Manual de Instruções do Sistema de Almoxarifado

Autor: Lourenço Levinski Fernandes

1. Introdução

1.1. Visão Geral do Sistema

1.2. Público-Alvo

1.3. Requisitos do Sistema

2. Pré-requisitos e Instalação

2.1. Instalação do Python

2.2. Instalação do VSCode

2.3. Instalação das Dependências Python

2.4. Configuração do Banco de Dados MySQL

2.4.1. Instalação do MySQL Server

2.4.2. Criação do Banco de Dados e Usuário

2.4.3. Estrutura do Banco de Dados (SQL)

2.4.4. Importação de Dados Iniciais (Opcional)

2.5. Configuração do Arquivo `default.png`

3. Configuração e Execução do Código

3.1. Clonando o Repositório (Opcional, para GitHub)

1. Introdução

Este manual tem como objetivo fornecer instruções detalhadas e passo a passo para a instalação, configuração e execução do Sistema de Almoxarifado desenvolvido em Python, utilizando a biblioteca PyQt6 para a interface gráfica e MySQL para o gerenciamento do banco de dados. O sistema foi projetado para auxiliar na gestão de empréstimos e devoluções de componentes em um ambiente educacional.

1.1. Visão Geral do Sistema

O Sistema de Almoxarifado é uma aplicação desktop que permite o controle de itens (componentes) emprestados a alunos, professores e funcionários. Suas principais funcionalidades incluem:

- **Pesquisa de Alunos:** Permite buscar alunos por matrícula ou nome, exibindo seus detalhes e histórico de empréstimos.
- **Pesquisa de Componentes:** Possibilita a busca de componentes por ID ou nome, facilitando a localização de itens para empréstimo.
- **Registro de Empréstimos:** Funcionalidade para registrar a saída de componentes, associando-os a um aluno específico e registrando a data do empréstimo.
- **Registro de Devoluções:** Permite marcar componentes como devolvidos, atualizando o status do empréstimo no sistema.

O sistema interage com um banco de dados MySQL para armazenar informações sobre alunos, componentes e registros de empréstimos, garantindo a persistência e integridade dos dados.

1.2. Público-Alvo

Este manual é destinado a usuários técnicos, incluindo administradores de sistemas, desenvolvedores e pessoal de suporte técnico, que serão responsáveis pela instalação, configuração e manutenção do Sistema de Almoxarifado. Além disso, os usuários finais, como alunos, professores e funcionários do almoxarifado da Fundação Escola Técnica Liberato Salzano Vieira da Cunha, encontrarão informações sobre o uso diário do sistema.

1.3. Requisitos do Sistema

Para a correta execução do Sistema de Almoxarifado, os seguintes requisitos de hardware e software devem ser atendidos:

Hardware:

- Processador: Intel Core i3 ou equivalente (mínimo)
- Memória RAM: 4 GB (mínimo), 8 GB (recomendado)

- Espaço em Disco: 500 MB de espaço livre para o software e banco de dados

Software:

- **Sistema Operacional:** Windows 10/11, macOS (versões recentes) ou distribuições Linux (Ubuntu, Fedora, etc.)
- **Python:** Versão 3.x (recomendado 3.8 ou superior)
- **VSCode:** Visual Studio Code (última versão estável)
- **MySQL Server:** Versão 8.0 ou superior
- **Dependências Python:**
  - `mysql-connector-python`
  - `PyQt6`
  - `pytz`

Certifique-se de que todos os softwares listados estejam instalados e configurados corretamente antes de prosseguir com a instalação do sistema.

2. Pré-requisitos e Instalação

Esta seção detalha os passos necessários para preparar o ambiente de desenvolvimento e execução do Sistema de Almoxarifado, incluindo a instalação de linguagens de programação, ferramentas e bancos de dados.

2.1. Instalação do Python

O Python é a linguagem de programação na qual o Sistema de Almoxarifado foi desenvolvido. Siga os passos abaixo para instalar o Python em seu sistema operacional:

No Windows:

1. Acesse o site oficial do Python: https://www.python.org/downloads/
2. Baixe o instalador da versão mais recente do Python 3.x (recomenda-se a versão estável mais recente).
3. Execute o instalador. **Importante:** Certifique-se de marcar a opção "Add Python X.X to PATH" (Adicionar Python X.X ao PATH) durante a instalação. Isso permitirá

que você execute o Python a partir de qualquer diretório no prompt de comando.

4. Siga as instruções do instalador para concluir a instalação.
5. Para verificar se o Python foi instalado corretamente, abra o Prompt de Comando (CMD) ou PowerShell e digite: `bash python --version` Você deverá ver a versão do Python instalada (ex: `Python 3.10.5`).

No macOS:

1. O macOS geralmente vem com uma versão antiga do Python 2.x pré-instalada. Para instalar o Python 3.x, a maneira mais recomendada é usar o Homebrew, um gerenciador de pacotes para macOS.
2. Se você não tem o Homebrew instalado, abra o Terminal e execute o seguinte comando: `bash /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`
3. Após a instalação do Homebrew, instale o Python 3.x com o seguinte comando: `bash brew install python`
4. Para verificar a instalação, abra o Terminal e digite: `bash python3 --version` Você deverá ver a versão do Python 3.x instalada.

No Linux (Ubuntu/Debian):

1. A maioria das distribuições Linux já vem com Python pré-instalado. No entanto, é uma boa prática garantir que você tenha a versão mais recente do Python 3.x e as ferramentas de desenvolvimento necessárias.
2. Abra o Terminal e atualize os pacotes: `bash sudo apt update sudo apt upgrade`
3. Instale o Python 3 e o pip (gerenciador de pacotes do Python): `bash sudo apt install python3 python3-pip`
4. Para verificar a instalação, digite: `bash python3 --version` Você deverá ver a versão do Python 3.x instalada.

2.2. Instalação do VSCode

O Visual Studio Code (VSCode) é um editor de código-fonte leve, mas poderoso, que oferece suporte a diversas linguagens de programação, incluindo Python. Ele será o

ambiente principal para editar e executar o código do Sistema de Almoxarifado. Siga os passos abaixo para instalá-lo:

1. Acesse o site oficial do Visual Studio Code: https://code.visualstudio.com/
2. Baixe o instalador compatível com o seu sistema operacional (Windows, macOS ou Linux).
3. Execute o instalador e siga as instruções. Durante a instalação, é recomendável marcar as opções para adicionar o VSCode ao PATH e criar atalhos na área de trabalho, se desejar.
4. Após a instalação, abra o VSCode. Para uma melhor experiência com Python, instale a extensão oficial do Python. Para fazer isso, clique no ícone de Extensões na barra lateral esquerda (ou pressione `Ctrl+Shift+X`), procure por "Python" e clique em "Install" (Instalar) na extensão publicada pela Microsoft.

2.3. Instalação das Dependências Python

O projeto do Sistema de Almoxarifado utiliza algumas bibliotecas Python que precisam ser instaladas para que o código funcione corretamente. Utilize o `pip`, o gerenciador de pacotes do Python, para instalar essas dependências. Abra o Terminal (Linux/macOS) ou Prompt de Comando/PowerShell (Windows) e execute os seguintes comandos:

```
pip install mysql-connector-python PyQt6 pytz
```

- `mysql-connector-python`: Esta biblioteca permite que o Python se conecte e interaja com bancos de dados MySQL.
- `PyQt6`: É um conjunto de bindings Python para a biblioteca de interface gráfica Qt. Ele é usado para construir a interface de usuário do aplicativo de almoxarifado.
- `pytz`: Fornece fusos horários do banco de dados `Olson` e permite a conversão de data/hora entre fusos horários, o que é útil para lidar com carimbos de data/hora de empréstimos e devoluções.

Após a execução do comando, o `pip` fará o download e a instalação de todas as bibliotecas necessárias e suas dependências. Certifique-se de que a instalação foi concluída sem erros.

2.4. Configuração do Banco de Dados MySQL

O Sistema de Almoxarifado utiliza o MySQL como seu sistema de gerenciamento de banco de dados. É fundamental ter o MySQL Server instalado e configurado corretamente para que o aplicativo funcione. Siga as instruções abaixo para instalar o MySQL Server.

2.4.1. Instalação do MySQL Server

No Windows:

1. Baixe o MySQL Installer para Windows no site oficial do MySQL: https://dev.mysql.com/downloads/installer/
2. Execute o instalador. Escolha a opção "Developer Default" para uma instalação completa com todas as ferramentas necessárias, ou "Custom" para selecionar apenas o MySQL Server e o MySQL Workbench (ferramenta gráfica para gerenciar o banco de dados).
3. Siga as instruções do instalador. Durante o processo, você será solicitado a definir uma senha para o usuário `root` do MySQL. **Anote esta senha**, pois ela será necessária para acessar o banco de dados.
4. Certifique-se de que o MySQL Server esteja configurado para iniciar automaticamente com o sistema.

No macOS:

1. Baixe o MySQL Community Server para macOS no site oficial do MySQL: https://dev.mysql.com/downloads/mysql/
2. Execute o arquivo `.dmg` e siga as instruções do instalador.
3. Durante a instalação, você será solicitado a definir uma senha para o usuário `root`. **Anote esta senha**.
4. Após a instalação, você pode gerenciar o MySQL Server através do painel de Preferências do Sistema (System Preferences) do macOS.

No Linux (Ubuntu/Debian):

1. Abra o Terminal e atualize os pacotes: `bash sudo apt update sudo apt upgrade`

2. Instale o MySQL Server: `bash sudo apt install mysql-server`
3. Durante a instalação, você será solicitado a definir uma senha para o usuário `root` do MySQL. **Anote esta senha**.
4. Execute o script de segurança para configurar o MySQL (opcional, mas recomendado): `bash sudo mysql_secure_installation` Siga as instruções, que incluem a validação da senha, remoção de usuários anônimos, desativação do login remoto do root e remoção do banco de dados de teste.
5. Verifique o status do serviço MySQL: `bash sudo systemctl status mysql` Ele deve estar `active (running)`.

2.4.2. Criação do Banco de Dados e Usuário

Após a instalação do MySQL Server, você precisará criar o banco de dados `almox` e um usuário com permissões adequadas para que o aplicativo possa se conectar a ele. Você pode fazer isso através do terminal ou de uma ferramenta gráfica como o MySQL Workbench.

