHub, utilizando guias de commits e contribuições para orientação do grupo, buscando seguir um padrão de desenvolvimento, seja em código, documentação e comentários.

O GitHub foi utilizado como meio de hospedagem e compartilhamento de projetos, com os testes e implementações ocorrendo em Branches separadas, nomeadas de acordo com sua funcionalidade ou problema implementados/resolvidos. Basicamento foi aplicado a ideia de teste de mesa, conhecendo o resultado final e com inputs variados, foram aplicados e validados durante a implementação.

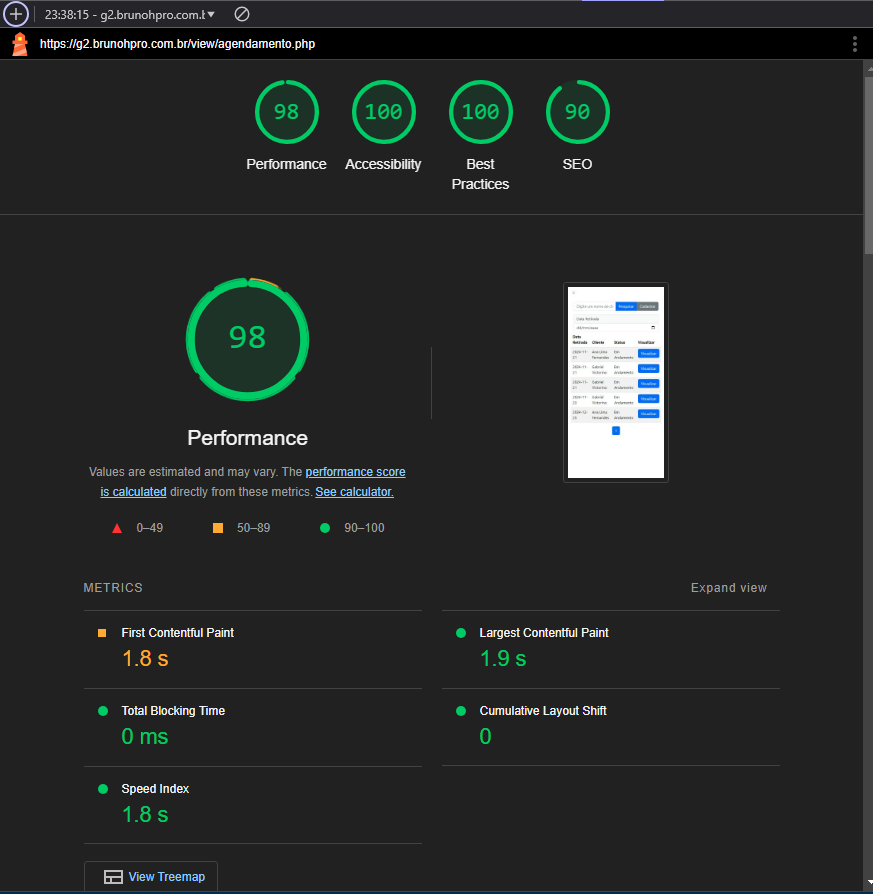
Outra ideia para teste foi o uso de ferramentas para medir a velocidade de carregamento e o próprio DevTools do Chrome, com a guia LightHouse sendo utilizada para verificar a boas práticas, performance e acessibilidade.

Em relação aos testes de banco de dados, o grupo implementou as ferramentas.

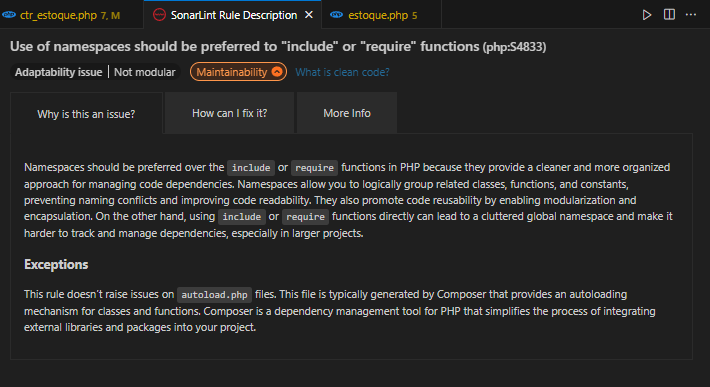
## 4.2 Resultados dos Testes: Apresentar os resultados dos testes realizados

