

Documentação Nexus Solutions

Autor: Alvaro Daniel Almeida Motta

Sumário

1 Introdução	3
1.1 Nome do Projeto	3
1.2 Objetivo	3
1.3 Escopo	3
1.4 Público-Alvo	4
1.5 Contexto	5
1.6 Visão Geral	5
2 Requisitos	7
2.1 Requisitos Funcionais	7
2.2 Requisitos Não Funcionais	9
3 Arquitetura do Sistema	10
3.1 Visão Geral	10
3.2 Tecnologias e Frameworks utilizados	10
3.3 Padrões de Projetos Adotados	10
3.4 Estrutura de Pacotes do Projeto	11
4 Modelagem	13
4.1 Diagrama Lógico	13
4.2 Diagrama Físico	14
4.3 Dicionário de Dados	15
5. Instalação e Configuração	21
5.1 Requisitos de Ambiente	21
5.2 Como Rodar o Projeto Localmente	21
5.2.1 Backend	21
5.2.2 Frontend	21
5.2.3 Mobile	21
5.3 Variáveis de Ambiente	22
5.3.1 Back-End	22
5.3.2 Front-End (Web)	23
5.3.3 Mobile (React Native)	23
6 Guia de Uso	24
6.1 Visão Geral do Funcionamento	24
6.2 Perguntas do Oráculo	25
6.3 Limitações do Oráculo	26
7 Manutenção e Contribuição	27
7.1 Organização do Código	27
7.2 Convenções de Código	28
7.3 Contribuição	28
8 Melhorias	29
8.1 Testes Automatizados	29
8.2 Expansão das Funcionalidades do Oráculo	29
9 Licença	30

1 Introdução

1.1 Nome do Projeto

Nexus Solutions

1.2 Objetivo

O sistema tem como principal objetivo otimizar e automatizar o controle de almoxarifado em empresas de pequeno e médio porte, oferecendo ferramentas intuitivas para o gerenciamento de produtos, estoques e movimentações. Através de uma solução integrada nas versões web e mobile, o sistema proporciona maior agilidade nos processos logísticos e maior precisão nas informações.

1.3 Escopo

O sistema contempla funcionalidades essenciais para a gestão eficiente do almoxarifado, incluindo:

- Cadastro, edição, consulta e exclusão de produtos;
- Registro e controle de movimentações de entrada e saída;
- Gerenciamento de estoques em tempo real;
- Leitura de produtos via código de barras e QR Code (gerado automaticamente no cadastro);
- Acesso a funcionalidades práticas por meio do aplicativo mobile;
- Cadastro, gerenciamento e exclusão de funcionários do sistema, com definição de permissões por tipo de perfil (MANAGER e OPERATOR).
- Integração com o Oráculo, uma assistente virtual com inteligência artificial capaz de responder a perguntas sobre o sistema, auxiliar na execução de tarefas e oferecer suporte inteligente aos usuários de forma natural e contextualizada.

O escopo não abrange, nesta versão inicial, integrações com sistemas externos (como ERPs), nem funcionalidades fiscais, contábeis ou financeiras.

1.4 Público-Alvo

O sistema é voltado para empresas que necessitam de maior controle e organização em seus almoxarifados, independentemente do porte ou segmento de atuação. Atende especialmente organizações que realizam movimentações frequentes de materiais, tais como:

- Indústrias;
- Construtoras;
- Prestadoras de serviços;
- Instituições públicas e privadas que mantêm estoques internos de insumos, equipamentos ou peças.

Além das organizações, o sistema é direcionado a profissionais que atuam diretamente na gestão e operação do almoxarifado, incluindo:

- Gestores;
- Almoxarifes;
- Colaboradores responsáveis pelo controle de estoque, compras e logística interna.

Esses usuários buscam uma solução intuitiva, eficiente e segura, capaz de otimizar suas rotinas operacionais e reduzir perdas decorrentes de falhas no controle manual.

A versão mobile do sistema foi desenvolvida para atender às necessidades dos profissionais que trabalham no ambiente físico do almoxarifado, oferecendo mobilidade e agilidade nas operações.

1.5 Contexto

A empresa **Meliy** enfrenta desafios significativos em sua rotina de gestão de almoxarifado, decorrentes da adoção de métodos manuais para o controle de estoque, como anotações em cadernos e registros informais. Essa abordagem tem se mostrado ineficaz, resultando em inconsistências nas contagens, atrasos na localização de itens, retrabalho da equipe e, principalmente, em decisões de compra emergenciais a preços elevados, devido à falta de visibilidade prévia sobre a reposição de produtos.

