FIAP - Faculdade de Informática e Administração Paulista

Fase 3: Cap 1 - Etapas de uma Máquina Agrícola

DeepThinker's

👩‍🎓 Integrantes Grupo 11:

- André Pessoa Gaidzakian - RM567877
- Erick Prados Pereira - RM566833
- Guilherme Ferreira Santos - RM568523
- Viviane de Castro - RM567367

👩‍🏫 Professores:

Tutor(a)

- Sabrina Otoni

Coordenador(a)

- André Godoi Chiovato

📜 Descrição

Link do GitHub

1. Introdução

O agronegócio brasileiro é um dos setores mais dinâmicos e estratégicos da economia nacional, representando cerca de 27% do PIB (IBGE, 2024). A modernização do campo, impulsionada por tecnologias digitais, sensores e automação, permite ganhos de produtividade, sustentabilidade e eficiência. Neste contexto, a FarmTech Solutions, startup fictícia, desenvolveu um sistema de monitoramento e automação agrícola, integrando sensores ambientais, lógica de decisão e banco de dados Oracle para armazenamento e análise dos dados coletados.
Este relatório detalha o processo de importação dos dados coletados na Fase 2 para o Oracle SQL Developer, explorando as etapas, boas práticas e evidências do funcionamento do sistema.

2. Objetivo

Demonstrar, de forma prática e documentada, a importação, consulta e manipulação dos dados agrícolas simulados (Fase 2) em um banco de dados Oracle, utilizando o Oracle SQL Developer, conforme orientações do PBL do curso de Inteligência Artificial.

3. Materiais Utilizados

- **Base de dados:** dados_agro.xlsx (simulada a partir de dados plausíveis do agro brasileiro, conforme CONAB, IBGE, Embrapa, etc.)
- **Ferramenta de banco de dados:** Oracle SQL Developer
- **Documentação e códigos:** Repositório GitHub do grupo
- **Sistema de sensores:** ESP32, sensores DHT22, botões NPK, LDR (pH), módulo relé (detalhado na Fase 2)
- **Códigos de integração:** C/C++ (ESP32) e Python (API clima)

4. Vídeo do projeto

🗀 Estrutura de pastas

Dentre os arquivos e pastas presentes na raiz do projeto, definem-se:

- **assets**: aqui estão os arquivos relacionados a elementos não-estruturados deste repositório, como imagens.
- **config**: Posicione aqui arquivos de configuração que são usados para definir parâmetros e ajustes do projeto.
- **document**: aqui estão todos os documentos do projeto que as atividades poderão pedir. Na subpasta "other", adicione documentos complementares e menos importantes.
- **scripts**: Posicione aqui scripts auxiliares para tarefas específicas do seu projeto. Exemplo: deploy, migrações de banco de dados, backups.
- **src**: Todo o código fonte criado para o desenvolvimento do projeto ao longo das 7 fases.
- **README.md**: arquivo que serve como guia e explicação geral sobre o projeto (o mesmo que você está lendo agora).

🔧 Como executar o código

Pré-requisitos

Antes de executar o projeto, certifique-se de ter instalado:

- **Python 3.8 ou superior** - Download Python
- **Oracle Database** - Oracle XE 21c ou superior (Download Oracle XE)
- **Oracle SQL Developer** (opcional, para gerenciar o banco) - Download SQL Developer
- **Git** - Para clonar o repositório

Bibliotecas Python utilizadas

- `oracledb` - Conexão com Oracle Database

- `python-dotenv` - Gerenciamento de variáveis de ambiente
- `tabulate` - Formatação de tabelas no terminal

Passo 1: Clonar o Repositório

```
git clone https://github.com/seu-usuario/seu-repositorio.git
cd seu-repositorio/Fase6Modificada
```

Passo 2: Configurar Ambiente Virtual

Crie e ative um ambiente virtual Python para isolar as dependências:

```
python -m venv .venv
.\.venv\Scripts\Activate.ps1
```

> **Nota:** Se houver erro de política de execução, execute: `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`

Passo 3: Instalar Dependências

```
pip install -r requirements.txt
```

Passo 4: Configurar o Banco de Dados Oracle

1. **Copie o arquivo de exemplo de configuração:**

   ```
   copy config\.env.example config\.env
   ```

2. **Edite o arquivo `config/.env` com suas credenciais Oracle:**

   ```
   DB_USER=seu_usuario
   DB_PASSWORD=sua_senha
   DB_DSN=localhost:1521/XE
   ```

   > **Importante:** O arquivo `.env` contém credenciais sensíveis e **NÃO** deve ser commitado no Git (já está no `.gitignore`).