Os resultados dos testes em gerais foram positivos, conforme se apresentam nos prints, realizado no site hospedado na nuvem. No entanto, compreende **ressaltar que os testes foram realizados com uma pequena quantidade de dados cadastrados na base.**

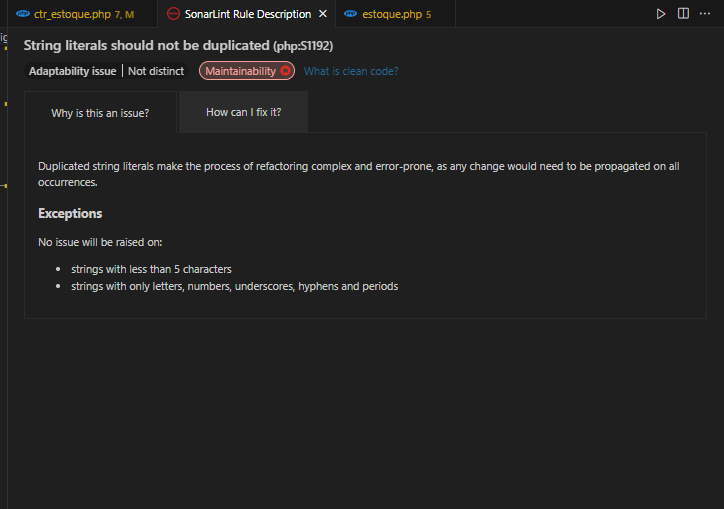
**Abaixo é listado um teste realizado com o DevTools**



De acordo com o que foi visto, na pequena escala de desenvolvimento, em geral, o site apresenta bom carregamento das informações e um bom frontend e adaptabilidade. Outros testes realizados foram o Sonar Lint, no entanto, as sugestões fornecidas pela extensão, compreendem um conhecimento maior de codificação e conhecimento técnico, por isso, a ferramenta foi utilizada, como caráter consultivo. Confira abaixo um exemplo de resultado:



A extensão é gratuita e está disponível no VsCode, com cárater apenas consultivo pare melhorias pontuais no código.



Uma questão interessante do SonarLint, é a categorização do contexto que ele aponta, considerando, como nas imagens anteriores, que a correção implica numa melhoria de Manutenabilidade do código.

Portanto, para ferramentas de testes, foram utilizadas principalmente estas duas, além de boas práticas aprendidas durante a aula de PHP da matéria de desenvolvimento web 2.

Por fim, vale ressaltar que diversas melhorias podem ser aplicadas, como seguir mais rigorosamente as **PSRs** (PHP Standards Recommendations), melhor uso da ferramenta SonarLint, buscando um conhecimento mais aprofundado de determinados conceitos de programação do sistema.

## 4.3 Garantia da Qualidade: Descrever as práticas adotadas para garantia da qualidade

Considerando o aprendizado do grupo, e que o cohecimento mais aprofundado em relação a testes com códigos é muito amplo, o grupo trabalhou com os seguintes itens para teste:

- **Testes de Funcionalidade:**

* **Exemplo:** Realizando testes de mesa, de código, usando ferramentas de teste, como SonarLint, além de conversar com especialistas da área, como os professores. Buscando a melhor implementação.

**- Revisão de Código:**

* **Exemplo:** Implementar revisões de código por parte da equipe antes de qualquer código ser mesclado ao repositório principal.

**- Controle de Versionamento:**

* **Exemplo:** Utilizar Git para controlar versões do código e evitar conflitos.