Diante desse cenário, identificou-se a necessidade de implementar uma solução digital que automatize e centralize os processos relacionados ao almoxarifado. A motivação para o desenvolvimento deste software está diretamente ligada à busca por maior eficiência operacional, organização dos dados e economia de recursos, além da redução de erros humanos nas atividades de controle de estoque.

A proposta é viabilizar uma ferramenta moderna, acessível e com recursos inovadores, como a leitura de QR Code e um módulo inteligente de consultas baseado em Inteligência Artificial (Oráculo), a fim de transformar a forma como as operações de almoxarifado são conduzidas. Com isso, espera-se que a empresa Meliy e demais organizações que vierem a utilizar o sistema possam alcançar um novo patamar de controle, segurança e agilidade em suas atividades logísticas.

1.6 Visão Geral

O sistema de gestão de almoxarifados desenvolvido para a empresa Meliy é uma solução integrada que visa modernizar e automatizar os processos de controle de estoque. Ele engloba módulos para cadastro e gerenciamento de produtos, almoxarifados e funcionários, além de controle detalhado das movimentações de materiais.

A solução contempla uma interface web para gestão administrativa e um aplicativo mobile para uso direto no almoxarifado, incluindo funcionalidades como leitura de QR Code para agilizar operações. Um módulo inteligente baseado em inteligência

artificial — o Oráculo — oferece consultas rápidas e precisas para auxiliar na tomada de decisões e no controle do estoque.

Com essa solução, a empresa poderá reduzir erros manuais, melhorar a acuracidade do inventário e otimizar o fluxo logístico, garantindo maior eficiência operacional e economia de recursos.

2 Requisitos

2.1 Requisitos Funcionais

Cadastro de Produtos

- O sistema deve permitir o cadastro de produtos com informações básicas, como nome, descrição e imagem.
- Deve gerar automaticamente QR Code para cada produto cadastrado para facilitar a identificação e rastreamento.
- Cada produto pode estar vinculado a um ou mais almoxarifados, com controle individualizado do estoque em cada um deles.

Gerenciamento de Almoxarifados e Estoques

- Deve ser possível cadastrar múltiplos almoxarifados, cada um com seu estoque próprio.
- O sistema deve permitir o controle do estoque de cada produto em cada almoxarifado, registrando a quantidade disponível individualmente.

Gerenciamento de Funcionários

- Cadastro de funcionários com informações como nome, email, senha e imagem, e definição de perfis de acesso.
- Perfis básicos:
 - **MANAGER**: permissões administrativas completas, incluindo controle de almoxarifados, produtos, funcionários e relatórios.
 - **OPERATOR**: permissões operacionais restritas, como realizar movimentações e consultas.

Controle de Movimentação de Estoque

- Registro detalhado de todas as movimentações de entrada e saída de produtos em almoxarifados específicos.

- Atualização automática da quantidade disponível em estoque no almoxarifado relacionado após cada movimentação.
- Histórico completo de movimentações para auditoria.

Consulta Inteligente via Oráculo

- Disponibilizar módulo inteligente de consultas, baseado em IA, capaz de responder dúvidas sobre produtos, estoque por almoxarifado, movimentações e outros dados relevantes.
- Permitir interação em linguagem natural, com respostas claras e objetivas.

Interface Mobile

- Interface mobile otimizada para profissionais que atuam diretamente nos almoxarifados.
- Permitir leitura de QR Codes para identificação rápida dos produtos durante as operações.

Relatórios e Alertas

- Envio de alertas para usuários MANAGER quando estoques atingirem níveis críticos em almoxarifados específicos.

Segurança e Controle de Acesso

- Sistema com autenticação obrigatória para acesso.
- Controle de acesso baseado no perfil do usuário, restringindo funcionalidades conforme permissão.

2.2 Requisitos Não Funcionais

- **Desempenho:**

O sistema deve ser capaz de processar operações de cadastro, listagem e movimentação em tempo real, com resposta inferior a 2 segundos em conexões estáveis.

- **Segurança:**

O sistema deve contar com autenticação de usuários, controle de permissões por perfil e criptografia de dados sensíveis. Backups devem ser realizados automaticamente em intervalos regulares.

- **Usabilidade:**

A interface deve ser simples, intuitiva e acessível, permitindo que usuários com conhecimento técnico básico consigam operar o sistema com facilidade.

- **Compatibilidade e Responsividade:**

A versão web deve ser compatível com os principais navegadores modernos. A versão mobile deve funcionar corretamente em dispositivos Android, com telas adaptadas para diferentes tamanhos.

