# PROJETO T.A.L.G.S

## Tia Ana Lanches Gerencial System

### Grupo 3 – Projeto Integrador

Eduardo Rodrigues Estrela - 232016600046
Luis Fernando Soares de Oliveira - 232016600042
Renan Eduardo da Silva Souza - 232016600039

# Índice

## Capítulos e Subseções

# Capítulo 1

## *1.1 – Sobre o Sistema*

- O que é?

O Sistema T.A.L.G.S (Tia Ana Lanches Gerencial System) será usado para o gerenciamento de vendas, estoque e pedidos de uma lanchonete, propriedade de Ana Kátia, localizada em Planaltina - DF, e a ideia principal é criar um sistema de fácil utilização, ultraleve, responsivo e intuitivo.

- O que fará?

O sistema será responsável pela organização e indexação dos estoques, datas de validade e pedidos de abastecimento de estoque, bem como criar e visualizar registros de compras e vendas de produtos e adicionar, remover e editar dados dos produtos que constam no sistema.

- Onde e quando será usado?

O sistema será implementado por meio de uma interface para web, podendo ser acessado de qualquer dispositivo remotamente.

Ele deverá ser usado quando a cliente, por necessidade de organização, realizar operações de compra e venda de produtos, atualização do estoque, adição ou subtração de produtos do catálogo. Em resumo, sua utilização ocorre em momentos em que a cliente deveria guardar, de forma manual e lenta, informações sobre seus produtos, vendas e operações do gênero, para a automação de alguns desses processos, e para a facilitação de outros.

- Existem sistemas que fazem o mesmo?

É notável a existência de sistemas que realizam todas as operações acima descritas, porém, nenhum deles executa todas as funções acima listadas, apenas uma parte delas. Outra característica deles é a implementação por meio de aplicativos locais escritos em linguagens específicas à um sistema operacional ou aparelho eletrônico específicos.

- O diferencial do nosso sistema

O sistema T.A.L.G.S. possui alguns diferenciais dos sistemas já existentes no campo da automação de vendas. Alguns desses diferenciais podem ser listados:

- Acesso multiplataforma por meio da Web
- Atualização de produtos em tempo real
- Gerenciamento de pedidos integrado
- Rápido, otimizado e intuitivo
- Gratuito

# 1.2 – Entrevista

*A cliente do sistema foi entrevistada no dia 11/09/2023, para o levantamento de funcionalidades requeridas do sistema. As seguintes perguntas foram feitas durante a entrevista, seguidas das respectivas respostas:*

- Quais são as necessidades que o sistema precisa cumprir?
  - “O sistema precisa cumprir funcionalidades para registrar produtos e registrar vendas”
- Existe alguma necessidade especial que o sistema precise cumprir?
  - “Relatórios das vendas, produtos, reestocagem é uma necessidade grande para o meu negócio, seria uma mão na roda.”
- O sistema deve operar para gerenciar o estoque da loja?

- “Sim, absolutamente. Gerenciar os produtos dentro da loja já é um desafio grande o suficiente.”

- O sistema deve operar para gerenciar o caixa da loja?

  - “Sim, com certeza.”

Após a entrevista, o que se pode esculpir das necessidades básicas do sistema foi:

| Necessidade N° | Descrição |
|---|---|
| NE 01 | *Melhorar o método de organização do controle de estoque.* |
| NE 02 | *Melhorar o processo de registrar compras para controle futuro.* |
| NE 03 | *Monitorar periodicamente a validade dos produtos no estoque.* |
| NE 04 | *Agilizar o processo de compra por parte do usuário cliente.* |
| NE 05 | *Agilizar o processo de venda por parte do usuário vendedor.* |
| NE 06 | *Criar uma relação de fornecedores* |
| NE 07 | *Demonstrar a disponibilidade de produtos* |

A partir desta lista de necessidades, foram formulados um conjunto de requisitos funcionais e não funcionais que serão abordados no capítulo seguinte.