**- Testes de Usabilidade:**

* **Exemplo:** Realizar testes com usuários reais para verificar se a interface do sistema é intuitiva e atende às necessidades dos usuários finais. Além de ferramentas, como o próprio teste do navegador, citado na sessão anterior.

**- Feedback Contínuo:**

* **Exemplo:** Manter canais de comunicação abertos com os stakeholders e clientes para obter feedback constante sobre o produto, tudo isso com entrevistas e mensagens.

## 4.4 Requisitos mínimos de hardware e software para o sistema

Os requisitos mínimos de software e de sistema estão correspondidos na tabela abaixo:

**Requisitos de Software.**

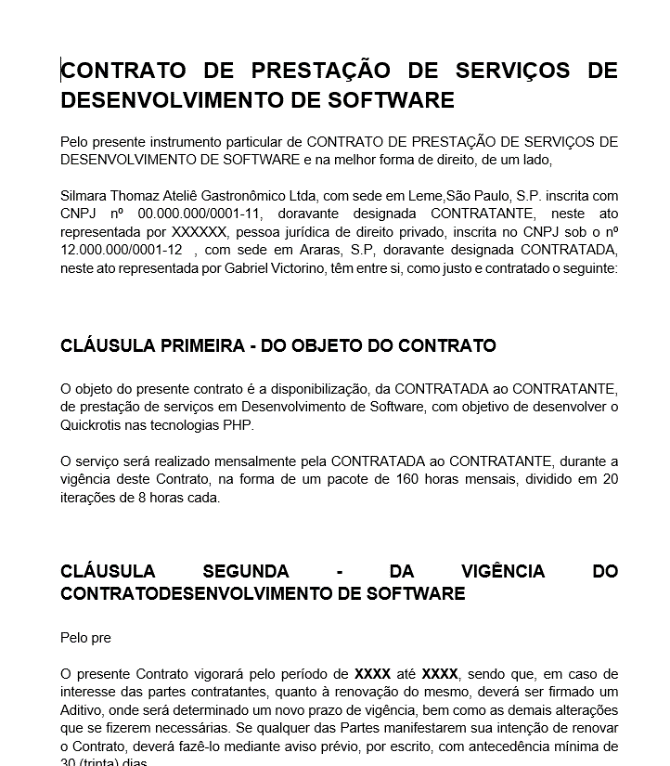
|  |  |  |
| --- | --- | --- |
| **Requisito** | **Mínimo Aceitável** | **Observações** |
| Sistema Operacional | Windows 10 ou Superior. | O sistema foi aplicado em um Windows 10 Pro 22H2 e Windows 11Pro 23H2. |
| Xampp | Xampp com a versão mais recente disponível: 8.2.12 | O xampp serve para rodar o projeto localmente, sem conexão com a internet. |
| Rotina do próprio Windows para iniciar Xampp automaticamente. | Ativado. | -- |

**Requisitos de Hardware**

|  |  |  |
| --- | --- | --- |
| **Requisito** | **Mínimo Aceitável** | **Observações** |
| Monitor | Resolução padrão (1920;1080). | Considera-se aceitável o mínimo de uma resolução de (1440;900) |
| Armazenamento | SSD ou na internet. | Considera-se o SSD para a abertura rápida do sistema operacional e carregamento de seus serviços. |
| Processamento | Qualquer processador com a performance semelhante ao utilizado no desenvolvimento: Ryzen 3 3200g, 3.6GHz, aceitando ser também, 2.6GHz. | O processamento pode variar dependendo de como o sistema operacional do cliente está configurado e carregado. Mas em testes, apenas executando o sistema, este requisito independe no impacto final da performance. |
| Demais componentes | Teclado, mouse e internet. | -- |

## 4.5 Contrato para desenvolvimento de software

Foi elaborado um contrato de desenvolvimento de software como exemplo entre a cliente a equipe de desenvolvimento, apenas para nos aproximais mais da realidade do ciclo de desenvolvimento de software.



Algumas considerações deste contrato são:

* Transferência de Tecnologia para a cliente (ela usaria o sistema e teria acesso a como foi desenvolvido)
* Definição do tempo de trabalho da equipe de desenvolvimento.