- **Escalabilidade:**

O sistema deve permitir o cadastro e gerenciamento de múltiplas empresas sem perda de desempenho, possibilitando o crescimento da base de usuários e dados.

- **Portabilidade:**

O sistema deve ser facilmente instalável em diferentes servidores com ambiente compatível.

3 Arquitetura do Sistema

3.1 Visão Geral

O sistema será desenvolvido com uma **arquitetura monolítica** organizada em camadas, seguindo o padrão **MVC (Model-View-Controller)** para garantir a separação clara entre as responsabilidades das camadas de apresentação, negócio e dados. Essa estrutura facilita a manutenção, o entendimento e a escalabilidade futura do sistema.

O projeto será desenvolvido com foco na **qualidade do código**, buscando aderir aos princípios do **SOLID** e às boas práticas de **Clean Code**.

3.2 Tecnologias e Frameworks utilizados

Linguagem de Programação: Java (backend)

Framework Backend: Spring Framework (Spring Boot para criação das APIs REST)

Banco de Dados: PostgreSQL

Frontend Web: React.js

Mobile: React Native

Armazenamento de Imagens: Integração com **AWS S3** para armazenamento e recuperação de imagens de produtos

Módulo Inteligente de Consultas (Oráculo): Integração com a **Gemini API** para processamento e respostas baseadas em inteligência artificial

3.3 Padrões de Projetos Adotados

MVC (Model-View-Controller) para organização das camadas do sistema

SOLID para garantir código modular, flexível e de fácil manutenção

Clean Code para melhorar a legibilidade e qualidade do código-fonte

3.4 Estrutura de Pacotes do Projeto

A organização do código-fonte no backend do projeto segue uma estrutura lógica e modular, que facilita o desenvolvimento, a manutenção e a escalabilidade do sistema. A base do projeto está no diretório `src/main/java/com/nexus`, onde os pacotes estão divididos conforme suas responsabilidades:

controller: contém as classes responsáveis por receber as requisições HTTP, fazer a validação inicial e direcionar as chamadas para a camada de serviço.

service: concentra a lógica de negócio da aplicação, incluindo regras de processamento e orquestração das operações.

repository: agrupa as interfaces para acesso e manipulação dos dados no banco. Dentro deste pacote, existe o subpacote **specification**, que abriga consultas personalizadas para o banco de dados.

model: define as entidades do domínio. Contém também um subpacote **enums**, onde ficam as enumerações utilizadas pelas entidades.

dto: agrupa os Data Transfer Objects (DTOs), utilizados para transportar dados entre as camadas e para comunicação com o frontend. Para organização, cada entidade possui seu próprio subpacote, como **Address** e **Product**. Além disso, existem classes gerais para respostas como **SuccessResponse**, **ErrorResponse** e **ImageResponse** diretamente no pacote `dto`.

exception: concentra as classes relacionadas ao tratamento de erros e exceções customizadas da aplicação.

infra: reúne configurações e implementações relacionadas à infraestrutura do sistema, tais como integrações externas e configurações diversas. Possui subpacotes como **scheduling** (para tarefas agendadas) e **security** (para configurações de segurança), além de classes gerais no pacote raiz, como **WebConfig** e **S3StorageAdapter**.

oracle: pacote dedicado ao módulo inteligente de consultas (Oráculo) que utiliza inteligência artificial. Inclui os subpacotes **command** (para os comandos processados pelo Oráculo) e **dispatcher** (responsável pelo roteamento das consultas).

ports: contém as interfaces que definem contratos para comunicação entre módulos internos ou externos, promovendo baixo acoplamento.