Via Terminal (MySQL Client):

1. Abra o Terminal (Linux/macOS) ou Prompt de Comando/PowerShell (Windows).
2. Acesse o cliente MySQL como usuário `root` (ou outro usuário com privilégios administrativos), usando a senha que você definiu durante a instalação do MySQL Server: `bash mysql -u root -p` Será solicitada a senha. Digite-a e pressione Enter.
3. Dentro do prompt do MySQL (`mysql>`), execute os seguintes comandos para criar o banco de dados e o usuário. **Substitua `sua_senha_aqui` por uma senha forte e segura para o novo usuário do banco de dados.**

   ```sql
   CREATE DATABASE almox;

   CREATE USER 'almox_user'@'localhost' IDENTIFIED BY 'sua_senha_aqui';

   GRANT ALL PRIVILEGES ON almox.* TO 'almox_user'@'localhost';

   FLUSH PRIVILEGES;
   ```

   - `CREATE DATABASE almox;`: Cria um novo banco de dados chamado `almox`.

- `CREATE USER 'almox_user'@'localhost' IDENTIFIED BY 'sua_senha_aqui';`: Cria um novo usuário chamado `almox_user` que pode se conectar apenas a partir do `localhost` (o mesmo computador onde o MySQL está rodando) e define sua senha.
- `GRANT ALL PRIVILEGES ON almox.* TO 'almox_user'@'localhost';`: Concede todas as permissões (selecionar, inserir, atualizar, deletar, etc.) no banco de dados `almox` (e todas as suas tabelas) para o usuário `almox_user`.
- `FLUSH PRIVILEGES;`: Recarrega as tabelas de privilégios, garantindo que as novas permissões sejam aplicadas imediatamente.

4. Para sair do cliente MySQL, digite `exit;` e pressione Enter.

Via MySQL Workbench (Ferramenta Gráfica):

1. Abra o MySQL Workbench e conecte-se ao seu servidor MySQL usando as credenciais do usuário `root`.
2. No painel "Navigator", clique em "Schemas" e depois no ícone de "Create a new schema in the connected server" (um cilindro com um sinal de mais).
3. Digite `almox` como o nome do esquema e clique em "Apply".
4. Para criar o usuário, vá em "Management" -> "Users and Privileges".
5. Clique em "Add Account" (o ícone de um usuário com um sinal de mais).
6. Preencha os campos:
   - **Login Name:** `almox_user`
   - **Authentication Type:** Standard
   - **Password:** `sua_senha_aqui` (e repita no campo "Confirm Password")
   - **Host:** `localhost`
7. Vá para a aba "Schema Privileges", clique em "Add Entry...", selecione o esquema `almox` e clique em "OK".
8. Marque a opção "All" para conceder todas as permissões no esquema `almox` ao usuário `almox_user`.
9. Clique em "Apply" para salvar as alterações.

2.4.3. Estrutura do Banco de Dados (SQL)

Para que o aplicativo funcione corretamente, o banco de dados `almox` precisa ter a estrutura de tabelas definida. Abaixo está o script SQL para criar as tabelas `alunos`, `componentes` e `emprestimos`. Você pode executar este script usando o cliente MySQL no terminal ou através de uma ferramenta gráfica como o MySQL Workbench.

```sql
-- Criação da tabela 'alunos'
CREATE TABLE IF NOT EXISTS alunos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    matricula VARCHAR(50) UNIQUE NOT NULL,
    nome VARCHAR(255) NOT NULL,
    email VARCHAR(255),
    turma VARCHAR(100),
    foto LONGBLOB
);

-- Criação da tabela 'componentes'
CREATE TABLE IF NOT EXISTS componentes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    descricao TEXT,
    quantidade_total INT NOT NULL DEFAULT 0,
    quantidade_disponivel INT NOT NULL DEFAULT 0
);

-- Criação da tabela 'emprestimos'
CREATE TABLE IF NOT EXISTS emprestimos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    aluno_id INT NOT NULL,
    componente_id INT NOT NULL,
    quantidade INT NOT NULL,
    data_emprestimo DATE NOT NULL,
    data_devolucao DATE,
    FOREIGN KEY (aluno_id) REFERENCES alunos(id),
    FOREIGN KEY (componente_id) REFERENCES componentes(id)
);
```

Como executar o script SQL:

Via Terminal (MySQL Client):

1. Salve o script acima em um arquivo chamado `schema.sql` (ou qualquer outro nome com extensão `.sql`).
2. Abra o Terminal e navegue até o diretório onde você salvou o arquivo.
3. Execute o seguinte comando, substituindo `almox_user` e `sua_senha_aqui` pelas credenciais que você definiu na seção anterior: `bash mysql -u almox_user -p sua_senha_aqui almox < schema.sql` Ou, se você já estiver logado no cliente MySQL: `sql USE almox; SOURCE /caminho/para/o/seu/arquivo/schema.sql;`

Via MySQL Workbench (Ferramenta Gráfica):

1. Abra o MySQL Workbench e conecte-se ao seu servidor MySQL.
2. Abra um novo editor de consultas (File -> New Query Tab).
3. Cole o script SQL acima no editor.
4. Certifique-se de que o banco de dados `almox` esteja selecionado no menu suspenso de esquemas (geralmente no canto superior esquerdo).
5. Clique no botão "Execute" (o ícone de um raio) para rodar o script.

2.4.4. Importação de Dados Iniciais (Opcional)

Para popular o seu banco de dados com dados iniciais (alunos, componentes), você pode criar scripts SQL com comandos `INSERT` ou utilizar ferramentas de importação de dados. Esta etapa é opcional e depende da sua necessidade de ter dados pré-existentes no sistema.

Exemplo de script SQL para inserção de dados:

```sql
-- Inserir dados na tabela 'alunos'
INSERT INTO alunos (matricula, nome, email, turma) VALUES
("2023001", "João Silva", "joao.silva@liberato.com.br", "3TADS"),
("2023002", "Maria Souza", "maria.souza@liberato.com.br", "2TEL");

-- Inserir dados na tabela 'componentes'
INSERT INTO componentes (nome, descricao, quantidade_total,
quantidade_disponivel) VALUES
("Resistor 1k Ohm", "Resistor de filme de carbono, 1/4W", 100, 100),
("LED Vermelho", "Diodo Emissor de Luz, 5mm", 200, 200);
```

Você pode executar scripts SQL como este da mesma forma que executou o script de criação de tabelas (via terminal com `mysql -u ... < arquivo.sql` ou via MySQL Workbench).

Importação via Ferramentas de Banco de Dados:

Ferramentas como MySQL Workbench, phpMyAdmin (para ambientes web) ou DBeaver permitem importar dados de arquivos CSV, Excel ou outros formatos diretamente para as tabelas do seu banco de dados. Consulte a documentação da ferramenta de sua preferência para obter instruções detalhadas sobre como realizar a importação.

2.5. Configuração do Arquivo `default.png`

O aplicativo de Almoxarifado utiliza um arquivo de imagem chamado `default.png` para exibir uma imagem padrão quando a foto de um aluno não está disponível no banco de dados. Este arquivo deve estar localizado no mesmo diretório do script principal do aplicativo (`main.py` ou o nome do seu arquivo Python).

1. Crie ou obtenha uma imagem PNG que você deseja usar como padrão para fotos de alunos.
2. Renomeie este arquivo para `default.png`.
3. Coloque o arquivo `default.png` na mesma pasta onde o seu arquivo Python principal (o código que você me forneceu) está localizado.

Exemplo:

Se o seu código Python estiver em `/caminho/para/seu/projeto/almoxarifado_app.py`, o arquivo `default.png` deve estar em `/caminho/para/seu/projeto/default.png`.

Certifique-se de que o arquivo `default.png` esteja acessível e com o nome correto para evitar erros de carregamento de imagem no aplicativo.

3. Configuração e Execução do Código

Esta seção guiará você através dos passos para configurar o código do Sistema de Almoxarifado e executá-lo em seu ambiente de desenvolvimento, bem como a criação de um executável.

3.1. Clonando o Repositório (Opcional, para GitHub)

Se o seu código estiver hospedado em um repositório Git (como o GitHub), você pode cloná-lo para o seu ambiente local. Caso contrário, você pode simplesmente copiar os arquivos do projeto para uma pasta de sua preferência.

1. **Instale o Git (se ainda não tiver):**
    - **Windows:** Baixe o instalador em https://git-scm.com/download/win e siga as instruções.

- **macOS:** Instale via Homebrew: `brew install git`.
- **Linux (Ubuntu/Debian):** `sudo apt install git`.

2. **Abra o Terminal ou Prompt de Comando** e navegue até o diretório onde você deseja clonar o projeto. Por exemplo: `bash cd C:\Users\SeuUsuario\Documentos\Projetos` ou `bash cd ~/Documentos/Projetos`

3. **Clone o repositório** usando o comando `git clone`, substituindo `URL_DO_SEU_REPOSITORIO` pela URL real do seu repositório Git: `bash git clone URL_DO_SEU_REPOSITORIO` Isso criará uma nova pasta com o nome do seu repositório contendo todos os arquivos do projeto.

4. **Navegue para a pasta do projeto clonado:** `bash cd nome_do_seu_repositorio`

Se você não estiver usando um repositório Git, simplesmente crie uma pasta para o projeto e copie todos os arquivos do código Python e o `default.png` para dentro dela.

3.2. Configurando as Credenciais do Banco de Dados no Código

O seu código Python contém as credenciais de conexão com o banco de dados MySQL. É crucial que estas credenciais correspondam às que você configurou na seção `2.4.2. Criação do Banco de Dados e Usuário`. Você precisará editar o arquivo Python principal para atualizar essas informações.

1. Abra o arquivo Python principal do seu projeto (o código que você me forneceu) no VSCode.

2. Localize as seguintes linhas no início do arquivo:

   `python DB_NAME = "almox" DB_USER = "root" DB_PASSWORD = "manulauro5308" DB_HOST = "localhost" DB_PORT = 3306`

3. **Altere os valores** de `DB_USER` e `DB_PASSWORD` para as credenciais do usuário `almox_user` que você criou para o banco de dados `almox`. Se você alterou o nome do banco de dados, `DB_NAME`, ou o host/porta, `DB_HOST` / `DB_PORT`, também os ajuste aqui.

   Exemplo:

Se você criou o usuário `almox_user` com a senha `minhasenhaforte`:

```
python DB_NAME = "almox" DB_USER = "almox_user" DB_PASSWORD = "minhasenhaforte" DB_HOST = "localhost" DB_PORT = 3306
```

4. Salve o arquivo (`Ctrl+S` ou `Cmd+S`).

É fundamental que essas credenciais estejam corretas para que o aplicativo consiga se conectar ao banco de dados e funcionar adequadamente.

3.3. Executando o Aplicativo no VSCode

Com todas as dependências instaladas e as credenciais do banco de dados configuradas, você está pronto para executar o aplicativo diretamente do VSCode.