# Capítulo 2

## 2.1 – Requisitos Funcionais:

A seguir encontra-se uma tabela contendo todos os requisitos funcionais levantados acerca do sistema:

| N° de requisito | Nome do requisito |
|---|---|
| RNF 01 | Registrar usuário |
| RNF 02 | Consultar usuário |
| RNF 03 | Efetuar Login |
| RNF 04 | Alterar Senha |
| RNF 05 | Registrar Produtos |
| RNF 06 | Consultar Produtos |
| RNF 07 | Alterar Produtos |
| RNF 08 | Remover Produtos |
| RNF 09 | Categorizar Produtos |
| RNF 10 | Consultar Categoria do Produto |
| RNF 11 | Consultar Preço do Produto |
| RNF 12 | Registrar Estoque |
| RNF 13 | Consultar Estoque |
| RNF 14 | Consultar Validade do Estoque |
| RNF 15 | Alterar Estoque |
| RNF 16 | Remover Produtos do Estoque |
| RNF 17 | Declarar Perda no Estoque |
| RNF 18 | Estocar Produtos |
| RNF 19 | Disponibilizar Itens do Estoque |

| | |
|---|---|
| RNF 20 | Realizar Venda |
| RNF 21 | Efetuar Compra |
| RNF 22 | Consultar Compra |
| RNF 23 | Consultar Data e Hora da Compra |
| RNF 24 | Cancelar Compra |
| RNF 25 | Alterar Status da Compra |
| RNF 26 | Registrar Fornecedor |
| RNF 27 | Consultar Fornecedor |
| RNF 28 | Alterar Fornecedor |
| RNF 29 | Remover Fornecedor |

## 2.2 Tabela de Requisitos X Necessidades

Abaixo segue uma tabela que lista as necessidades do sistema, e os requisitos que as cumprem.

| Necessidade | Requisitos que cumprem |
|---|---|
| NE1 | RF 05, RF 07, RF 08, RF 09, RF 10, RF 11, RF 12, RF 15, RF 16, RF 17, RF 18, RF 19 |
| NE2 | RF 20, RF 21, RF 22, RF 24, RF 25 |
| NE3 | RF 12, RF 13, RF 14, RF 17 |
| NE4 | RF 01, RF 02, RF 03, RF 21, RF 22, RF 24, RF 25 |
| NE5 | RF 20, RF 22, RF 24, RF 25 |
| NE6 | RF 26, RF 27, RF 28, RF 29 |
| NE7 | RF 06, RF 13, RF 19 |

## 2.3 Definição de responsabilidades dos atores do sistema

Definiremos neste subcapítulo quais serão os atores de nosso sistema, e quais são suas responsabilidades no contexto da aplicação.

Começaremos lembrando que a o sistema é desenvolvido para ser utilizado em uma lanchonete, tanto pelos clientes do espaço, quanto pela vendedora do estabelecimento.

Deste modo, já é possível definir dois atores imediatamente, o ator cliente, e o ator vendedor. Um terceiro ator pode ser levantado quando se levanta a possibilidade de que, futuramente, o estabelecimento possa ter mais de um ator vendedor, e este ator seria o ator gerente.

Agora que temos os atores delimitados, podemos definir seus papéis.

### 2.3.1 *Ator Cliente*:

O ator cliente possui o nível mais básico de acesso ao sistema, tanto de informações quanto de funcionalidades. O papel do ator cliente no sistema se limita a realizar compras. Este ator tem acesso à quase toda a informação disponível sobre produtos que estão listados como “disponíveis” no contexto do sistema, bem como as informações sobre seus próprios pedidos tanto atuais como passados no sistema.

### *2.3.2 Ator Vendedor:*

O ator vendedor possui um nível mais alto de permissões e funcionalidades no sistema. Ele é responsável por verificar os pedidos abertos, realizar vendas dentro do estabelecimento para aqueles que não utilizar a aplicação e avisar por meio do sistema quando um pedido está pronto ou foi finalizado.

O ator vendedor possui informações sobre todos os produtos que são marcados como disponíveis no contexto do sistema, bem como informações sobre vendas passadas, tanto para pessoas no estabelecimento quanto pela aplicação.

### 2.3.3 *Ator Gerente:*

O ator gerente é o ator de maior hierarquia dentro do sistema. Idealmente este será um usuário único dentro do sistema, responsável por, primariamente, gerenciar os estoques de produtos, as próprias instâncias de produtos, as informações sobre fornecedores da loja. Porém, nada impede o usuário gerente de exercer funções que seriam de um ator vendedor, uma vez que nem sempre pode haver um ator vendedor no sistema. O ator gerente possui informações sobre fornecedores, sobre diferentes estoques de produtos e sobre próprios produtos e sobre vendas passadas.

## 2.4 *Diagrama de Casos de Uso do Sistema*

Abaixo segue um link para o diagrama demonstrando o relacionamento entre os casos de uso previamente listados e seus atores:

Clique aqui para visualizar!

# Capítulo 3

## 3.1 – Requisitos Não-Funcionais

Abaixo encontram-se listados e explicados os requisitos não-funcionais do sistema, separados por diferentes tipos de restrições diferentes.

