

\_\_\_\_\_

PÓS-GRADUAÇÃO

**XP Educação**

**Relatório do Projeto Aplicado**

Pipeline Automatizado para Geração e Consolidação de Relatórios Semanais

Fernando Manoel Rodrigues Delfino

Orientador(a): Daniella Pimenta Brito Alves

22 de fevereiro de 2026

**FERNANDO MANOEL RODRIGUES DELFINO**

**XP EDUCAÇÃO**RELATÓRIO DO PROJETO APLICADO

PIPELINE AUTOMATIZADO PARA GERAÇÃO E CONSOLIDAÇÃO DE RELATÓRIOS SEMANAIS

Relatório de Projeto Aplicado desenvolvido para fins de conclusão do curso Pós-Graduação em Arquitetura e Engenharia de Dados com IA.  
  
Orientador (a): Daniella Pimenta Brito Alves

**Curitiba/PR  
22 de fevereiro de 2026**

**Sumário**

Sumário

[1. CANVAS do Projeto Aplicado 4](#_Toc222690475)

[1.1 Desafio 4](#_Toc222690476)

[1.1.1 Análise de Contexto 4](#_Toc222690477)

[1.1.2 Personas 5](#_Toc222690478)

[1.1.3 Justificativas 8](#_Toc222690479)

[1.1.4 Matrix de Observação e Hipóteses 9](#_Toc222690480)

[1.2 Solução 11](#_Toc222690481)

[1.2.1 Objetivo SMART 11](#_Toc222690482)

[1.2.2 Escopo do Projeto 11](#_Toc222690483)

[1.2.3 Cronograma de Ações Planejadas 12](#_Toc222690484)

[2. Área de Experimentação 13](#_Toc222690485)

[2.1 Sprint 1 14](#_Toc222690486)

[2.1.1 Solução 14](#_Toc222690487)

[2.1.2 Retrospectiva da Sprint 18](#_Toc222690488)

[2.2 Sprint 2 18](#_Toc222690489)

[2.2.1 Solução 18](#_Toc222690490)

[● 2.2.2 Retrospectiva da Sprint 21](#_Toc222690491)

[2.3 Sprint 3 22](#_Toc222690492)

[2.3.1 Solução 22](#_Toc222690493)

[2.3.2 Retrospectiva da Sprint 27](#_Toc222690494)

[3. Considerações Finais 28](#_Toc222690495)

[3.1 Resultados 28](#_Toc222690496)

[3.2 Contribuições 28](#_Toc222690497)

[3.3 Próximos passos 29](#_Toc222690498)