1. **Abra a pasta do projeto no VSCode:**
    - No VSCode, vá em `File > Open Folder...` (Arquivo > Abrir Pasta...) e selecione a pasta onde você clonou ou copiou os arquivos do projeto.
2. **Abra o arquivo Python principal:**
    - No painel "Explorer" (Explorador) à esquerda, clique no arquivo Python principal (o que contém a classe `AlmoxarifadoApp` e o bloco `if __name__ == "__main__":`).
3. **Execute o arquivo:**
    - Com o arquivo Python aberto, você pode executá-lo de algumas maneiras:
        - **Botão "Run" (Executar):** No canto superior direito do editor, clique no ícone de "Play" (um triângulo verde) ou "Run Python File" (Executar Arquivo Python).
        - **Terminal Integrado:** Abra o Terminal Integrado do VSCode (`Terminal > New Terminal` ou `Ctrl+Shift+``). Certifique-se de que você está no diretório raiz do seu projeto. Em seguida, execute o comando: `bash python seu_arquivo_principal.py` (Substitua `seu_arquivo_principal.py` pelo nome real do seu arquivo Python, por exemplo, `almoxarifado_app.py`).

4. O aplicativo do Sistema de Almoxarifado deverá ser iniciado em uma nova janela. Se ocorrerem erros, verifique o terminal do VSCode para mensagens de erro e consulte a seção de Solução de Problemas deste manual.

3.4. Como Modificar Funções Existentes e a Interface do Usuário

Uma das grandes vantagens de ter o código-fonte é a capacidade de adaptá-lo às suas necessidades específicas. Esta seção detalha como você pode modificar as funções existentes e a interface do usuário (UI) do Sistema de Almoxarifado, utilizando o VSCode e compreendendo a estrutura do código PyQt6.

3.4.1. Entendendo a Estrutura do Código PyQt6

O aplicativo é construído usando a biblioteca PyQt6, que segue um padrão de programação orientada a objetos. A classe principal `AlmoxarifadoApp` herda de `QWidget` e é responsável por toda a interface e lógica do aplicativo. A UI é construída programaticamente, ou seja, os elementos visuais (botões, campos de texto, tabelas) são criados e organizados diretamente no código Python.

Principais Componentes da UI (Widgets):

- `QWidget`: A classe base para todos os objetos de interface do usuário.
- `QGridLayout`, `QVBoxLayout`, `QHBoxLayout`: Gerenciadores de layout que organizam os widgets na janela.
- `QGroupBox`: Usado para agrupar logicamente outros widgets, geralmente com um título.
- `QLabel`: Exibe texto ou imagens (como a foto do aluno).
- `QTableWidget`: Exibe dados em formato de tabela (usado para empréstimos).
- `QPushButton`: Botões clicáveis.
- `QLineEdit`: Campos de entrada de texto.
- `QListWidget`: Exibe uma lista de itens selecionáveis.
- `QRadioButton`: Botões de opção (para seleção de tipo de pesquisa).
- `QMessageBox`: Usado para exibir mensagens de aviso, erro ou informação ao usuário.

Conexão de Sinais e Slots:

No PyQt6, a interação do usuário (como clicar em um botão ou digitar em um campo de texto) é tratada através de um mecanismo chamado "sinais e slots". Um **sinal** é emitido por um widget quando um evento específico ocorre (ex: `clicked` de um `QPushButton`). Um **slot** é um método (função) que é executado em resposta a esse sinal. Você verá linhas como:

```
self.botao_pesquisa.clicked.connect(self.pesquisar_aluno)
```

Isso significa que, quando o `botao_pesquisa` é clicado, o método `pesquisar_aluno` da classe `AlmoxarifadoApp` será executado.

3.4.2. Modificando a Interface (Layout e Widgets)

Para alterar a aparência ou o layout do aplicativo, você precisará modificar os métodos `criar_grupo_detalhes`, `criar_grupo_pesquisa`, `criar_grupo_pesquisa_componente` e `criar_grupo_emprestimos`.

Exemplo 1: Adicionar um novo campo de texto para o telefone do aluno

1. **Localize o método `criar_grupo_detalhes`:** Este método é responsável por criar a seção de detalhes do aluno.
2. **Adicione um novo `QLabel`:** Dentro do layout `info` (que é um `QVBoxLayout`), adicione uma nova linha para o telefone.

   ```python

# ... dentro de criar_grupo_detalhes

info = QVBoxLayout() self.label_matricula = QLabel("Matrícula: -") self.label_nome = QLabel("Nome: -") self.label_email = QLabel("E-mail: -") self.label_turma = QLabel("Turma: -") self.label_telefone = QLabel("Telefone: -") # <--- Nova linha info.addWidget(self.label_matricula) info.addWidget(self.label_nome) info.addWidget(self.label_email) info.addWidget(self.label_turma) info.addWidget(self.label_telefone) # <--- Nova linha
   ```

...

```

3. **Atualize o método `mostrar_detalhes_aluno`:** Para que o novo campo exiba os dados, você precisará:
    - Adicionar uma coluna `telefone` à sua tabela `alunos` no banco de dados (se ainda não o fez).
    - Modificar a consulta SQL para selecionar também o campo `telefone`.
    - Atualizar o `QLabel` com o valor do telefone.

```python

# ... dentro de mostrar_detalhes_aluno

---

cursor.execute( "SELECT nome, email, turma, foto, telefone " # <--- Adicione 'telefone' "FROM alunos WHERE matricula = %s", (matricula,) ) aluno = cursor.fetchone()

if aluno: nome, email, turma, foto_bytes, telefone = aluno # <--- Desempacote 'telefone' # ... (outras linhas de label.setText) self.label_telefone.setText(f"Telefone: {telefone}") # <--- Nova linha # ... ```

**Exemplo 2: Mudar a posição de um botão**

Se você quisesse mover o botão "Registrar Devolução" para dentro do layout `form` no grupo de empréstimos, você alteraria o método `criar_grupo_emprestimos`:

```
 # ... dentro de criar_grupo_emprestimos
form = QHBoxLayout()
# ... (outros widgets no form)
form.addWidget(self.botao_adicionar)
form.addWidget(self.botao_devolver) # <--- Mova esta linha para cá
layout.addLayout(form)

Remova a linha original:
layout.addWidget(self.botao_devolver)
...
```

3.4.3. Modificando a Lógica (Funções)

As funções do aplicativo são os métodos da classe `AlmoxarifadoApp`. Cada método geralmente corresponde a uma ação específica ou a uma parte da lógica do negócio.

Exemplo 1: Alterar a lógica de pesquisa de aluno

O método `pesquisar_aluno` é responsável por buscar alunos no banco de dados. Se você quisesse que a pesquisa fosse case-insensitive (não diferencia maiúsculas de minúsculas) ou buscasse também por e-mail, você modificaria a consulta SQL dentro deste método.

```
 # ... dentro de pesquisar_aluno
termo = self.pesquisa_input.text().strip()
# ...

cursor.execute(
    "SELECT matricula, nome FROM alunos "
    "WHERE LOWER(matricula) LIKE LOWER(%s) OR LOWER(nome) LIKE LOWER(%s) OR
LOWER(email) LIKE LOWER(%s)", # <--- Modificação da consulta
    (f"%{termo}%", f"%{termo}%", f"%{termo}%")
)
# ...
```

Exemplo 2: Adicionar validação extra ao adicionar empréstimo

No método `adicionar_emprestimo`, você pode adicionar validações para garantir que a quantidade emprestada não exceda a quantidade disponível de um componente.

```
 # ... dentro de adicionar_emprestimo
# ... (validações existentes)

# Obter quantidade disponível do componente
cursor.execute(
    "SELECT quantidade_disponivel FROM componentes WHERE id = %s",
    (comp_id,)
)
quantidade_disponivel = cursor.fetchone()[0]

if int(qtd) > quantidade_disponivel:
    QMessageBox.warning(self, "Aviso", "Quantidade solicitada excede a
disponível.")
    return

# ... (restante da lógica de inserção)

# Atualizar quantidade_disponivel após o empréstimo
cursor.execute(
    "UPDATE componentes SET quantidade_disponivel = quantidade_disponivel - %s
WHERE id = %s",
    (int(qtd), comp_id)
)
conn.commit()
# ...
```

3.4.4. Dicas para Modificação

- **Use o VSCode para Navegar:** O VSCode possui excelentes ferramentas para navegar pelo código. Use "Go to Definition" (`F12`) para pular para a definição de uma função ou variável, e "Find All References" (`Shift+F12`) para ver onde uma função ou variável é usada.
- **Comentários:** Adicione comentários ao seu código para explicar suas modificações e a lógica por trás delas. Isso ajuda você e outros a entenderem o código no futuro.
- **Testes:** Após cada modificação, teste o aplicativo para garantir que suas alterações funcionaram como esperado e não introduziram novos erros.
- **Controle de Versão:** Se você estiver usando Git (como sugerido na seção 3.1), faça commits frequentes de suas alterações. Isso permite que você reverta para versões anteriores se algo der errado.
- **Documentação PyQt6:** Para modificações mais avançadas, consulte a documentação oficial do PyQt6. Ela é uma fonte rica de informações sobre todos os widgets e funcionalidades disponíveis: https://www.riverbankcomputing.com/static/Docs/PyQt6/

Ao seguir estas diretrizes, você poderá personalizar e estender o Sistema de Almoxarifado para atender às necessidades específicas da Fundação Escola Técnica Liberato Salzano Vieira da Cunha.

3.5. Criando um Executável Funcional

Para que o Sistema de Almoxarifado possa ser distribuído e executado em computadores que não possuem o Python e suas dependências instaladas, você pode criar um executável. Uma ferramenta popular e eficaz para isso é o `PyInstaller`.

O `PyInstaller` empacota seu aplicativo Python e todas as suas dependências em um único pacote executável. Isso significa que o usuário final não precisará instalar Python, `mysql-connector-python`, `PyQt6`, ou `pytz` separadamente.

3.5.1. Instalação do PyInstaller

Abra o Terminal (Linux/macOS) ou Prompt de Comando/PowerShell (Windows) e execute o seguinte comando para instalar o `PyInstaller`:

```
pip install pyinstaller
```

3.5.2. Empacotando o Aplicativo

Após a instalação do `PyInstaller`, navegue até o diretório onde o seu arquivo Python principal (`seu_arquivo_principal.py`) e o arquivo `default.png` estão localizados. Em seguida, execute o comando `pyinstaller` com as opções apropriadas:

```
pyinstaller --onefile --windowed --add-data "default.png:."
seu_arquivo_principal.py
```

Vamos entender as opções:

- `--onefile`: Cria um único arquivo executável, o que é mais conveniente para distribuição.
- `--windowed` ou `-w`: Indica que o aplicativo é uma aplicação de janela (GUI) e não deve abrir uma janela de console (terminal) ao ser executado. Use esta opção para aplicativos PyQt6.