# Considerações Finais

Durante todo o desenvolvimento do projeto, o grupo aprendeu e descobriu que o desenvolvimento de um projeto maior e envolvendo diversos componentes, exige uma boa comunicação entre as partes, o uso adequado e correto do compartilhamento no GitHub, a organização da estrutura do projeto, criando pastas e mantendo a documentação atualizada, pelo menos, a cada quinzena, para assim todos compreenderem para onde o projeto está caminhando. Também foi visto que a realização de alinhamentos com reuniões foram positivos e mais eficientes do que somente uma comunicação por mensagens. Já analisando a habilidade em bancos de dados, foi mais compreensível o uso das triggers, procedures e as consultas mais complexas, como usando os JOINS.

Em relação as habilidades de comunicação com o cliente, o projeto permitiu compreender o que seria mais próximo de um projeto real, entendendo a dificuldade de um cliente visualizar o sistema e do engenheiro/componente compreender as regras do negócio do cliente. Além disso, o conhecimento e aprimoramento em testes, e complexidade de código foram essenciais para implantar o sistema e para compreensão para os componentes do grupo.

Portanto, o projeto foi enriquecedor em todos os aspectos que abrangeu, fornecendo uma grande compreensão em diversas matérias e permitindo enxergar diversas maneiras de chegar no resultado final, explorando além de diversas habilidades individuais de cada membro da equipe.

Das contribuições Individuais, resumidamente, foram:

Gabriel Victorino – Gerenciamento da equipe, desenvolvimento backend, design, frontend e alinhamento direto com entrevistas e reuniões com a cliente.

Gustavo H. – Desenvolvimento Frontend, reuniões com a cliente e apresentação do projeto.

João Vitor – codificação backend, documentação e apresentação do projeto.

Rafael Botezelli – documentação do projeto, desenvolvimento do contrato de software e apresentação do projeto.

# 6 Referências

**Livro: Sommerville - Engenharia de Software**

SOMMERVILLE, Ian. *Engenharia de Software*. 10. ed. São Paulo: Pearson, 2018.

**Site: IBM - Diagramas de Classes**

IBM. *RSAS 7.5.0 - Estrutura dos Diagramas de Classe*. Disponível em: <https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=structure-class-diagrams>. Acesso em: 18 nov. 2024.

**Site: IBM - Diagramas de Sequência UML**

IBM. *RSMS 7.5.0 - Diagramas de Sequência UML*. Disponível em: <https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=uml-sequence-diagrams>. Acesso em: 18 nov. 2024.

**Site: Sines Contábil - Contrato de Licença de Uso**

SINES CONTÁBIL. *Contrato de Licença de Uso e Prestação de Serviços de Software*. Disponível em: <https://www.sinescontabil.com.br/modelos/modelos/modelosdc/contrato_de_licenca_de_uso_e_prestacao_de_servicos_de_software.htm>. Acesso em: 18 nov. 2024.

**Site: UX Design - Sites de Paletas de Cores**

UX DESIGN BRASIL. *Os 9 Sites de Paletas de Cores que Todo UI Designer Precisa Conhecer*. Disponível em: <https://brasil.uxdesign.cc/os-9-sites-de-paletas-de-cores-que-todo-ui-designer-precisa-conhecer-3742480ef761>. Acesso em: 18 nov. 2024.

**Site: DevMedia - Técnicas e Fundamentos de Testes de Software**

DEVMEDIA. *Técnicas e Fundamentos de Testes de Software*. Disponível em: <https://www.devmedia.com.br/guia/tecnicas-e-fundamentos-de-testes-de-software/34403>. Acesso em: 18 nov. 2024.

# Anexo I - Diário de bordo

**Físico**

Abaixo se encontra o link para o arquivo .pdf do diário de bordo geral, pois há muitas páginas para serem inseridas individualmente.

[Link](https://github.com/GabrielVictorino8266/pi_2_semestre/tree/main/docs/diarios) - Diário de Bordo Geral

**Digital**

Abaixo é apresentado os diários de bordo de cada integrante com o respectivo link do repositório.