3. **Teste a conexão com o banco:**

   ```
   python -m scripts.teste_conexao
   ```

Passo 5: Executar o Sistema

Executar o programa principal (menu interativo):

```
python -m src.main
```

O sistema apresentará um menu com as seguintes opções:

- **1. Registrar colheita** - Cadastra nova colheita no banco
- **2. Gerar relatório** - Gera relatórios em JSON, TXT e CSV
- **3. Sair** - Encerra o sistema

Funcionalidades Principais

📊 Registrar Colheita

Permite registrar dados de colheita com informações de:

- Tipo de colheita (manual ou mecânica)
- Produtividade estimada (t/ha)
- Produtividade real (t/ha)
- Valor por tonelada (R$)

Observação importante: O sistema **apenas registra perdas**. Se a produtividade real for maior que a estimada (ganho), o sistema informará o usuário e **não salvará** o registro, pois o foco é monitorar perdas agrícolas. Verifique as mensagens de confirmação:

- [OK] = Dados salvos com sucesso
- [ERRO] = Problema na conexão/banco de dados

📈 Gerar Relatórios

Gera relatórios automáticos em três formatos:

- **JSON** (`document/relatorio.json`) - Dados estruturados
- **TXT** (`document/relatorio.txt`) - Relatório completo com sumário executivo
- **CSV** (`document/relatorio.csv`) - Planilha para análise

Estrutura de Execução por Fase

Fase 1 - Coleta de Dados com Sensores

- Sistema físico com ESP32 e sensores (DHT22, LDR, botões NPK)
- Código Arduino/C++ para leitura de sensores

Fase 2 - Integração com Oracle Database

- Conexão com Oracle Database via `oracledb`
- CRUD completo de colheitas
- Geração de relatórios analíticos
- Sistema de menu interativo

Comandos Úteis

Verificar versão do Python:

```
python --version
```

Listar pacotes instalados:

```
pip list
```

Atualizar pip:

```
python -m pip install --upgrade pip
```

Desativar ambiente virtual:

```
deactivate
```

Solução de Problemas

Erro ao ativar ambiente virtual:

- Execute: `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`

Erro de conexão Oracle:

- Verifique se o Oracle Database está rodando
- Confirme as credenciais no arquivo `.env`
- Teste com: `python -m scripts.teste_conexao`

Módulo não encontrado:

- Certifique-se de que o ambiente virtual está ativado
- Reinstale as dependências: `pip install -r requirements.txt`

Histórico de lançamentos

- 2.0.0 - 02/11/2025 (Versão Atual)
    - **Refatoração completa baseada no feedback do professor**
    - Migração de `cx_Oracle` para `oracledb` (biblioteca moderna)
    - Implementação de variáveis de ambiente (.env) para credenciais Oracle
    - Tratamento robusto de exceções (try/except) em todas operações de I/O e banco
    - Sistema de commit/rollback para garantir integridade transacional
    - Funções modulares com docstrings, parâmetros e retornos bem definidos
    - Operações CRUD completas em memória com listas/dicionários
    - Formatação tabular de consultas (biblioteca `tabulate`)
    - Geração de relatórios em três formatos: JSON, TXT e CSV
    - Estrutura modular organizada (src/, config/, document/, scripts/)
    - Scripts de teste de conexão e utilitários
    - Documentação completa de execução no README
    - Evidências de execução com exemplos de relatórios
- 1.0.0 - 12/10/2025 (Primeira Entrega)
    - Versão inicial do sistema de monitoramento agrícola
    - Integração básica com Oracle Database
    - Problema bem definido e solução coerente para o agronegócio
    - Sistema de colheita com registro de perdas
    - Geração de relatórios em TXT e JSON
    - Organização modular do código
    - Persistência básica de dados
    - Menu interativo para operações do sistema
    - **Recebeu ponto extra por equipe com 4 integrantes**

Licença