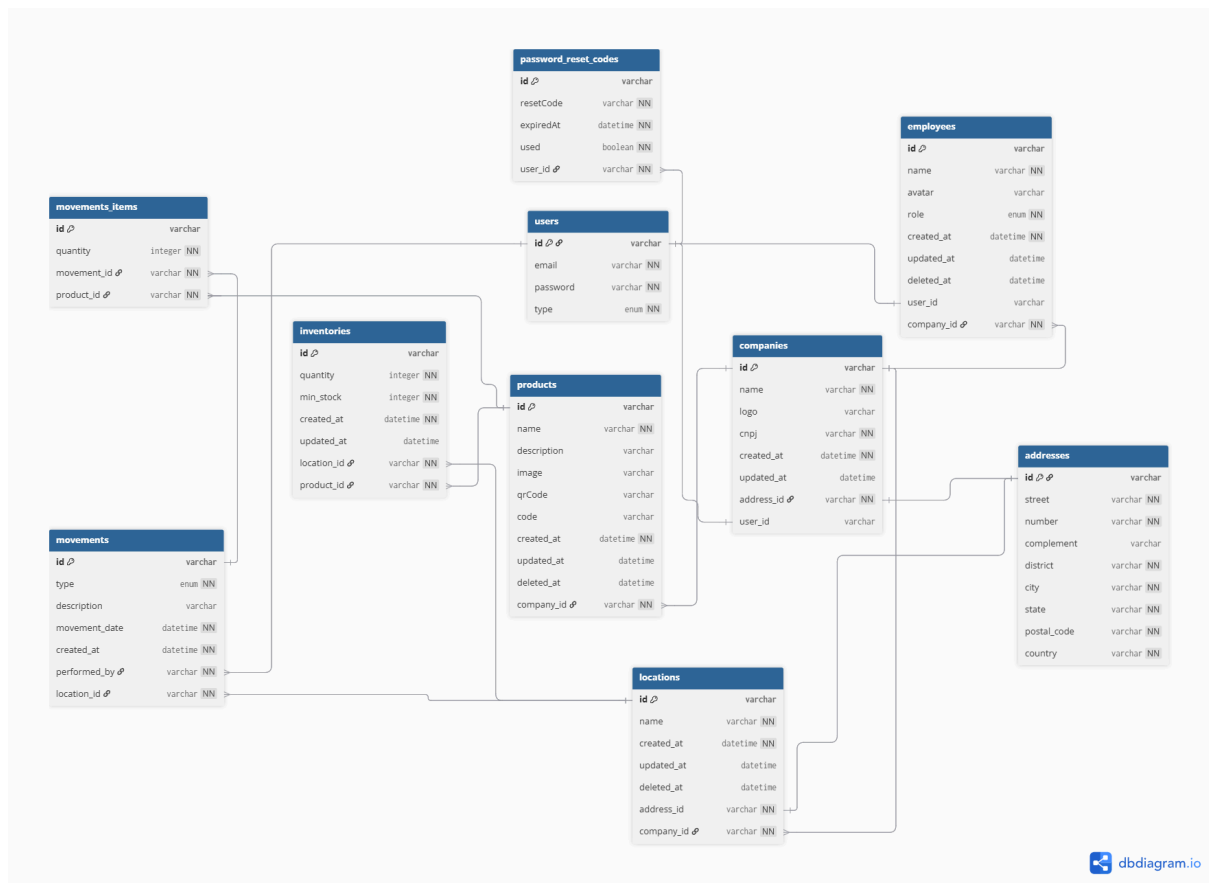
utils: abriga classes utilitárias e funções auxiliares que são usadas em diferentes partes da aplicação.

core.springdoc e **openapi:** pacotes dedicados à configuração e arquivos relacionados à documentação da API, utilizando Swagger e OpenAPI.