Gabriel Victorino – [Link](https://github.com/GabrielVictorino8266/pi_2_semestre/blob/main/docs/diarios/gabriel-victorino.md)

Gustavo Francisco Habermann – Link

João Vitor de Camargo – Link

Rafael Botezelli – Link

# Anexo II – Cronograma efetivo

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DATAS** | **22/08/2024 - 31/08/2024** | **01/09/2024 - 15/09/2024** | **16/09/2024 - 30/09/2024** | **01/10/2024 - 15/10/2024** | **16/10/2024 - 30/10/2024** | **01/11/2024 - 15/11/2024** | **16/11/2024 - 31/11/2024** | **03/12/2024** |
| **Definição Grupos** | **finished** |  |  |  |  |  |  |  |
| **Criação Empresa** |  | **finished** |  |  |  |  |  |  |
| **Escopo Sistema** |  |  | **finished** |  |  |  |  |  |
| **Requisitos** |  |  |  | **finished** |  |  |  |  |
| **Diagramas** |  |  |  |  | **finished** | | |  |
| **Protótipo** |  |  |  |  | **finished** | | |  |
| **Documentação** |  |  |  |  | **finished** | | |  |
| **Entrega** |  |  |  |  |  |  | **finished** |  |
| **Apresentação** |  |  |  |  |  |  |  | **finished** |

# Anexo III – Evidências

Abaixo seguem algumas evidências da participação dos integrantes do grupo, considerando os prints, links de reuniões e documentos criados pelos mesmos (Alguns exemplares).

Links de reuniões e formulários google para pesquisa entre membros do projeto:

**Avaliação da ideia inicial (ideia foi mudada ao longo do projeto)** <https://forms.gle/NaLcdAE6p134rs8V6>

**Logotipo**: <https://forms.gle/z4JjLiFeuZeCDTPs8>

**Resultados de formulários:** <https://docs.google.com/document/d/1A1oqQpvxOffy4h_AugkrgyvWg-U3kD6ig5W66h1MooQ/edit>

**Alinhamentos:** <https://docs.google.com/document/d/1H6x5KEFKQ3sXI98-2IN6GErRA8-FVDECpOGqThCkYgo/edit?usp=drivesdk>

**Entrevista:** <https://docs.google.com/document/d/1XPisYCycRQHxI1ZXn3UUgjGwnx4_fjYwVYUNOFp9dLU/edit?usp=drivesdk>

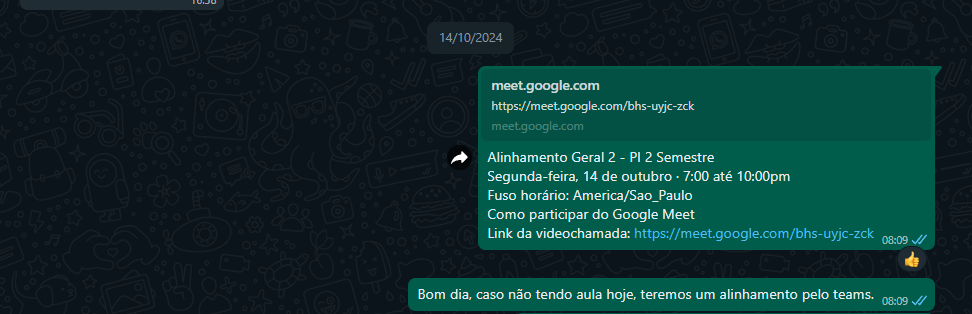
**Nome escolhido para o sistema:** <https://docs.google.com/document/d/145zaLsaNHpUATXLzHggzZJEtASC6q1-8lS36T3luXkw/edit?usp=sharing>

