Sistema de Monitoramento de Perdas na Colheita de Cana-de-Açúcar

> Projeto da Fase 2 - Capítulo 6: Python e Além

Integrantes do GRUPO 13

Nome	RM	Email
Viviane de Castro	RM567367	vivi.topproducer@gmail.com
GUILHERME FERREIRA SANTOS	RM566833	gifisi.channel@gmail.com
André Pessoa Gaidzakian	RM567877	andrepgaidzak@gmail.com
Erick Prados Pereira	RM566833	erick.prados.pereira@gmail.com

1. O Problema do Agronegócio: Perdas na Colheita de Cana

O agronegócio é um pilar da economia brasileira, representando mais de 20% do PIB nacional. Dentro desse vasto setor, a cultura da cana-de-açúcar é uma das mais relevantes, com o Brasil sendo o maior produtor mundial. Na safra 2018/2019, por exemplo, a produção atingiu cerca de 620 milhões de toneladas.

No entanto, um dos maiores desafios enfrentados pelos produtores é a **perda durante o processo de colheita**. Estudos da SOCICANA indicam que as perdas na colheita mecanizada podem chegar a **15% da produção**. Esse percentual, aparentemente pequeno, representa um prejuízo financeiro gigantesco, equivalente à capacidade de aquisição de novas usinas de etanol.

As causas dessas perdas são variadas, incluindo velocidade inadequada das colhedoras, falhas na regulagem dos equipamentos e compactação do solo. Diante desse cenário, o monitoramento preciso e a análise de dados tornam-se ferramentas essenciais para a tomada de decisões estratégicas que visam mitigar esses prejuízos e aumentar a eficiência operacional.

2. Nossa Solução: Uma Ferramenta de Análise e Gestão

Para endereçar este problema, desenvolvemos um sistema em Python que permite ao produtor rural registrar, monitorar e analisar as perdas ocorridas em cada evento de colheita de cana-de-açúcar. A solução visa transformar dados brutos em insights claros, auxiliando na gestão e na melhoria contínua dos processos agrícolas.

Funcionalidades Principais

1. **Registro Detalhado de Colheitas:**
    - O usuário pode registrar dados essenciais de cada colheita, como o tipo (manual ou mecânica), a produtividade estimada (em t/ha) e a produtividade real obtida.
    - O sistema solicita o valor da tonelada para calcular o impacto financeiro de forma precisa.

2. **Cálculo Automático de Perdas e Prejuízos:**
    - Com base nos dados de produtividade, o sistema calcula automaticamente o **percentual de perda** e o **prejuízo financeiro** correspondente em Reais (R$).
    - A data e a hora de cada registro são salvas para garantir a rastreabilidade e análise temporal.
3. **Lógica Inovadora de Análise de Ganhos:**
    - O sistema foi projetado para focar no problema central: as perdas. Caso a produtividade real seja **maior** que a estimada, a ferramenta parabeniza o usuário pelo **ganho de produtividade** e não armazena o registro, mantendo a base de dados focada exclusivamente em eventos de prejuízo que necessitam de análise.
4. **Consistência e Validação de Dados:**
    - Para garantir a integridade dos dados, todas as entradas numéricas (produtividade, valor) são validadas para aceitar apenas números, evitando erros que poderiam comprometer as análises futuras.
5. **Geração de Relatórios Estruturados:**
    - A ferramenta gera dois tipos de relatórios para diferentes finalidades:
        - `relatorio.txt`: Um relatório de fácil leitura, formatado para o entendimento humano. Ele apresenta cada registro de perda de forma detalhada e finaliza com um **sumário executivo** contendo o total de registros, a média de perda percentual e, mais importante, o **prejuízo total acumulado**.
        - `relatorio.json`: Um arquivo com dados estruturados, ideal para a integração com outros sistemas, dashboards de Business Intelligence (BI) ou análises de dados mais avançadas.