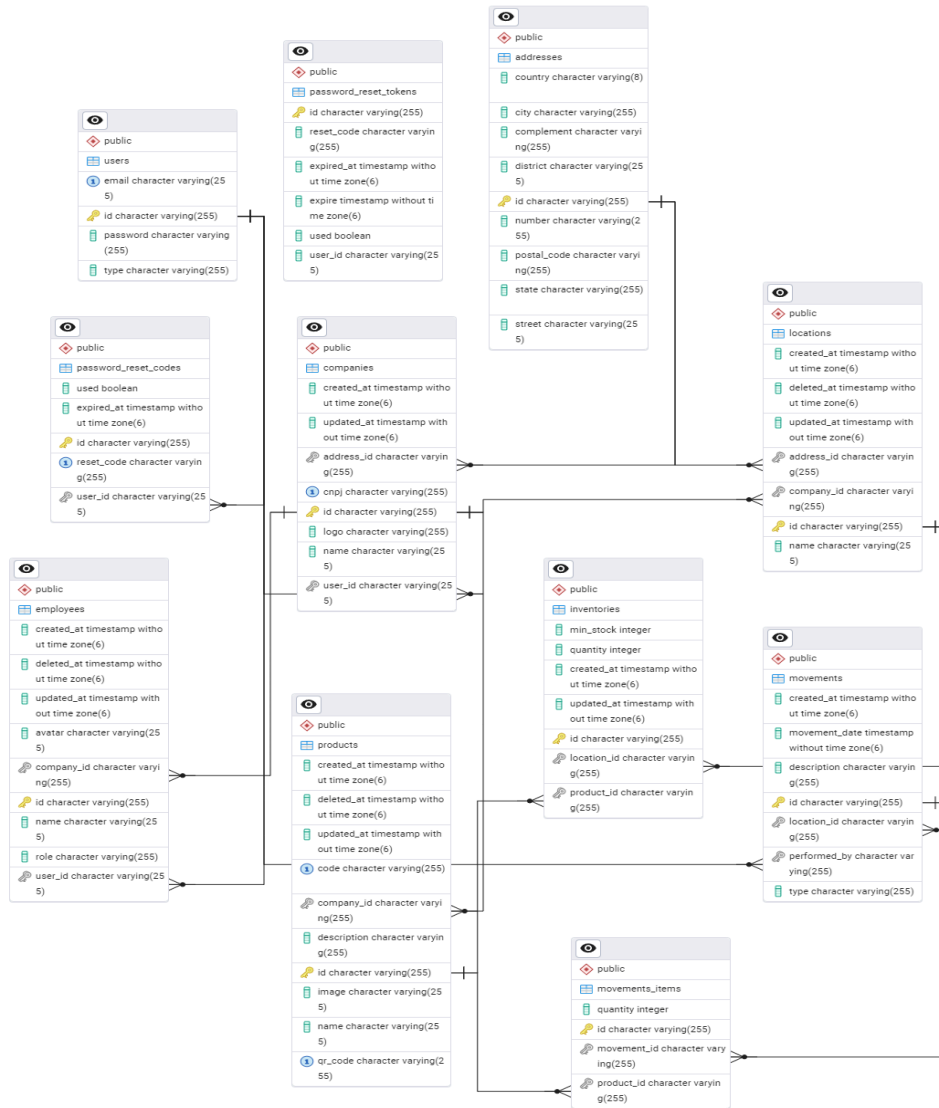
Essa estrutura modular e bem definida contribui para uma arquitetura organizada, facilitando o desenvolvimento em equipe e a manutenção futura do sistema.

4 Modelagem

4.1 Diagrama Lógico



4.2 Diagrama Físico



4.3 Dicionário de Dados

4.3.1 Users

Tabela para autenticação e controle de acesso ao sistema, contendo dados como nome de usuário, e-mail, senha e tipo (employee ou company).

Atributo	Descrição	Tipo	Restrições
id	Identificador único do usuário	varchar	pk
email	Email do usuário	varchar	not null, unique
password	Senha do usuário	varchar	not null
type	Tipo de usuário (EMPLOYEE ou COMPANY)	enum	not null

4.3.2 Companies

Representa as empresas cadastradas no sistema, contendo dados como nome, CNPJ, logo e endereço associado.

Atributo	Descrição	Tipo	Restrições
id	Identificador da empresa	varchar	pk
name	Nome da empresa	varchar	not null
logo	URL do logo	varchar	-
cnpj	CNPJ da empresa	varchar	not null, unique
created_at	Data de criação	datetime	not null
updated_at	Última atualização	datetime	-
address_id	Endereço da empresa	varchar	fk, not null
user_id	Usuário associado	varchar	fk

4.3.3 Employees

Contém os dados dos funcionários, como nome, cargo, e-mail, senha e vínculo com uma empresa específica.

Atributo	Descrição	Tipo	Restrições
id	Identificador do funcionário	varchar	pk
name	Nome do funcionário	varchar	not null
avatar	URL da imagem de perfil	varchar	-
role	Cargo do funcionário (OPERATOR ou MANAGER)	enum	not null
created_at	Data de criação	datetime	not null
updated_at	Última atualização	datetime	-
deleted_at	Exclusão lógica	datetime	-
user_id	Usuário associado	varchar	fk
company_id	Empresa vinculada	varchar	fk, not null

4.3.4 Products

Contém os dados dos produtos cadastrados no sistema, como nome, descrição, código de barras, imagem, etc.

Atributo	Descrição	Tipo	Restrições
id	Identificador do produto	varchar	pk
name	Nome do produto	varchar	not null
description	Descrição	varchar	-
image	URL da imagem	varchar	-
qrCode	Código QR do produto	varchar	unique
code	Código de barras	varchar	unique
created_at	Data de criação	datetime	not null
updated_at	Última atualização	datetime	-
deleted_at	Exclusão lógica	datetime	-
company_id	Empresa proprietária	varchar	fk, not null

4.3.5 Locations

Representa os almoxarifados (ou depósitos) de cada empresa, armazenando informações como nome, endereço e empresa responsável.