**Reunião**: <https://meet.google.com/gtw-eexi-mef>

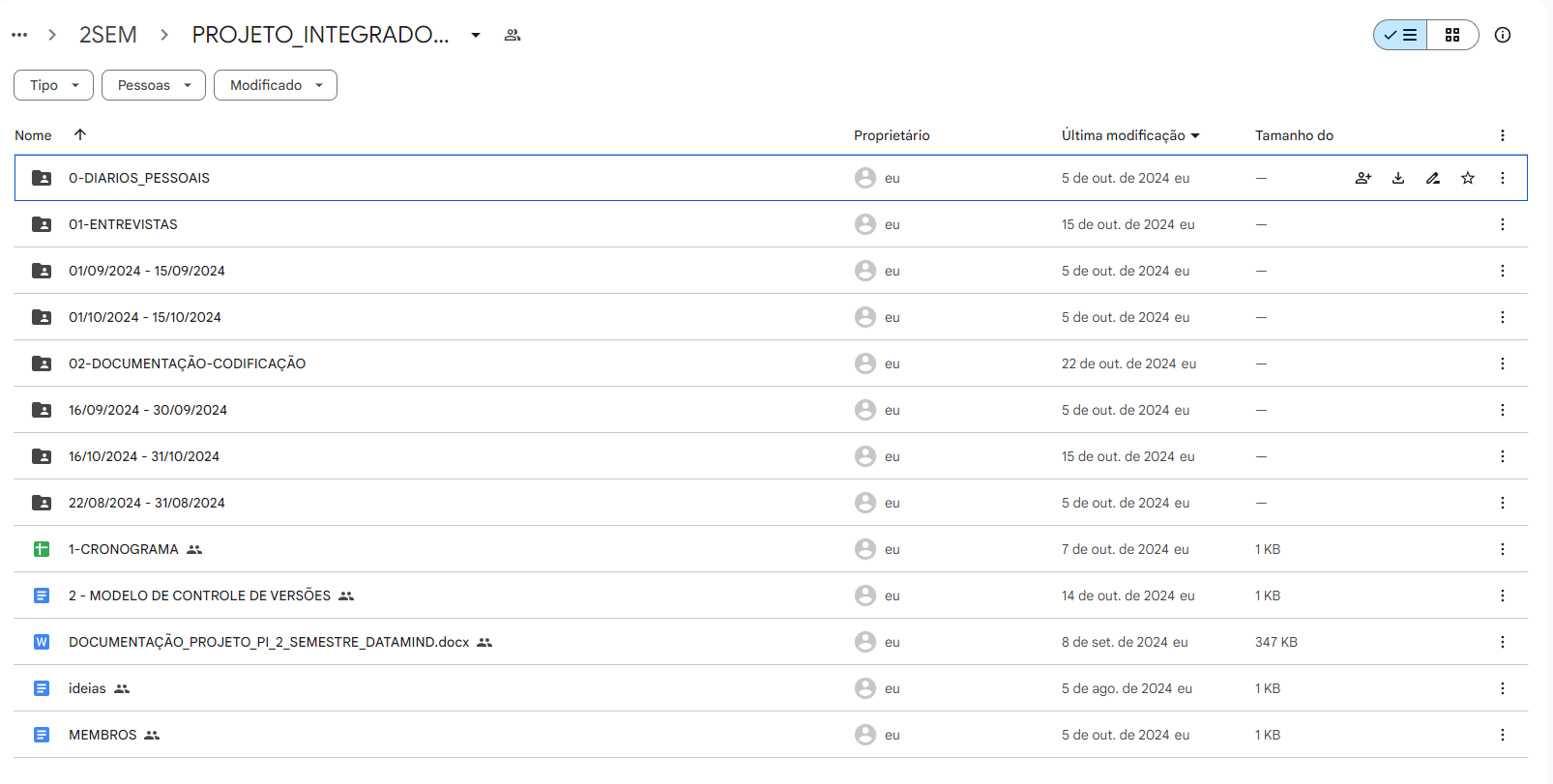
**Entrevista**: <https://docs.google.com/document/d/1XPisYCycRQHxI1ZXn3UUgjGwnx4_fjYwVYUNOFp9dLU/edit?usp=drive_link>