### 3.1.1 Usabilidade:

- RNF 01 - A interface gráfica para web deve ser simples, objetiva e intuitiva para máxima compreensão e velocidade de navegação dos usuários.

- RNF 02 - No quesito de “profundidade de telas”, quanto ao que se refere à quantidade de telas que se pode avançar a partir da tela inicial, o número máximo de “profundidade” deve ser de 4 telas, para garantir uma navegação mais limpa.

- RNF 03 - Links e botões serão, no geral, estilizados de maneira que se destaquem em relação ao seu ambiente gráfico.

### 3.1.2 Implementação:

- RNF 06 - O sistema deverá ser implementado em linguagem com paradigma de programação funcional, garantindo uma remoção completa, ou o máximo possível, de abstrações desnecessárias.

- RNF 07 - A linguagem escolhida para o sistema também deve ser uma linguagem que restrinja o uso tanto de funcionalidades de sua

biblioteca padrão, quanto de bibliotecas externas a usar código considerado “seguro”, no que se referem: a manipulação indireta e direta de registradores e a segurança e integridade de memória (RAM e cache) para não permitir vazamentos de memória.

- RNF 08 - O sistema deverá ser executado em qualquer sistema operacional baseado em Linux ou em Windows.

- RNF 09 - O sistema deverá ser capaz de ser visualizado, primariamente em navegadores que utilizam os motores de renderização gráficos Gecko (Firefox, Zen, GNU IceCat, etc) e Chromium (Google Chrome, Ungoogled Chromium, Arc, Opera, Brave, etc).

- RNF 10 - O banco de dados escolhido para o sistema necessita ter sido testado de antemão para problemas aleatórios e diversos que causariam problemas no próprio arquivo do banco de dados como falta de energia e corrupção de arquivos.

- RNF 11 - O sistema deverá ser implementado utilizando ferramentas que permitam a prática da “renderização por meio de servidor”, onde o servidor renderiza a página que o usuário irá acessar de antemão, e a entrega para o usuário permitindo uma navegação mais rápida uma vez que o navegador do cliente não precisará renderizar os componentes da página.

### 3.1.3 Confiabilidade:

- RNF 12 - O sistema deverá se manter operacional, idealmente até que o administrador do sistema decida que é necessário ser encerrado, para economia de energia do próprio, uma vez que seu estabelecimento não opera 24/7 ou até um momento em que o sistema operacional atinja um ponto de falha irrecuperável.

### 3.1.4 Desempenho:

- RNF 13 - O tempo máximo de resposta para o sistema é de 5 segundos inteiros sob condições normais ambientais como porcentagem de uso da memória RAM e da CPU. O tempo considerado normal de resposta deve ser entre 5 e 450 milissegundos.

- RNF 14 - A taxa de uso da CPU e de RAM da máquina por parte do sistema não deve exceder 20% e 20% respectivamente.

### 3.1.5 Segurança:

- RNF 15 - O sistema só poderá ser acessado uma vez que um usuário disponibilize suas credenciais de acesso para o sistema validá-las.

- RNF 16 - O sistema deverá ser implementado para possuir camadas ou níveis de acesso ao sistema, onde diferentes tipos de usuários têm acessos diferentes a funcionalidades do sistema.

# Capítulo 4

Este capítulo do documento irá apresentar alguns diagramas restantes que explicam melhor a separação das diferentes estruturas de dados em relação aos atores e objetos que serão trabalhados no sistema

## 4.1. Banco de Dados

### 4.1.1. Diagrama Entidade – Relacionamento:

Clique aqui para visualizar

### 4.1.2 Diagrama de Modelo de Nível Lógico:

Clique aqui para visualizar

### 4.1.3 Diagrama de Classes:

Clique aqui para visualizar

# Capítulo 5

Este capítulo do documento irá tratar das tecnologias que serão utilizadas ao longo do desenvolvimento da aplicação, seu tipo, e a razão de sua escolha

## 5.1. Linguagens

- HTML: será usada como a linguagem para estruturação de páginas web.
- CSS: será usada como a linguagem utilizada para estilizar a estrutura das páginas e animá-las.
- Javascript: será a linguagem utilizada para adicionar funcionalidade a alguns trechos das páginas web.
- Rust: A linguagem Rust será utilizada como principal linguagem de desenvolvimento do sistema. Os motivos para sua escolha variam desde segurança de memória ( o que preenche requisitos de segurança), tipagem forte, linguagem compilada, boa funcionalidade de estruturação de tipos,  código seguro ( a linguagem requer que você indique um trecho de código considerado “inseguro” para poder compilar), ótima performance, ótimo desempenho de memória se utilizarmos as funções da linguagem como ela mesmo sugere, ótima legibilidade de código, entre muitas outras questões que compõem a sua escolha como linguagem principal.