Atributo	Descrição	Tipo	Restrições
id	Identificador do almoxarifado	varchar	pk
name	Nome do almoxarifado	varchar	not null
created_at	Data de criação	datetime	not null
updated_at	Última atualização	datetime	-
deleted_at	Exclusão lógica	datetime	-
address_id	Endereço vinculado	varchar	fk, not null
company_id	Empresa proprietária	varchar	fk, not null

4.3.6 Inventories

Reflete os estoques dos almoxarifados, contendo registros dos produtos disponíveis em uma determinada localização e suas quantidades.

Atributo	Descrição	Tipo	Restrições
id	Identificador do estoque	varchar	pk
quantity	Quantidade em estoque	integer	not null, default: 0
min_stock	Estoque mínimo	integer	not null, default: 1
created_at	Data de criação	datetime	not null
updated_at	Última atualização	datetime	-
location_id	Local de armazenamento	varchar	fk, not null, unique
product_id	Produto vinculado	varchar	fk, not null

4.3.7 Movements

Registra as movimentações de entrada e saída de itens no estoque, o tipo de operação (IN/OUT), e a data.

Atributo	Descrição	Tipo	Restrições
id	Identificador da movimentação	varchar	pk
type	Tipo da movimentação (IN / OUT)	enum	not null
description	Descrição	varchar	-
movement_date	Data da movimentação	datetime	not null
created_at	Data de criação	datetime	not null
performed_by	Usuário responsável	varchar	fk, not null
location_id	Local de armazenamento afetado	varchar	fk, not null

4.3.8 Movements Items

Detalha os produtos movimentados em cada operação de entrada ou saída, incluindo o produto, quantidade e referência à movimentação principal.

Atributo	Descrição	Tipo	Restrições
id	Identificador do item	varchar	pk
quantity	Quantidade movimentada	integer	not null
movement_id	Movimentação relacionada	varchar	fk, not null
product_id	Produto movimentado	varchar	fk, not null

4.3.9 Addresses

Detalha os produtos movimentados em cada operação de entrada ou saída, incluindo o produto, quantidade e referência à movimentação principal.

Atributo	Descrição	Tipo	Restrições
id	Identificador do endereço	varchar	pk
street	Rua	varchar	not null
number	Número	varchar	not null
complement	Complemento	varchar	-
district	Bairro	varchar	not null
city	Cidade	varchar	not null
state	Estado	varchar	not null
postal_code	CEP	varchar	not null
country	País	varchar	not null

4.3.10 Password Reset Codes

Detalha os produtos movimentados em cada operação de entrada ou saída, incluindo o produto, quantidade e referência à movimentação principal.

Atributo	Descrição	Tipo	Restrições
id	Identificador do código	varchar	pk
resetCode	Código de redifinição de senha	varchar	not null, unique
expiredAt	Expiração do código	datetime	not null
used	Código usado	boolean	not null
user_id	Usuário associado	varchar	fk, not null

5. Instalação e Configuração

5.1 Requisitos de Ambiente

Java 23, Maven – Backend

Node.js – Frontend e Mobile

Docker – Banco de dados local (PostgreSQL)

Android Studio / Emulador – Para rodar o mobile localmente

5.2 Como Rodar o Projeto Localmente

5.2.1 Backend

Subir o banco de dados no docker

```
docker-compose up -d
```

Rodar o projeto

```
mvn clean install
```

```
mvn spring-boot:run
```

5.2.2 Frontend

Instalar as dependências

```
npm install
```

Rodar o projeto

```
npm run dev
```

5.2.3 Mobile

Instalar as dependências

```
npm install
```

Rodar o projeto

```
npm run android
```

5.3 Variáveis de Ambiente

Para garantir o correto funcionamento do sistema, tanto em ambiente local quanto em produção, é necessário configurar variáveis de ambiente nas três camadas do projeto: **Back-End**, **Front-End (Web)** e **Mobile**. Essas variáveis definem valores sensíveis ou específicos do ambiente de execução, como credenciais de banco de dados, URLs de serviços, chaves de API e configurações de e-mail.