- `--add-data "default.png:."`: Esta é uma opção crucial para incluir arquivos adicionais que seu aplicativo precisa. Neste caso, estamos adicionando o `default.png`. O formato é `"<caminho_do_arquivo_origem>:<caminho_do_destino_no_executavel>"`. O `.` no destino significa que o `default.png` será colocado na raiz do diretório temporário do executável, onde seu script Python espera encontrá-lo.
- `seu_arquivo_principal.py`: Substitua pelo nome do seu arquivo Python principal (ex: `almoxarifado_app.py`).

Exemplo Completo:

Se o seu arquivo principal for `almoxarifado_app.py`:

```
pyinstaller --onefile --windowed --add-data "default.png:." almoxarifado_app.py
```

3.5.3. Localizando o Executável

Após a execução do comando `pyinstaller`, ele criará algumas pastas no seu diretório de projeto:

- `build/`: Contém arquivos temporários gerados durante o processo de empacotamento.
- `dist/`: Esta pasta conterá o seu executável final.

O executável estará dentro da pasta `dist`. Por exemplo, no Windows, o arquivo será `almoxarifado_app.exe`. No Linux ou macOS, será `almoxarifado_app` (sem extensão).

3.5.4. Considerações Importantes

- **Tamanho do Executável:** Executáveis gerados com `--onefile` podem ser grandes, pois incluem o interpretador Python e todas as bibliotecas empacotadas.
- **Dependências Externas (MySQL):** O executável empacota apenas as dependências Python. O MySQL Server **ainda precisará estar instalado e rodando** no computador onde o executável será utilizado, pois o aplicativo se conecta a um servidor MySQL externo.
- **Testes:** Sempre teste o executável em diferentes máquinas para garantir que ele funcione conforme o esperado e que todas as dependências (como o

`default.png`) foram incluídas corretamente.

- **Plataforma:** O `PyInstaller` cria executáveis específicos para o sistema operacional onde ele é executado. Ou seja, um executável criado no Windows só funcionará no Windows, um no macOS só no macOS, e um no Linux só no Linux.

4. Funcionalidades do Sistema

Esta seção descreve as principais funcionalidades do Sistema de Almoxarifado, guiando o usuário através de cada recurso.

4.1. Pesquisa de Aluno

A funcionalidade de pesquisa de aluno permite que você localize rapidamente os registros de alunos no sistema, seja pela matrícula ou pelo nome. Uma vez que um aluno é selecionado, seus detalhes são exibidos e seu histórico de empréstimos é carregado.

Passos para Pesquisar um Aluno:

1. **Localize o painel "Pesquisar Aluno":** Este painel está localizado na parte superior direita da janela principal do aplicativo.

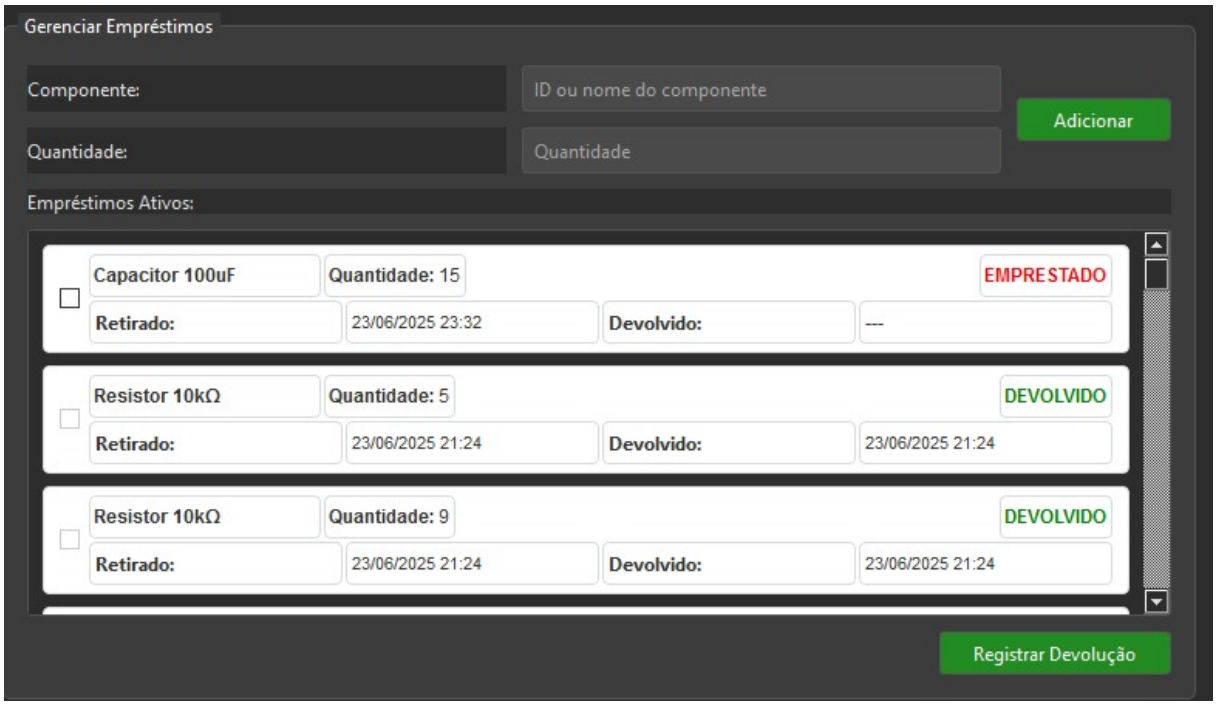

Figura 1: Painel de Detalhes do Aluno (Modo Escuro)

2. **Digite o termo de pesquisa:** No campo de texto "Digite matrícula ou nome", insira a matrícula completa ou parte do nome do aluno que você deseja encontrar.

3. **Clique no botão "Pesquisar":** Após digitar o termo, clique no botão "Pesquisar" para iniciar a busca no banco de dados.

4. **Selecione o aluno na lista:** Os resultados da pesquisa serão exibidos na lista abaixo do botão "Pesquisar". Se houver múltiplos resultados, clique no nome do aluno desejado na lista. Ao clicar, os detalhes do aluno (Matrícula, Nome, E-mail, Turma e Foto) serão preenchidos no painel "Detalhes do Aluno" (canto superior esquerdo) e a tabela "Lista de Empréstimos" (parte inferior da tela) será atualizada com os empréstimos ativos e históricos desse aluno.

    - Se nenhum aluno for encontrado, uma mensagem "Nenhum aluno encontrado." será exibida na lista.

Exemplo de Uso:

- Para encontrar "João Silva", digite `João` ou `Silva` ou a matrícula `2023001` no campo de pesquisa e clique em "Pesquisar".
- Selecione "2023001 - João Silva" na lista para ver seus detalhes e empréstimos.

4.2. Pesquisa de Componente

A funcionalidade de pesquisa de componente permite que você encontre componentes específicos no almoxarifado, seja pelo nome ou pelo código do componente. Isso é útil para verificar a disponibilidade de um item ou para preparar um empréstimo.

Passos para Pesquisar um Componente:

1. **Localize o painel "Pesquisar Componente":** Este painel está localizado na parte central da janela principal do aplicativo.

Figura 2: Painel de Pesquisa de Componente

2. **Digite o termo de pesquisa:** No campo de texto "Digite nome ou código do componente", insira o nome completo ou parcial do componente, ou o seu código de identificação.

3. **Selecione o tipo de pesquisa:**

    - **Por Nome:** Marque a opção "Por Nome" se você estiver pesquisando pelo nome do componente.
    - **Por Código:** Marque a opção "Por Código" se você estiver pesquisando pelo código do componente.

4. **Clique no botão "Buscar Componente":** Após digitar o termo e selecionar o tipo de pesquisa, clique no botão "Buscar Componente" para iniciar a busca.

5. **Selecione o componente na lista:** Os resultados da pesquisa serão exibidos na lista abaixo do botão "Buscar Componente".
    - **Para ver detalhes ou selecionar para empréstimo:** Dê um **clique duplo** no componente desejado na lista. O ID do componente será automaticamente preenchido no campo "ID ou nome do componente" na seção "Lista de Empréstimos", pronto para ser adicionado a um novo empréstimo.
    - Se nenhum componente for encontrado, uma mensagem "Nenhum componente encontrado." será exibida na lista.

Exemplo de Uso:

- Para encontrar "Resistor 1k Ohm", digite `Resistor` no campo de pesquisa, selecione "Por Nome" e clique em "Buscar Componente".
- Para encontrar o componente com ID `123`, digite `123` no campo de pesquisa, selecione "Por Código" e clique em "Buscar Componente".
- Dê um clique duplo em "Resistor 1k Ohm" na lista para que seu ID seja preenchido automaticamente no campo de empréstimo.

4.3. Registro de Empréstimos

O registro de empréstimos é a funcionalidade central do sistema, permitindo que você registre a saída de componentes para um aluno selecionado. Antes de registrar um empréstimo, certifique-se de que o aluno desejado esteja selecionado no painel "Detalhes do Aluno" (ver seção 4.1).

Passos para Registrar um Empréstimo:

1. **Selecione o Aluno:** Primeiro, pesquise e selecione o aluno que está realizando o empréstimo, conforme descrito na seção 4.1. Os detalhes do aluno aparecerão no painel "Detalhes do Aluno" e a tabela "Lista de Empréstimos" será atualizada com seus empréstimos atuais e passados.
2. **Localize o painel "Lista de Empréstimos":** Este painel está localizado na parte inferior da janela principal do aplicativo.