**Reunião:** <https://meet.google.com/vsy-szzz-hdk>

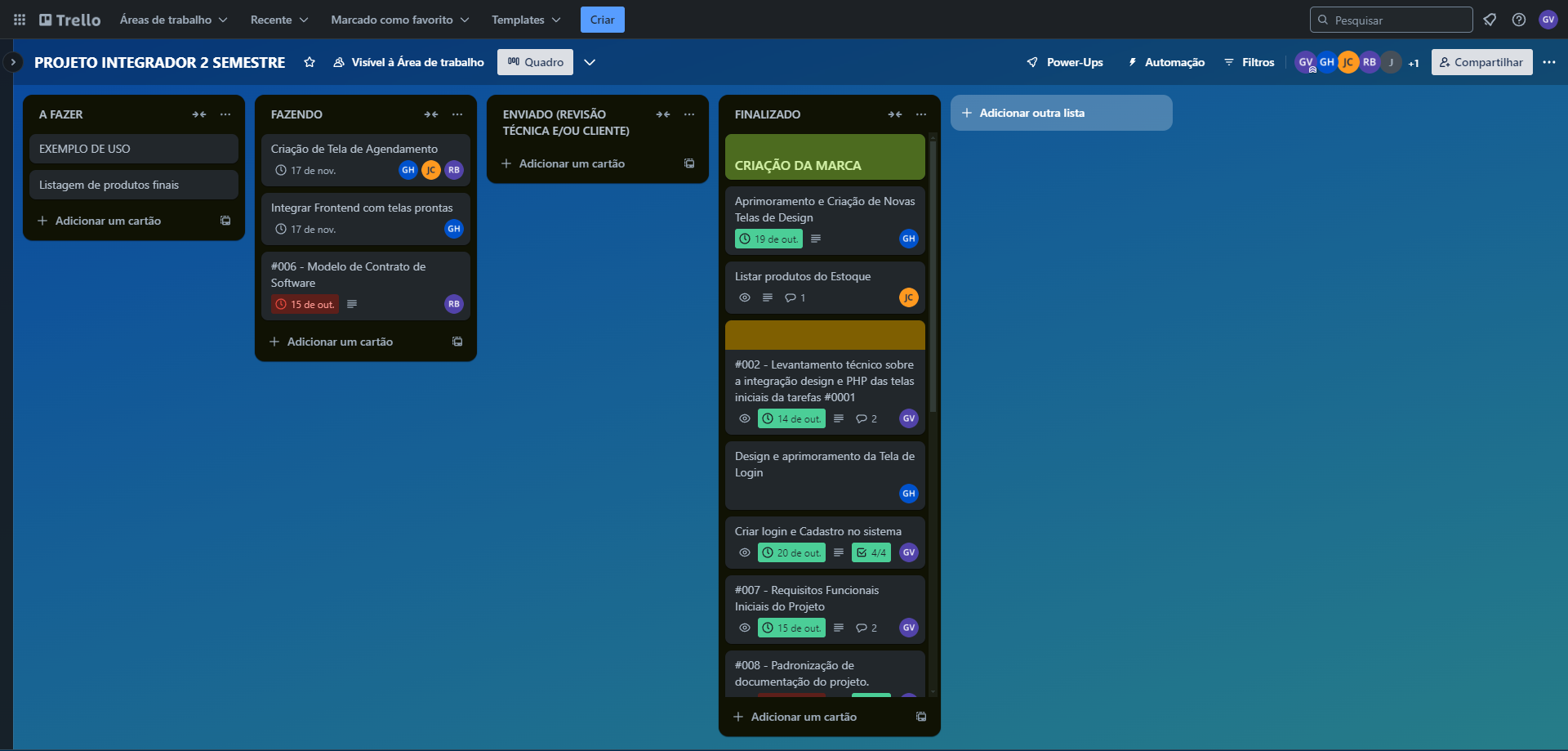
Exemplo de print de alinhamento:



Pasta do Drive para formulários e Pesquisas:



Trello como Evidência



Alinhamento com componente do grupo.