5.3.1 Back-End

As variáveis de ambiente utilizadas no back-end estão referenciadas no arquivo `application.properties`. Essas variáveis são responsáveis por armazenar informações sensíveis e parâmetros de configuração, como:

Credenciais de banco de dados:

- `DBHOST`, `DBPORT`, `DBDATABASE`, `DBUSER`, `DBPASSWORD`

Segurança:

- `JWT_SECRET`: chave usada para geração e verificação de tokens JWT

Integração com o Front-End:

- `NEXUS_FRONTEND_URL`: URL do front-end para configuração de CORS

Serviços AWS (S3):

- `AWS_REGION`, `AWS_BUCKET_NAME`: utilizados para armazenar imagens no Amazon S3
- `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`: chaves de acesso para autenticação com a AWS

Envio de e-mails:

- `MSHOST`, `MSPORT`, `MSUSERNAME`, `MSPASSWORD`, `MAIL_FROM`: dados para configurar o provedor SMTP

Integração com a API da Oracle (IA):

- `API_KEY`, `PROMPT`

O prompt você pode encontrar aqui:

https://www.notion.so/Nexus-Solutions-Documenta-o-23812b6797b3805e93a5d9b59cf67bab?source=copy_link

5.3.2 Front-End (Web)

No projeto front-end desenvolvido com **React**, é utilizado um arquivo `.env` na raiz do projeto, contendo a variável:


`VITE_API_URL`

Essa variável indica a URL base da API que será consumida pelo front-end. Para funcionar corretamente, é necessário instalar as dependências com `npm install` e iniciar o projeto com `npm run dev`

5.3.3 Mobile (React Native)

De forma semelhante ao front-end, o projeto mobile utiliza um arquivo `.env` com a seguinte variável:

`API_URL`

 Importante: no emulador Android, o IP `10.0.2.2` representa o **localhost da máquina host** (para acessar o back-end rodando localmente).

Após definir a variável corretamente, o projeto pode ser executado com:

`npm install`

`npm run android`

6 Guia de Uso

6.1 Visão Geral do Funcionamento

O sistema foi desenvolvido para gerenciar o fluxo de produtos dentro de um ambiente de almoxarifado, permitindo o cadastro, movimentação e rastreo eficiente de itens, usuários, locais e empresas envolvidas. Além disso, um módulo inteligente chamado **Oráculo** oferece assistência automática com base em perguntas feitas pelos usuários.

Funcionalidades Principais:

- **Autenticação e Autorização**

Usuários podem se registrar e fazer login. A autenticação utiliza JWT, e o sistema controla os níveis de acesso conforme o papel do usuário (Ex: Gerente, Operador).

- **Cadastro de Entidades**

O sistema permite o gerenciamento completo de entidades como:

- Empresas
- Funcionários
- Produtos
- Endereços
- Almoxarifados (Locations)

- **Movimentação de Itens**

Os usuários podem registrar movimentações de entrada e saída de itens entre locais, com controle individualizado por produto. Cada movimentação é composta por múltiplos itens e armazena dados como data, local de origem/destino e responsável.

- **Oráculo**

Um módulo de IA que responde a perguntas pré-cadastradas, auxiliando usuários com dúvidas frequentes.

- **Responsividade e Multiplataforma**

A aplicação é acessível via Web e Mobile, oferecendo uma experiência adaptada a diferentes dispositivos.

- **Armazenamento e Configuração**

A aplicação utiliza banco de dados PostgreSQL e pode ser executada tanto

em ambiente local quanto em produção, com configuração por variáveis de ambiente.

6.2 Perguntas do Oráculo

O Oráculo é um assistente inteligente integrado ao sistema, desenvolvido para responder automaticamente a perguntas comuns relacionadas ao gerenciamento de estoque, movimentações, produtos, funcionários e indicadores. As perguntas são interpretadas com base em regras pré-definidas, e a resposta é retornada em formato JSON padronizado, seguindo uma estrutura que permite integração direta com o sistema.

Exemplos de perguntas compreendidas pelo Oráculo:

Produtos

- "Quais produtos estão com estoque baixo?"
- "Me mostre todos os produtos cadastrados."
- "Qual a quantidade do produto [X]?"
- "Qual a quantidade de produtos em estoque?"

Localizações / Almoxarifados

- "Quais são os almoxarifados cadastrados?"
- "Onde está armazenado o produto [X]?"

Movimentações

- "Quais foram as últimas movimentações?"
- "Quais foram as últimas movimentações de entrada?"

Funcionários

- "Quantos funcionários temos cadastrados?"
- "Quem são os operadores?"

Mensagens Gerais

- "Olá"
- "Quem é você?"
- "Tudo bem?"

Essas interações resultam em respostas estruturadas com status HTTP, mensagens amigáveis e comandos que podem acionar rotinas internas do sistema, como consultas, filtros e geração de relatórios.