Arquitetura e Estrutura do Projeto

O código está organizado em módulos com responsabilidades específicas, seguindo princípios SOLID:

```
.
├── main.py           # Ponto de entrada, menu principal
├── banco.py          # Conexão e operações Oracle
├── colheita.py       # Lógica de registro de colheitas
├── relatorios.py     # Geração de relatórios (TXT/JSON/CSV)
├── utils.py          # Funções utilitárias
├── requirements.txt  # Dependências do projeto
└── .env.example      # Modelo para configuração
```

Detalhamento dos Módulos

`main.py`

- Ponto de entrada do programa
- Menu interativo com opções:

    1. Registrar nova colheita
    2. Gerar relatório
    3. Sair
- Inicializa estruturas do banco

`banco.py`

- Gerencia conexões Oracle usando `oracledb`
- Funções principais:
    - `conectar_oracle()`: Estabelece conexão usando variáveis de ambiente
    - `criar_tabela()`: Verifica/cria estrutura inicial
    - `inserir_dados(dados)`: Insere novo registro com transação
    - `listar_dados()`: Recupera todos os registros
- Implementa tratamento de erros e transações ACID

`colheita.py`

- Interface para registro de colheitas
- Coleta e validação de dados:
    - Tipo de colheita (manual/mecanizada)
    - Produção estimada e real (t/ha)
    - Valor por tonelada (R$)
- Cálculos automáticos:
    - Percentual de perda
    - Prejuízo financeiro (R$)

`relatorios.py`

- Geração de relatórios em múltiplos formatos
- Saídas:
    1. Console: Tabela formatada via `tabulate`
    2. TXT: Relatório detalhado com sumário
    3. JSON: Dados estruturados para programas
    4. CSV: Planilha para Excel/análises
- Cálculos estatísticos:
    - Médias de perda
    - Prejuízos totais
    - Contagens por tipo

`utils.py`

- Funções utilitárias compartilhadas
- `validar_float(mensagem)`: Validação de entrada numérica
- `print_table(headers, rows)`: Formatação tabular de dados

Estrutura do Banco de Dados

```
CREATE TABLE colheitas (
    id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    tipo VARCHAR2(20),           -- 'manual' ou 'mecanica'
    prod_estimada FLOAT,         -- em toneladas/hectare
    prod_real FLOAT,             -- em toneladas/hectare
    valor_tonelada FLOAT,        -- em R$
    perda_percentual FLOAT,      -- calculado
    prejuizo FLOAT,              -- calculado em R$
    data VARCHAR2(50)            -- timestamp do registro
)
```

3. Como Executar o Projeto

Pré-requisitos

- Python 3.8 ou superior
- Acesso a um banco de dados Oracle
- Git (opcional, para clonar o repositório)

Instalação e Configuração

1. Clone ou baixe este repositório
2. Crie e ative um ambiente virtual:

```
python -m venv .venv
.\.venv\Scripts\Activate.ps1  # Windows/PowerShell
source .venv/bin/activate     # Linux/macOS
```

3. Instale as dependências:

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

4. Configure as credenciais do banco:

```
copy .env.example .env
# Edite o arquivo .env com suas credenciais:
# DB_USER=seu_usuario
# DB_PASSWORD=sua_senha
# DB_DSN=host:porta/servico
```

> **Nota**: O projeto usa `oracledb` em modo "thin", que não requer Oracle Instant Client instalado.
> Se precisar do modo "thick", instale o Instant Client e configure o PATH conforme documentação do oracledb.