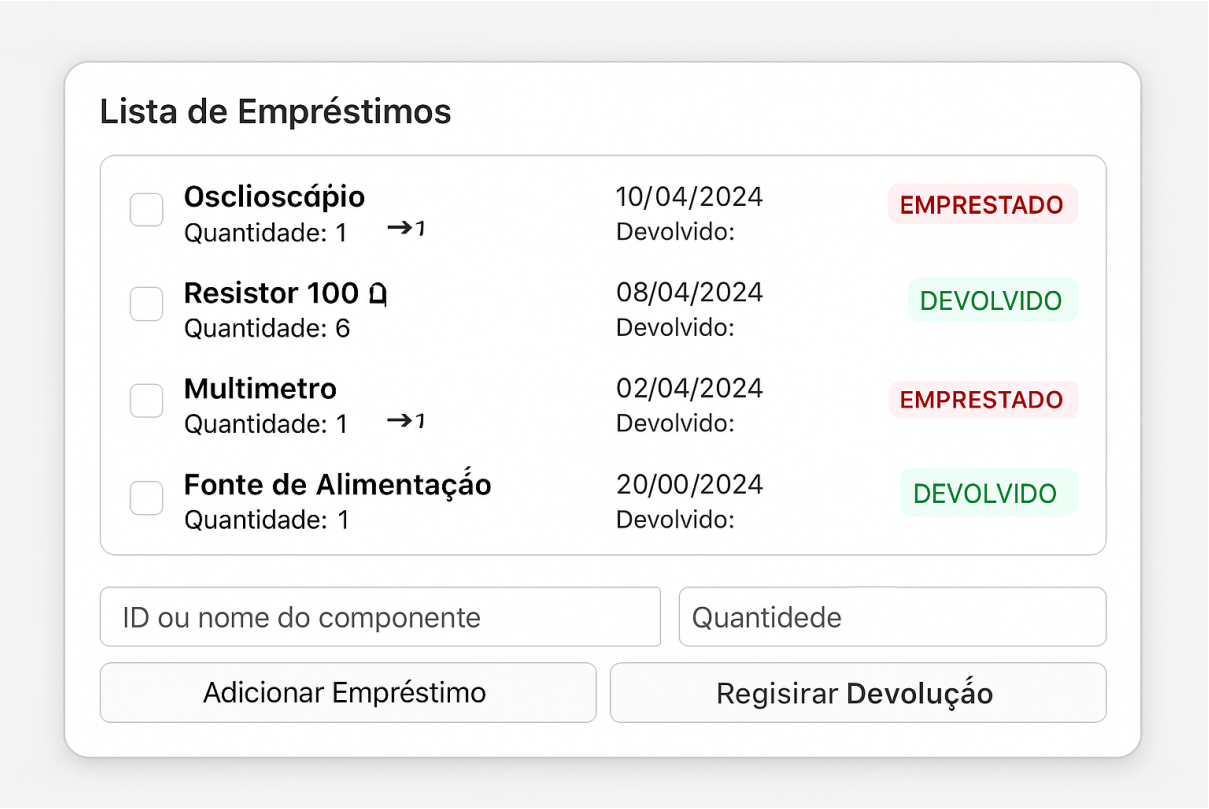

Figura 3: Painel de Empréstimos

3. **Insira o Componente:** No campo de texto "ID ou nome do componente", você pode:
    - Digitar o ID exato do componente.
    - Digitar o nome do componente (o sistema tentará encontrar o ID correspondente).
    - Alternativamente, você pode usar a funcionalidade de "Pesquisar Componente" (seção 4.2) e dar um clique duplo no componente desejado na lista de resultados; o ID será preenchido automaticamente neste campo.
4. **Insira a Quantidade:** No campo de texto "Quantidade", digite o número de unidades do componente que estão sendo emprestadas.
5. **Clique no botão "Adicionar Empréstimo":** Após preencher o ID/nome do componente e a quantidade, clique no botão "Adicionar Empréstimo".
6. **Confirmação:** Uma mensagem de "Sucesso" será exibida se o empréstimo for registrado com êxito. A tabela "Lista de Empréstimos" será atualizada para incluir o novo registro.

- **Validações:** O sistema verificará se um aluno foi selecionado e se os campos de componente e quantidade foram preenchidos corretamente. Erros serão exibidos em mensagens de aviso.

Exemplo de Uso:

- Após selecionar "João Silva", digite `Resistor 1k Ohm` no campo de componente e `5` no campo de quantidade.
- Clique em "Adicionar Empréstimo". O empréstimo será registrado e aparecerá na tabela.

4.4. Registro de Devoluções

O registro de devoluções permite que você marque um componente emprestado como devolvido, atualizando o status no banco de dados. Para registrar uma devolução, o empréstimo correspondente deve estar visível na tabela "Lista de Empréstimos" do aluno selecionado.

Passos para Registrar uma Devolução:

1. **Selecione o Aluno:** Certifique-se de que o aluno que está devolvendo o componente esteja selecionado no painel "Detalhes do Aluno" (ver seção 4.1). A tabela "Lista de Empréstimos" deve exibir os empréstimos desse aluno.
2. **Selecione o Empréstimo na Tabela:** Na tabela "Lista de Empréstimos", clique na linha correspondente ao empréstimo que você deseja registrar como devolvido. A linha selecionada ficará destacada.
3. **Clique no botão "Registrar Devolução":** Após selecionar a linha do empréstimo, clique no botão "Registrar Devolução", localizado abaixo dos campos de adição de empréstimo.
4. **Confirmação:** Uma mensagem de "Sucesso" será exibida se a devolução for registrada com êxito. A tabela "Lista de Empréstimos" será atualizada, e a coluna "Devolução" para o item devolvido mostrará a data atual, indicando que o item foi devolvido.
    - **Validações:** O sistema verificará se um empréstimo foi selecionado na tabela. Se nenhum empréstimo estiver selecionado, uma mensagem de aviso será exibida.

Exemplo de Uso:

- Selecione "João Silva" para ver seus empréstimos.
- Clique na linha do empréstimo do "Resistor 1k Ohm" que você registrou anteriormente.
- Clique em "Registrar Devolução". A data de devolução será preenchida na tabela.

5. Solução de Problemas

Esta seção aborda alguns problemas comuns que você pode encontrar ao instalar ou executar o Sistema de Almoxarifado e como resolvê-los.

5.1. Erro de Conexão com o Banco de Dados

Mensagem de Erro Comum: `mysql.connector.errors.ProgrammingError: 1045 (28000): Access denied for user '...'@'localhost' (using password: YES)` ou `mysql.connector.errors.InterfaceError: 2003: Can't connect to MySQL server on 'localhost:3306' (...).`

Causas Possíveis:

- **Credenciais Incorretas:** O nome de usuário (`DB_USER`) ou a senha (`DB_PASSWORD`) no seu código Python (`almoxarifado_app.py`) não correspondem às credenciais do usuário do MySQL que você configurou.
- **Servidor MySQL Não Iniciado:** O serviço do MySQL Server não está rodando no seu computador.
- **Porta Incorreta:** A porta (`DB_PORT`) especificada no código Python não é a porta correta do seu servidor MySQL (a porta padrão é 3306).
- **Usuário Sem Permissões:** O usuário do MySQL não tem as permissões necessárias para acessar o banco de dados `almox` .

Soluções:

1. **Verifique as Credenciais:** Revise a seção `3.2. Configurando as Credenciais do Banco de Dados no Código` e certifique-se de que `DB_USER` e `DB_PASSWORD` no seu código correspondem exatamente ao usuário e senha que você criou no MySQL.

2. **Inicie o Servidor MySQL:**
    - **Windows:** Procure por "Services" (Serviços) no menu Iniciar, encontre "MySQL" ou "MySQL80" (ou a versão que você instalou) e certifique-se de que o status é "Running" (Em execução). Se não estiver, clique com o botão direito e selecione "Start" (Iniciar).
    - **macOS:** Vá em "System Preferences" (Preferências do Sistema) -> "MySQL" e clique em "Start MySQL Server".
    - **Linux:** Abra o terminal e execute `sudo systemctl start mysql`.
3. **Verifique a Porta:** A porta padrão do MySQL é 3306. Se você a alterou, certifique-se de que `DB_PORT` no seu código reflita essa mudança.
4. **Verifique as Permissões do Usuário:** Revise a seção `2.4.2. Criação do Banco de Dados e Usuário` para garantir que o usuário `almox_user` tenha `ALL PRIVILEGES` no banco de dados `almox`.

5.2. Erro de Módulo Não Encontrado

Mensagem de Erro Comum: `ModuleNotFoundError: No module named 'mysql.connector'` ou `ModuleNotFoundError: No module named 'PyQt6'`

Causas Possíveis:

- **Dependência Não Instalada:** A biblioteca Python necessária não foi instalada.
- **Ambiente Python Incorreto:** Você está executando o código com uma versão do Python ou um ambiente virtual onde as bibliotecas não foram instaladas.

Soluções:

1. **Instale as Dependências:** Revise a seção `2.3. Instalação das Dependências Python` e execute o comando `pip install mysql-connector-python PyQt6 pytz` novamente para garantir que todas as bibliotecas estejam instaladas.
2. **Verifique o Ambiente Python:** No VSCode, certifique-se de que você selecionou o interpretador Python correto onde as bibliotecas foram instaladas. Você pode fazer isso clicando na versão do Python na barra de status inferior do VSCode e selecionando o interpretador apropriado.

5.3. Imagem `default.png` Não Encontrada

Mensagem de Erro Comum: O aplicativo inicia, mas a imagem padrão do aluno não aparece, ou um erro relacionado a `default.png` é exibido no console.

Causas Possíveis:

- **Arquivo Ausente ou Nome Incorreto:** O arquivo `default.png` não está na mesma pasta que o seu script Python principal, ou o nome do arquivo está incorreto (sensível a maiúsculas e minúsculas).

Soluções:

1. **Verifique a Localização e o Nome:** Revise a seção `2.5. Configuração do Arquivo default.png` e certifique-se de que o arquivo `default.png` está na pasta correta e com o nome exato.

5.4. Erros de Sintaxe ou Lógica no Código

Mensagem de Erro Comum: `SyntaxError`, `IndentationError`, ou erros lógicos que causam comportamento inesperado do aplicativo.

Causas Possíveis:

- **Erros de Digitação:** Pequenos erros de digitação no código.
- **Indentação Incorreta:** Python usa indentação para definir blocos de código; indentação incorreta causará erros.
- **Lógica de Programação:** O fluxo lógico do programa não está correto para a funcionalidade desejada.

Soluções:

1. **Revise o Código:** Use o VSCode para revisar o código. O VSCode destaca erros de sintaxe e indentação. Preste atenção às mensagens de erro no terminal, pois elas geralmente indicam a linha e o tipo de erro.
2. **Depuração:** Utilize as ferramentas de depuração do VSCode para percorrer o código passo a passo, inspecionar variáveis e entender o fluxo de execução. Isso é extremamente útil para identificar erros lógicos.

3. **Consulte a Documentação:** Para problemas relacionados a funções específicas do Python, PyQt6 ou `mysql.connector`, consulte a documentação oficial dessas bibliotecas.