6.3 Limitações do Oráculo

Apesar de o Oráculo ser uma ferramenta eficiente para consulta de informações relacionadas ao sistema, seu funcionamento possui algumas limitações que devem ser consideradas durante o uso:

- **Interpretação Literal:**
O Oráculo entende apenas perguntas estruturadas com base nos dados existentes no banco. Perguntas muito abertas, vagas ou ambíguas podem não retornar resultados relevantes.
- **Dependência da Base de Dados:**
O Oráculo responde somente com base nos dados armazenados no sistema. Se uma informação não estiver cadastrada, ele não conseguirá gerar uma resposta adequada.
- **Contexto Limitado:**
O Oráculo não guarda o histórico de interações. Cada pergunta deve ser feita de forma completa e independente da anterior.
- **Respostas Não Adivinhadas:**
O sistema não “chuta” respostas. Se a informação não estiver completa ou corretamente registrada, o Oráculo não fornecerá retorno.
- **Não Interativo:**
O Oráculo responde com dados diretos. Ele não realiza ações no sistema, como atualizar registros, cadastrar informações ou tomar decisões.

Essas limitações foram definidas para manter a robustez, segurança e previsibilidade do comportamento do Oráculo. Melhorias futuras poderão ampliar suas capacidades.

7 Manutenção e Contribuição

7.1 Organização do Código

O projeto adota uma estrutura de pacotes bem definida para promover clareza, separação de responsabilidades e facilitar a manutenção. Abaixo estão os principais pacotes:

controller: define os pontos de entrada da API REST.

dto: objetos de transferência de dados entre camadas.

service: lógica de negócio.

repository: interfaces JPA que acessam o banco de dados.

model: entidades persistidas no banco.

exception: exceções personalizadas e tratadores globais.

infra: implementações auxiliares como autenticação, conexões externas, etc.

ports: interfaces responsáveis pela comunicação com serviços.

utils: classes utilitárias e auxiliares.

oracle: camada voltada para comandos do oráculo.

core.springdoc e **openapi**: usados para configuração e documentação via Swagger/OpenAPI.

7.2 Convenções de Código

Nome de classes no formato PascalCase (ex: `InventoryController`).

Nome de variáveis e métodos em camelCase (ex: `getInventoryById`).

Comentários claros e objetivos quando necessário.

Boas práticas com tratamento de exceções e uso de `Optional` onde apropriado.

Utilização de `@Transactional` nos métodos de escrita.

Controle de versão via Git, com mensagens de commit descritivas.

7.3 Contribuição

Desenvolvedores que desejarem contribuir devem:

1. Realizar um fork do repositório.
2. Criar uma branch com um nome descritivo (ex: `feature/nova-funcao`).
3. Seguir o padrão de organização e nomeação do projeto.
4. Enviar um Pull Request com uma descrição clara das mudanças realizadas.

8 Melhorias

Esta seção apresenta possíveis aprimoramentos planejados para versões futuras do sistema, visando aumentar a robustez, escalabilidade e utilidade da aplicação.

8.1 Testes Automatizados

Pretende-se implementar **testes unitários e de integração** utilizando ferramentas como **JUnit** e **Mockito**, garantindo a confiabilidade do sistema em atualizações futuras e facilitando a manutenção do código.

8.2 Expansão das Funcionalidades do Oráculo

Atualmente, o Oráculo responde a um conjunto pré-definido de perguntas. No entanto, há planos para torná-lo mais **inteligente e contextual**, com capacidade de:

- Reconhecer variações de perguntas.
- Aprender com interações passadas (via histórico).
- Consultar dados em tempo real da base.

9 Licença

Este projeto é distribuído sob a **Licença MIT**.

A Licença MIT é uma licença de software livre permissiva, o que significa que ela concede aos usuários a liberdade de usar, modificar, distribuir e até mesmo vender o software, seja em sua forma original ou modificada. A única exigência é que a nota de copyright e a própria licença MIT sejam incluídas em todas as cópias ou partes substanciais do software.

Principais características da Licença MIT:

- **Permissão de Uso:** Você pode usar o software para qualquer finalidade, incluindo projetos comerciais.
- **Permissão de Modificação:** Você pode modificar o código-fonte como desejar.
- **Permissão de Distribuição:** Você pode distribuir o software e suas modificações.
- **Permissão de Sublicenciamento:** Você pode conceder sublicenças do software.
- **Isenção de Responsabilidade:** O software é fornecido "como está", sem garantia de qualquer tipo, expressa ou implícita. Os autores e detentores dos direitos autorais não são responsáveis por quaisquer danos ou outras responsabilidades decorrentes do uso do software.

A escolha da Licença MIT visa promover a colaboração e a inovação, permitindo que a comunidade se aproprie e melhore o projeto, com o mínimo de restrições.