Execução e Uso

1. Certifique-se de que o ambiente virtual está ativo:

```
.\.venv\Scripts\Activate.ps1  # Windows/PowerShell
```

2. Execute o programa:

```
python main.py
```

3. Use o menu interativo:

```
=== Sistema de Monitoramento de Perdas na Colheita de Cana ===
1 - Registrar nova colheita
2 - Gerar relatório
3 - Sair
Escolha uma opção:
```

Formatos de Relatório

O sistema gera relatórios em quatro formatos diferentes:

1. Console (Tempo Real)

```
=== REGISTROS DE COLHEITA ===
+----+---------+--------+-------+-------+------+----------+
| ID | Tipo    | Prod.E | Prod.R| Valor | Perda| Prejuízo|
+====+=========+========+=======+=======+======+==========+
|  1 | Manual  | 85.50  | 80.20 |   150 | 6.2% |    795  |
+----+---------+--------+-------+-------+------+----------+

Total de registros: 1
Média de perda: 6.20%
Prejuízo total: R$ 795,00
```

2. Relatório TXT

- Cabeçalho com data/hora
- Registros individuais detalhados
- Sumário executivo com:
    - Total de registros
    - Contagem por tipo de colheita
    - Média de perda percentual

- Prejuízo total acumulado

3. Relatório JSON

```
[
  {
    "ID": 1,
    "Tipo de Colheita": "Manual",
    "Prod. Estimada (t/ha)": "85.50",
    "Prod. Real (t/ha)": "80.20",
    "Valor por Tonelada (R$)": "R$ 150,00",
    "Perda Percentual (%)": "6.20%",
    "Prejuízo (R$)": "R$ 795,00",
    "Data e Hora": "2025-10-26 14:30"
  }
]
```

4. Planilha CSV

- Formato tabular compatível com Excel
- Inclui cabeçalho descritivo
- Valores formatados para análise
- Ideal para importação em ferramentas BI

Aspectos Técnicos

Segurança e Boas Práticas

1. **Gestão de Credenciais**
   - Uso de variáveis de ambiente via `python-dotenv`
   - Arquivo `.env` no `.gitignore`
   - Exemplo de configuração em `.env.example`
2. **Tratamento de Erros**
   - Try/except em operações I/O
   - Validação de entrada de dados
   - Mensagens de erro amigáveis
   - Rollback automático em falhas
3. **Transações de Banco**
   - Operações ACID garantidas
   - Commit explícito após sucesso
   - Rollback em caso de erro
   - Conexões sempre fechadas
4. **Código e Arquitetura**

    - Módulos com responsabilidade única
    - Funções documentadas (docstrings)
    - Validações consistentes
    - Nomes descritivos de variáveis

🛠️ Stack Tecnológica

- **Python 3.8+**
    - Tipagem moderna
    - f-strings
    - Context managers
- **Banco de Dados**
    - Oracle via `oracledb`
    - Modo "thin" (sem Instant Client)
    - Pooling de conexões
    - Prepared statements
- **Bibliotecas**
    - `oracledb`: Conexão Oracle moderna
    - `python-dotenv`: Configurações seguras
    - `tabulate`: Formatação de dados

📈 Roadmap e Melhorias Futuras

1. **Interface**
    - Interface web via Flask/FastAPI
    - Dashboard interativo
    - Gráficos de tendência
2. **Banco de Dados**
    - Migrations para versionamento
    - Índices otimizados
    - Particionamento por data
3. **Análise**
    - Machine Learning para previsões
    - Alertas automáticos
    - Integração com BI
4. **Deployment**
    - Containerização (Docker)
    - CI/CD pipeline
    - Monitoramento

Conclusão

Este projeto demonstra a aplicação prática de conceitos modernos de desenvolvimento Python no contexto do agronegócio. Além de resolver um problema real de monitoramento de perdas na colheita, implementa boas práticas de engenharia de software como:

- Segurança no tratamento de credenciais
- Modularização e responsabilidade única
- Tratamento robusto de erros
- Documentação clara e abrangente
- Múltiplos formatos de saída para análise

O sistema fornece uma base sólida para que produtores de cana-de-açúcar otimizem suas operações através de dados estruturados e análises objetivas, alinhando-se às tendências de agricultura digital e decisões baseadas em dados.