6. Apêndice

6.1. Estrutura das Tabelas do Banco de Dados

O banco de dados `almox` é composto por três tabelas principais, que armazenam as informações sobre alunos, componentes e os registros de empréstimos. Abaixo, detalhamos a estrutura de cada tabela:

Tabela `alunos`:

Coluna	Tipo de Dado	Descrição	Observações
`id`	`INT`	Chave primária, identificador único do aluno.	Auto-incremento.
`matricula`	`VARCHAR(50)`	Matrícula do aluno.	Única e não nula.
`nome`	`VARCHAR(255)`	Nome completo do aluno.	Não nulo.
`email`	`VARCHAR(255)`	Endereço de e-mail do aluno.	Pode ser nulo.
`turma`	`VARCHAR(100)`	Turma ou curso do aluno.	Pode ser nulo.

Tabela `componentes`:

Coluna	Tipo de Dado	Descrição	Observações
`id`	`INT`	Chave primária, identificador único do componente.	Auto-incremento.
`nome`	`VARCHAR(255)`	Nome do componente.	Não nulo.
`descricao`	`TEXT`	Descrição detalhada do componente.	Pode ser nulo.

Tabela `emprestimos`:

Coluna	Tipo de Dado	Descrição	Observações
`id`	`INT`	Chave primária, identificador único do empréstimo.	Auto-incremento.
`aluno_id`	`INT`	Chave estrangeira, referencia a tabela `alunos`.	Não nulo.
`componente_id`	`INT`	Chave estrangeira, referencia a tabela `componentes`.	Não nulo.
`quantidade`	`INT`	Quantidade de componentes emprestados.	Não nulo.
`data_emprestimo`	`DATETIME`	Data e hora em que o empréstimo foi realizado.	Não nulo.
`data_devolucao`	`DATETIME`	Data e hora em que o componente foi devolvido.	Pode ser nulo (se o componente ainda não foi devolvido).
`status`	`VARCHAR(50)`	Status do empréstimo (Emprestado/Devolvido).	Não nulo.

6.2. Script SQL Completo para Criação do Banco de Dados e Tabelas

Este script pode ser usado para criar o banco de dados `almox` e todas as tabelas necessárias (`alunos`, `componentes`, `emprestimos`) em seu servidor MySQL. Certifique-se de ter as permissões adequadas para criar bancos de dados e tabelas.

```sql
-- Criação do banco de dados
CREATE DATABASE IF NOT EXISTS almox;

-- Seleciona o banco de dados para uso
USE almox;

-- Criação da tabela 'alunos'
CREATE TABLE IF NOT EXISTS alunos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    matricula VARCHAR(50) UNIQUE NOT NULL,
    nome VARCHAR(255) NOT NULL,
    email VARCHAR(255),
    turma VARCHAR(100),
    foto LONGBLOB
);

-- Criação da tabela 'componentes'
CREATE TABLE IF NOT EXISTS componentes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    descricao TEXT,
    quantidade_total INT NOT NULL DEFAULT 0,
    quantidade_disponivel INT NOT NULL DEFAULT 0
);

-- Criação da tabela 'emprestimos'
CREATE TABLE IF NOT EXISTS emprestimos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    aluno_id INT NOT NULL,
    componente_id INT NOT NULL,
    quantidade INT NOT NULL,
    data_emprestimo DATE NOT NULL,
    data_devolucao DATE,
    FOREIGN KEY (aluno_id) REFERENCES alunos(id),
    FOREIGN KEY (componente_id) REFERENCES componentes(id)
);
```

Instruções para Execução do Script SQL:

Você pode executar este script usando o cliente MySQL no terminal ou através de uma ferramenta gráfica como o MySQL Workbench, conforme detalhado na seção `2.4.3. Estrutura do Banco de Dados (SQL)`.

2.4.5. Conexão com Banco de Dados Remoto (Opcional)

Se o seu banco de dados MySQL não estiver rodando no mesmo computador que o aplicativo (ou seja, não está em `localhost`), você precisará ajustar a configuração de `DB_HOST` no seu código Python e garantir que o servidor MySQL esteja acessível remotamente.

1. Configurando o `DB_HOST` no Código Python:

No arquivo Python principal do seu projeto, localize as credenciais de conexão com o banco de dados (conforme a seção `3.2. Configurando as Credenciais do Banco de Dados no Código`). Altere o valor de `DB_HOST` para o endereço IP ou nome de domínio do servidor onde o MySQL está hospedado.

Exemplo:

Se o seu servidor MySQL estiver em `192.168.1.100`:

```
DB_NAME = "almox"
DB_USER = "almox_user"
DB_PASSWORD = "minhasenhaforte"
DB_HOST = "192.168.1.100" # <--- Endereço IP do servidor remoto
DB_PORT = 3306
```

2. Configurando o Usuário para Acesso Remoto:

Por padrão, o usuário `almox_user` foi criado para acessar o banco de dados apenas a partir de `localhost`. Para permitir o acesso de um servidor remoto, você precisará modificar as permissões do usuário no MySQL. Conecte-se ao seu servidor MySQL (como `root`) e execute o seguinte comando, substituindo `seu_ip_aqui` pelo endereço IP do computador onde o aplicativo será executado (ou `%` para permitir acesso de qualquer IP, o que não é recomendado para ambientes de produção):

```
ALTER USER 'almox_user'@'localhost' IDENTIFIED BY 'sua_senha_aqui';
CREATE USER 'almox_user'@'seu_ip_aqui' IDENTIFIED BY 'sua_senha_aqui';
GRANT ALL PRIVILEGES ON almox.* TO 'almox_user'@'seu_ip_aqui';
FLUSH PRIVILEGES;
```

Se você deseja permitir acesso de qualquer host (menos seguro, mas útil para testes ou redes controladas):

```
CREATE USER 'almox_user'@'%' IDENTIFIED BY 'sua_senha_aqui';
GRANT ALL PRIVILEGES ON almox.* TO 'almox_user'@'%';
FLUSH PRIVILEGES;
```

3. Configurações de Rede e Firewall:

- **Firewall do Servidor MySQL:** Certifique-se de que o firewall do servidor onde o MySQL está rodando permite conexões de entrada na porta do MySQL (padrão 3306) a partir do endereço IP do computador que executará o aplicativo. Você pode precisar adicionar uma regra de firewall para permitir essa conexão.

- **Rede:** Verifique se o computador que executa o aplicativo tem conectividade de rede com o servidor MySQL. Isso pode envolver configurações de rede, VPNs ou outras configurações de infraestrutura.

Ao seguir esses passos, seu aplicativo poderá se conectar a um banco de dados MySQL hospedado em um servidor remoto, proporcionando maior flexibilidade na implantação do sistema.

1. Introdução

Este manual tem como objetivo fornecer instruções detalhadas e passo a passo para a instalação, configuração e execução do Sistema de Almoxarifado desenvolvido em Python, utilizando a biblioteca PyQt6 para a interface gráfica e MySQL para o gerenciamento do banco de dados. O sistema foi projetado para auxiliar na gestão de empréstimos e devoluções de componentes em um ambiente educacional, agora com funcionalidades aprimoradas e uma interface de usuário mais flexível.

1.1. Visão Geral do Sistema

O Sistema de Almoxarifado é uma aplicação desktop que permite o controle de itens (componentes) emprestados a alunos, professores e funcionários. Suas principais funcionalidades incluem:

- **Pesquisa de Alunos:** Permite buscar alunos por matrícula, nome ou turma, exibindo seus detalhes e histórico de empréstimos.
- **Pesquisa de Componentes:** Possibilita a busca de componentes por ID ou nome, facilitando a localização de itens para empréstimo.
- **Registro de Empréstimos:** Funcionalidade para registrar a saída de componentes, associando-os a um aluno específico e registrando a data do empréstimo.
- **Registro de Devoluções:** Permite marcar componentes como devolvidos, atualizando o status do empréstimo no sistema.
- **Alternância de Tema (Dark/Light Mode):** Permite ao usuário alternar entre um tema claro e um tema escuro para a interface do aplicativo, melhorando a experiência visual.

- **Navegação por Turmas:** Funcionalidade para filtrar alunos por turmas específicas, incluindo uma categoria para "EX-Alunos".

O sistema interage com um banco de dados MySQL para armazenar informações sobre alunos, componentes e registros de empréstimos, garantindo a persistência e integridade dos dados.

1.3. Requisitos do Sistema

Para a correta execução do Sistema de Almoxarifado, os seguintes requisitos de hardware e software devem ser atendidos:

Hardware:

- Processador: Intel Core i3 ou equivalente (mínimo)
- Memória RAM: 4 GB (mínimo), 8 GB (recomendado)
- Espaço em Disco: 500 MB de espaço livre para o software e banco de dados

Software:

- **Sistema Operacional:** Windows 10/11, macOS (versões recentes) ou distribuições Linux (Ubuntu, Fedora, etc.)
- **Python:** Versão 3.x (recomendado 3.8 ou superior)
- **VSCode:** Visual Studio Code (última versão estável)
- **MySQL Server:** Versão 8.0 ou superior
- **Dependências Python:**
  - `mysql-connector-python`
  - `PyQt6`
  - `pytz`

Certifique-se de que todos os softwares listados estejam instalados e configurados corretamente antes de prosseguir com a instalação do sistema.

3. Configuração e Execução do Código

Esta seção guiará você através dos passos para configurar o código do Sistema de Almoxarifado e executá-lo em seu ambiente de desenvolvimento, bem como a criação de um executável. As atualizações no código trouxeram uma interface mais moderna e funcionalidades adicionais, como o modo escuro e a navegação por turmas.

3.1. Clonando o Repositório (Opcional, para GitHub)

Se o seu código estiver hospedado em um repositório Git (como o GitHub), você pode cloná-lo para o seu ambiente local. Caso contrário, você pode simplesmente copiar os arquivos do projeto para uma pasta de sua preferência.

1. **Instale o Git (se ainda não tiver):**
    - **Windows:** Baixe o instalador em https://git-scm.com/download/win e siga as instruções.
    - **macOS:** Instale via Homebrew: `brew install git`.
    - **Linux (Ubuntu/Debian):** `sudo apt install git`.
2. **Abra o Terminal ou Prompt de Comando** e navegue até o diretório onde você deseja clonar o projeto. Por exemplo: `bash cd C:\Users\SeuUsuario\Documentos\Projetos` ou `bash cd ~/Documentos/Projetos`
3. **Clone o repositório** usando o comando `git clone`, substituindo `URL_DO_SEU_REPOSITORIO` pela URL real do seu repositório Git: `bash git clone URL_DO_SEU_REPOSITORIO` Isso criará uma nova pasta com o nome do seu repositório contendo todos os arquivos do projeto.
4. **Navegue para a pasta do projeto clonado:** `bash cd nome_do_seu_repositorio`

Se você não estiver usando um repositório Git, simplesmente crie uma pasta para o projeto e copie todos os arquivos do código Python e o `default.png` para dentro dela.