## 5.2. Bibliotecas

- Askama (Rust): A biblioteca Askama (Rust) será utilizada para criar estruturas reutilizáveis, injeção de dados, e uma população mais eficiente das páginas em HTML. A biblioteca irá operar com renderização em servidor das páginas, ao invés de renderização em cliente.

- Diesel (Rust): A biblioteca Diesel (Rust) é a responsável pelo papel de ORM (Mapeador de entidade-relacionamento) entre a linguagem Rust e o banco de dados.

- R2D2 (Rust): A biblioteca R2D2 é responsável pelo papel de criar uma fila de operações síncrona mediante a uma quantidade imprevisível de operações assíncronas e fora de ordem. Isso previne contra a possibilidade de condições de corrida e outras casualidades que acontecem devido ao SQLite não possuir funcionalidades de escrita assíncrona.

- JsonWebToken (Rust): A biblioteca JsonWebToken (Rust) é a responsável por criar e ler tokens que validam a autenticidade de um usuário.

- Colored (Rust): A biblioteca Colored é a responsável por colorir a saída padrão por meio do terminal. É essencialmente utilizada para imprimir requisições e avisos no terminal do servidor.

- RSA (Rust): A biblioteca RSA é a responsável pelas funções de criptografia de duas chaves para dados.

- Rand (Rust): A biblioteca Rand será a responsável pela criação de geradores aleatórios de dados, utilizados em diversos contextos.

- Chrono (Rust): A biblioteca Chrono (Rust) será utilizada para utilização de métricas de tempo nativamente dentro da linguagem Rust.

- Serde (Rust): A biblioteca Serde (Rust) será utilizada para serializar/desserializar estruturas de dados nativas de Rust para estruturas externas como JSON.

- HTMX (Javascript): A biblioteca HTMX (Javascript) será utilizada para aumentar a interatividade entre a relação cliente-servidor, injetando código HTML diretamente em uma página sem necessidade de recarregamento da página (AJAX).

- Bulma (CSS): A biblioteca Bulma (CSS) será utilizada para maior velocidade de produção de páginas HTML, pois dispensará a necessidade de criar cada vez mais classes customizadas para diferentes elementos. A biblioteca usa estilos responsivos por padrão.

## 5.3. Frameworks

- Ntex (Rust): O framework Ntex (Rust) será utilizado como framework back-end para desfrutar as funcionalidades de um servidor web, como rotas, recebimento de formulários e gerenciamento de sessões.

- SQLite (Banco de dados): SQLite é a tecnologia responsável pelo banco de dados do nosso sistema. Essa tecnologia foi escolhida por sua simplicidade de funcionamento e forte garantia de integridade dos dados, uma vez que é constantemente testada contra falhas.

# Capítulo 6

## 6.1. Considerações finais

- Todos os diagramas e imagens que deveriam aparecer neste documento estão em formatos de links para que o leitor possa acessar as imagens na sua mais alta qualidade.

## 6.2. Referências


- Rust Lang: https://www.rust-lang.org/pt-BR, https://doc.rust-lang.org/book/, https://doc.rust-lang.org/rust-by-example/
- Askama: https://github.com/djc/askama, https://djc.github.io/askama/
- Diesel: https://diesel.rs/, https://github.com/diesel-rs/diesel, https://docs.rs/diesel/latest/diesel/
- R2D2: https://github.com/sfackler/r2d2, https://docs.rs/diesel/latest/diesel/r2d2/index.html
- JsonWebToken: https://docs.rs/jsonwebtoken/latest/jsonwebtoken/, https://github.com/Keats/jsonwebtoken
- Colored: https://docs.rs/colored/latest/colored/, https://github.com/colored-rs/colored
- RSA: https://github.com/RustCrypto/RSA, https://docs.rs/rsa/latest/rsa/
- Rand: https://github.com/rust-random/rand, https://docs.rs/rand/latest/rand/

- Chrono: https://github.com/chronotope/chrono/releases, https://docs.rs/chrono/latest/chrono/

- Serde: https://github.com/serde-rs/serde, https://docs.rs/serde/latest/serde/index.html, https://serde.rs/

- HTMX: https://htmx.org/

- Bulma CSS: https://bulma.io/

- Ntex: https://ntex.rs/, https://www.techempower.com/benchmarks/#hw=ph&test=composite&section=data-r22, https://github.com/ntex-rs/ntex, https://docs.rs/ntex/latest/ntex/

- SQLite: https://www.sqlite.org/