3.2. Configurando as Credenciais do Banco de Dados no Código

O seu código Python contém as credenciais de conexão com o banco de dados MySQL. É crucial que estas credenciais correspondam às que você configurou na seção `2.4.2. Criação do Banco de Dados e Usuário`. Você precisará editar o arquivo Python principal para atualizar essas informações.

1. Abra o arquivo Python principal do seu projeto (o código que você me forneceu) no VSCode.
2. Localize as seguintes linhas no início do arquivo:

   ```
   python DB_NAME = "almox" DB_USER = "root" DB_PASSWORD = "manulauro5308" DB_HOST = "localhost" DB_PORT = 3306
   ```

3. **Altere os valores** de `DB_USER` e `DB_PASSWORD` para as credenciais do usuário `almox_user` que você criou para o banco de dados `almox`. Se você alterou o nome do banco de dados, `DB_NAME`, ou o host/porta, `DB_HOST` / `DB_PORT`, também os ajuste aqui.

   Exemplo:

   Se você criou o usuário `almox_user` com a senha `minhasenhaforte`:

   ```
   python DB_NAME = "almox" DB_USER = "almox_user" DB_PASSWORD = "minhasenhaforte" DB_HOST = "localhost" DB_PORT = 3306
   ```

4. Salve o arquivo (`Ctrl+S` ou `Cmd+S`).

É fundamental que essas credenciais estejam corretas para que o aplicativo consiga se conectar ao banco de dados e funcionar adequadamente.

3.3. Executando o Aplicativo no VSCode

Com todas as dependências instaladas e as credenciais do banco de dados configuradas, você está pronto para executar o aplicativo diretamente do VSCode.

1. **Abra a pasta do projeto no VSCode:**
   - No VSCode, vá em `File > Open Folder...` (Arquivo > Abrir Pasta...) e selecione a pasta onde você clonou ou copiou os arquivos do projeto.

2. **Abra o arquivo Python principal:**
    - No painel "Explorer" (Explorador) à esquerda, clique no arquivo Python principal (o que contém a classe `AlmoxarifadoApp` e o bloco `if __name__ == "__main__":`).
3. **Execute o arquivo:**
    - Com o arquivo Python aberto, você pode executá-lo de algumas maneiras:
        - **Botão "Run" (Executar):** No canto superior direito do editor, clique no ícone de "Play" (um triângulo verde) ou "Run Python File" (Executar Arquivo Python).
        - **Terminal Integrado:** Abra o Terminal Integrado do VSCode (`Terminal > New Terminal` ou `Ctrl+Shift+``). Certifique-se de que você está no diretório raiz do seu projeto. Em seguida, execute o comando: `bash python seu_arquivo_principal.py` (Substitua `seu_arquivo_principal.py` pelo nome real do seu arquivo Python, por exemplo, `almoxarifado_app.py`).
4. O aplicativo do Sistema de Almoxarifado deverá ser iniciado em uma nova janela, apresentando uma interface dividida em dois painéis (esquerdo e direito) e uma barra de ferramentas superior. Se ocorrerem erros, verifique o terminal do VSCode para mensagens de erro e consulte a seção de Solução de Problemas deste manual.

3.4. Como Modificar Funções Existentes e a Interface do Usuário

Uma das grandes vantagens de ter o código-fonte é a capacidade de adaptá-lo às suas necessidades específicas. Esta seção detalha como você pode modificar as funções existentes e a interface do usuário (UI) do Sistema de Almoxarifado, utilizando o VSCode e compreendendo a estrutura do código PyQt6, que foi significativamente aprimorada na versão mais recente.

3.4.1. Entendendo a Estrutura do Código PyQt6 Aprimorada

O aplicativo é construído usando a biblioteca PyQt6, seguindo um padrão de programação orientada a objetos. A classe principal `AlmoxarifadoApp` herda de `QWidget` e é responsável por toda a interface e lógica do aplicativo. A UI é construída

programaticamente, com uma estrutura mais modular e visualmente organizada, incluindo novos componentes e um sistema de temas.

Novos e Principais Componentes da UI (Widgets):

Além dos widgets já mencionados, a versão atualizada do código introduz:

- `QScrollArea`: Permite que o conteúdo dentro dele seja rolado, útil para listas longas de empréstimos ou resultados de pesquisa.
- `QFrame`: Um widget que pode ser usado para agrupar outros widgets e aplicar estilos visuais, como bordas e cores de fundo. A classe `EmprestimoItem` agora herda de `QFrame` para um visual mais organizado.
- `QCheckBox`: Caixas de seleção, usadas na nova classe `EmprestimoItem` para selecionar empréstimos para devolução.
- `QSplitter`: Permite que o usuário redimensione os painéis de uma janela, dividindo a interface em seções ajustáveis (neste caso, painel esquerdo e direito).
- `QDialog`: Uma janela de diálogo independente, usada para a seleção de turmas (`TurmasDialog`) e para a visualização de ex-alunos (`ExAlunosDialog`).
- `QTableWidget`, `QTableWidgetItem`, `QHeaderView`, `QAbstractItemView`: Componentes para exibir dados em formato de tabela, com maior controle sobre a apresentação e interação.
- `QMenu`, `QMenuBar`, `QToolBar`: Elementos para criar menus de aplicação e barras de ferramentas, melhorando a navegação e acesso a funcionalidades.
- `QFont`, `QIcon`, `QAction`, `Qt`, `QSize`: Classes para controle de fontes, ícones, ações de menu/botão, constantes Qt e tamanhos de widgets, respectivamente.

Novas Classes para Modularidade:

- `EmprestimoItem(QFrame)`: Uma classe personalizada que representa visualmente um item de empréstimo na lista. Cada instância desta classe é um "card" que exibe informações do empréstimo e um checkbox para seleção. Isso torna a lista de empréstimos mais rica e interativa.
- `TurmasDialog(QDialog)`: Uma janela de diálogo para selecionar turmas, permitindo filtrar a pesquisa de alunos por grupos específicos.
- `ExAlunosDialog(QDialog)`: Uma janela de diálogo (atualmente em desenvolvimento no código) que pode ser usada para gerenciar ou visualizar

informações de ex-alunos.

Conexão de Sinais e Slots (Aprimorada):

O mecanismo de sinais e slots continua sendo fundamental. Agora, você verá conexões para novas ações, como a alternância de tema (`self.botao_dark_mode.clicked.connect(self.alternar_dark_mode)`) e a exibição de turmas (`acao_turmas.triggered.connect(self.mostrar_turmas)`).

3.4.2. Modificando a Interface (Layout e Widgets)

A estrutura da interface foi reorganizada com o uso de `QSplitter` para dividir a janela principal em painéis esquerdo e direito, e a introdução de uma `QToolBar` e `QMenuBar`.

Exemplo 1: Personalizar o Tema (Dark/Light Mode)

A nova funcionalidade de tema é controlada pelo método `aplicar_estilo` e pela variável `self.dark_mode`. Você pode modificar as cores e estilos definidos nas strings CSS dentro deste método para personalizar a aparência do aplicativo nos modos claro e escuro.

```
# ... dentro de aplicar_estilo
        if self.dark_mode:
            estilo = """
                QWidget {
                    background-color: #2D2D2D; # Cor de fundo do modo escuro
                    color: #E0E0E0; # Cor do texto no modo escuro
                }
                QGroupBox {
                    background-color: #3C3C3C;
                    # ... outras propriedades de estilo
                }
                # ... e assim por diante para outros widgets
            """
        else:
            estilo = """
                QWidget {
                    background-color: #F5F7FA; # Cor de fundo do modo claro
                    color: #333333; # Cor do texto no modo claro
                }
                QGroupBox {
                    background-color: #FFFFFF;
                    # ... outras propriedades de estilo
                }
                # ... e assim por diante para outros widgets
            """
        self.setStyleSheet(estilo)
```

Exemplo 2: Adicionar um novo botão na barra de ferramentas

Para adicionar um novo botão na `QToolBar`:

1. **Localize a criação da `toolbar` no `__init__` da `AlmoxarifadoApp`:**

   ```python

# ... dentro de init

toolbar = QToolBar("Ferramentas")

# ... (botões existentes)

self.botao_dark_mode = QPushButton("🌙 Dark Mode")
self.botao_dark_mode.clicked.connect(self.alternar_dark_mode)
toolbar.addWidget(self.botao_dark_mode)

botao_turmas = QPushButton("Turmas")
botao_turmas.clicked.connect(self.mostrar_turmas)
toolbar.addWidget(botao_turmas)

# Adicione seu novo botão aqui

self.novo_botao = QPushButton("Novo Botão")
self.novo_botao.clicked.connect(self.sua_nova_funcao) # Conecte a uma nova função
toolbar.addWidget(self.novo_botao)

# ...

   ```

2. **Crie a função `sua_nova_funcao` na classe `AlmoxarifadoApp`:**

```python

# ... dentro da classe AlmoxarifadoApp

def sua_nova_funcao(self): QMessageBox.information(self, "Novo Botão", "Você clicou no novo botão!") ```

**Exemplo 3: Modificar a aparência dos itens de empréstimo**

A classe `EmprestimoItem` controla a aparência de cada item de empréstimo. Você pode modificar o método `__init__` desta classe para alterar o layout, as fontes, as cores ou adicionar novos widgets a cada item.

```
... dentro de EmprestimoItem.__init__
        # ...
        # 🔥 Definindo a cor dos textos igual do modo claro
        self.setStyleSheet("""
            QFrame {
                border: 2px solid #3498DB; # Exemplo: mudar a cor da borda
                border-radius: 8px; # Exemplo: aumentar o arredondamento
                background-color: #EBF5FB; # Exemplo: mudar a cor de fundo
            }
            QLabel {
                color: #2C3E50; # Exemplo: mudar a cor do texto
            }
            QCheckBox {
                color: #2C3E50;
            }
            QFrame:hover {
                background-color: #D6EAF8; # Exemplo: mudar a cor de hover
            }
        """)
```

### 3.4.3. Modificando a Lógica (Funções)

As funções do aplicativo são os métodos da classe `AlmoxarifadoApp` e das novas classes `EmprestimoItem`, `TurmasDialog` e `ExAlunosDialog`.

**Exemplo 1: Adicionar uma nova turma na `TurmasDialog`**

Para adicionar uma nova opção de turma na janela de seleção de turmas:

1. **Localize a classe `TurmasDialog` e o método `__init__`:**

```python

# ... dentro de TurmasDialog.init

```
turmas = [
    "4111", "4211", "4311", "4411", "4511",
    "4112", "4212", "4312", "4412",
    "4123", "4223", "4323", "4423",
    "4124", "4224", "4324", "4424",
    "EX-Alunos",
    "Nova Turma" # <--- Adicione sua nova turma aqui
]
```

...

```

#### Exemplo 2: Aprimorar a pesquisa de alunos por turma

O método `pesquisar_por_turma` já lida com a pesquisa por turma e a exceção para "EX-Alunos". Você pode estender essa lógica para incluir outras condições ou tratamentos especiais para turmas específicas.

```
# ... dentro de pesquisar_por_turma
            if turma == "EX-Alunos":
                cursor.execute(
                    "SELECT matricula, nome FROM alunos"
)
            elif turma == "Turma Especial": # <--- Exemplo de nova condição
                cursor.execute(
                    "SELECT matricula, nome FROM alunos WHERE turma = %s AND
status = 'especial'",
                    (turma,)
)
            else:
                cursor.execute(
                    "SELECT matricula, nome FROM alunos WHERE turma = %s",
                    (turma,)
)
...
```

#### 3.4.4. Dicas para Modificação

- **Use o VSCode para Navegar:** O VSCode possui excelentes ferramentas para navegar pelo código. Use "Go to Definition" ( `F12` ) para pular para a definição de
```

uma função ou variável, e "Find All References" (`Shift+F12`) para ver onde uma função ou variável é usada.

- **Comentários:** Adicione comentários ao seu código para explicar suas modificações e a lógica por trás delas. Isso ajuda você e outros a entenderem o código no futuro.
- **Testes:** Após cada modificação, teste o aplicativo para garantir que suas alterações funcionaram como esperado e não introduziram novos erros.
- **Controle de Versão:** Se você estiver usando Git (como sugerido na seção 3.1), faça commits frequentes de suas alterações. Isso permite que você reverta para versões anteriores se algo der errado.
- **Documentação PyQt6:** Para modificações mais avançadas, consulte a documentação oficial do PyQt6. Ela é uma fonte rica de informações sobre todos os widgets e funcionalidades disponíveis: https://www.riverbankcomputing.com/static/Docs/PyQt6/

Ao seguir estas diretrizes, você poderá personalizar e estender o Sistema de Almoxarifado para atender às necessidades específicas da Fundação Escola Técnica Liberato Salzano Vieira da Cunha.

4.5. Alternância de Tema (Dark/Light Mode)

O sistema de almoxarifado oferece a flexibilidade de alternar entre um tema claro e um tema escuro, proporcionando uma experiência visual mais agradável e adaptável às preferências do usuário ou às condições de iluminação do ambiente. Esta funcionalidade é especialmente útil para reduzir a fadiga ocular durante o uso prolongado.

Como Alternar o Tema:

1. **Localize o botão "Light Mode" ou "Dark Mode":** Na barra de ferramentas superior da janela principal do aplicativo, você encontrará um botão que permite alternar o tema. O texto do botão indicará o tema para o qual ele alternará (por exemplo, se o aplicativo estiver no modo claro, o botão dirá "🌙 Dark Mode").

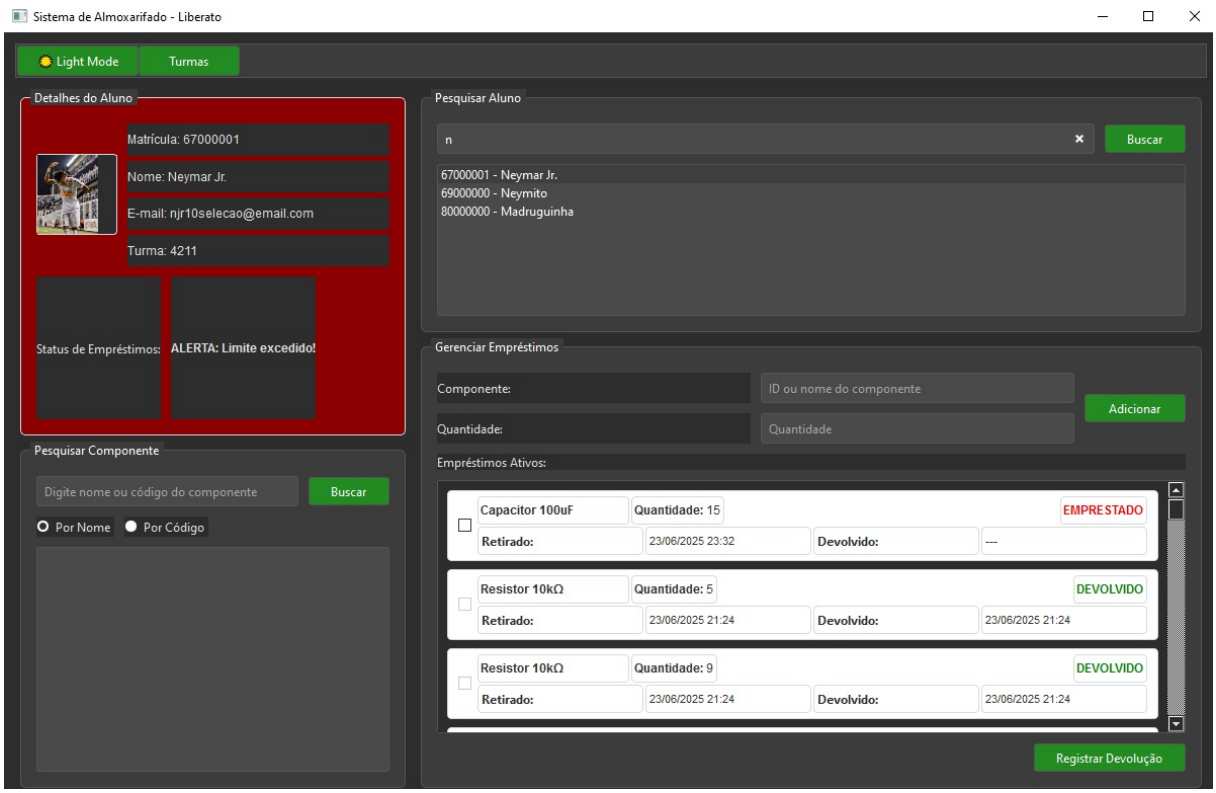

Figura 4: Aplicativo no Modo Escuro

2. **Clique no botão:** Ao clicar no botão, a interface do aplicativo mudará instantaneamente para o tema oposto. Todas as cores de fundo, texto, bordas e outros elementos visuais serão ajustados para o novo tema.

Esta funcionalidade é controlada pelo método `alternar_dark_mode` e `aplicar_estilo` no código, permitindo personalizações futuras nas cores e estilos dos temas.

2.4.5. Gerenciando a Coluna `email` na Tabela `alunos`

Para garantir que a tabela `alunos` contenha a coluna `email` e que esta seja preenchida automaticamente ou de forma controlada, siga as instruções abaixo. Isso é útil para padronizar os endereços de e-mail dos alunos, especialmente para fins de comunicação interna ou integração com outros sistemas.

1. Verifique a Estrutura Atual da Tabela (Opcional)

Antes de realizar qualquer alteração, é uma boa prática verificar a estrutura atual da tabela `alunos` para confirmar a existência ou não da coluna `email`.

```
DESCRIBE alunos;
```

Ou, para visualizar os primeiros registros e o conteúdo atual da tabela:

```
SELECT * FROM alunos LIMIT 5;
```

2. Adicione a Coluna `email` (se ainda não existir)

Se a coluna `email` ainda não estiver presente na sua tabela `alunos`, você pode adicioná-la usando o seguinte comando SQL. A cláusula `AFTER matricula` é opcional e define a posição da nova coluna na tabela.

```
ALTER TABLE alunos
ADD COLUMN email VARCHAR(100) AFTER matricula; -- Opcional: define a posição da coluna
```

3. Atualize os Emails Existentes

Após adicionar a coluna, você pode preencher os endereços de e-mail para os registros já existentes na tabela `alunos`. Este comando gera um e-mail no formato `matricula@liberato.com.br` para cada aluno.

```
UPDATE alunos
SET email = CONCAT(matricula, '@liberato.com.br');
```

4. (Opcional) Crie um TRIGGER para Autopreencher Novos Registros

Para automatizar o preenchimento da coluna `email` para novos alunos inseridos na tabela, você pode criar um `TRIGGER`. Este `TRIGGER` será executado `BEFORE INSERT` (antes de cada inserção) na tabela `alunos`, definindo o valor da coluna `email` para o novo registro.

```
DELIMITER //
CREATE TRIGGER gerar_email_aluno
BEFORE INSERT ON alunos
FOR EACH ROW
BEGIN
    SET NEW.email = CONCAT(NEW.matricula, '@liberato.com.br');
END//
DELIMITER ;
```

5. (Opcional) Use uma Coluna Gerada (MySQL 5.7+)

Para versões do MySQL 5.7 ou superiores, você pode usar uma coluna gerada. Esta abordagem calcula o valor da coluna `email` automaticamente com base na

`matricula` e armazena o resultado fisicamente (`STORED`). Isso elimina a necessidade de `TRIGGERS` ou `UPDATES` manuais para novos registros.

```
ALTER TABLE alunos
ADD COLUMN email VARCHAR(100)
GENERATED ALWAYS AS (CONCAT(matricula, '@liberato.com.br')) STORED;
```

6. Valide os Resultados

Após realizar as alterações, verifique se a coluna `email` foi preenchida corretamente para alguns registros:

```
SELECT matricula, email FROM alunos LIMIT 10;
```

Saída esperada:

```
+-----------+-------------------------+
| matricula | email                   |
+-----------+-------------------------+
| 21000264  | 21000264@liberato.com.br |
| 21000265  | 21000265@liberato.com.br |
+-----------+-------------------------+
```

Problemas Comuns e Soluções:

- **Erro de sintaxe no `CONCAT`:**
  - Verifique se a coluna `matricula` é do tipo `VARCHAR` / `CHAR`. Se for `INT`, faça a conversão explícita: `sql SET NEW.email = CONCAT(CAST(NEW.matricula AS CHAR), '@liberato.com.br');`
- **Permissões negadas:**
  - Certifique-se de executar os comandos SQL com um usuário que tenha privilégios `ALTER` e `UPDATE` na tabela `alunos`.
- **Tabela muito grande:**
  - Se a tabela `alunos` contiver milhões de registros, o comando `UPDATE` pode ser demorado. Recomenda-se executá-lo em horários de baixo uso do banco de dados para evitar impacto no desempenho.