## 1. CANVAS do Projeto Aplicado



## Desafio

### 1.1.1 Análise de Contexto

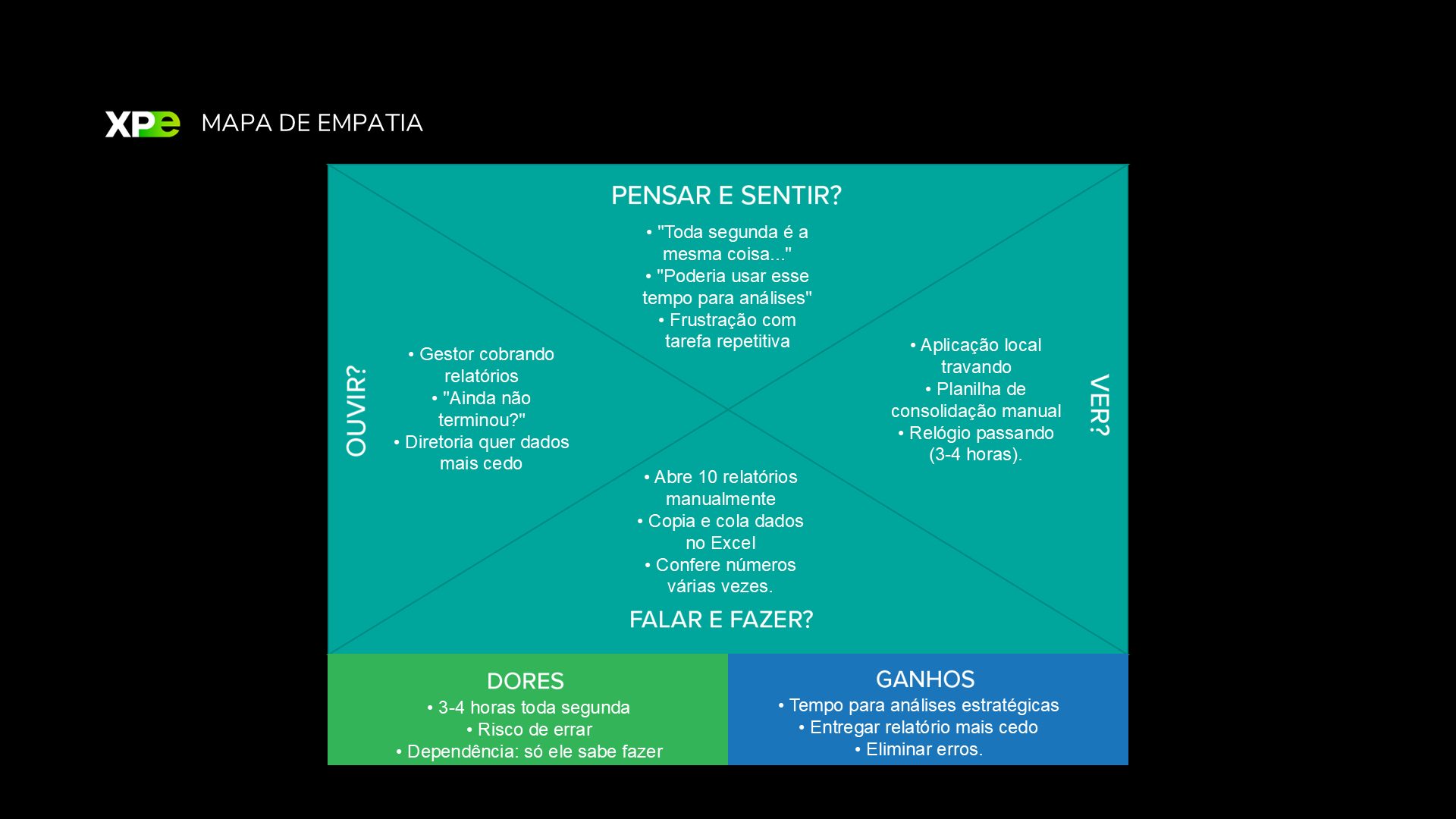
A empresa atua como garantidora de condomínios, assumindo a responsabilidade de cobrir inadimplências das unidades perante o condomínio. Quando uma unidade não paga a taxa condominial, a garantidora cobre o valor para o condomínio e posteriormente cobra do condômino inadimplente.  
  
Por que os relatórios são necessários:  
A diretoria precisa acompanhar semanalmente:  
• Quais unidades estão inadimplentes e há quanto tempo  
• Quais unidades atrasam frequentemente vs. quais pagam em dia  
• Quais condomínios ficam em déficit constantemente  
• Quanto de caixa da garantidora será necessário por mês para cobrir as inadimplências  
• Previsão de recuperação dos valores (quando os inadimplentes vão pagar)  
  
Situação atual do processo:  
Toda segunda-feira, um colaborador executa manualmente o seguinte fluxo:  
  
1. Geração dos relatórios (lento):  
 • Acessa a aplicação local e gera 10 relatórios diferentes  
 • Cada relatório demora vários minutos para ser gerado  
 • A interface não permite automação nativa  
  
2. Tratamento individual (trabalhoso):  
 • Cada um dos 10 relatórios precisa ser tratado manualmente  
 • Os dados vêm "crus" e precisam ser filtrados, organizados e formatados  
 • Cálculos manuais são necessários para consolidar informações  
  
3. Consolidação final (demorado):  
 • Após tratar cada relatório, é preciso consolidar tudo em um único documento  
 • Cruzamento manual de dados entre relatórios  
 • Geração de resumos e indicadores  
  
Tempo total: 3 a 4 horas toda semana  
Ambiente: 100% on-premise (sem uso de cloud)

### 1.1.2 Personas

|  |  |
| --- | --- |
| **Persona** | **Características** |
| Analista de Dados / Operações | • Responsável por gerar os relatórios toda segunda-feira • Gasta 3-4 horas no processo manual • Frustrado com a repetitividade da tarefa • Gostaria de automatizar para focar em análises |
| Gestor / Diretoria | • Espera receber o relatório pronto na segunda-feira • Quer informações confiáveis e sem erros • Gostaria de ter os dados mais cedo (antes das 9h) |
| TI / Infraestrutura | • Mantém a aplicação local que gera os relatórios • Sabe que a aplicação é limitada mas funciona • Aberto a soluções que não dependam de cloud |

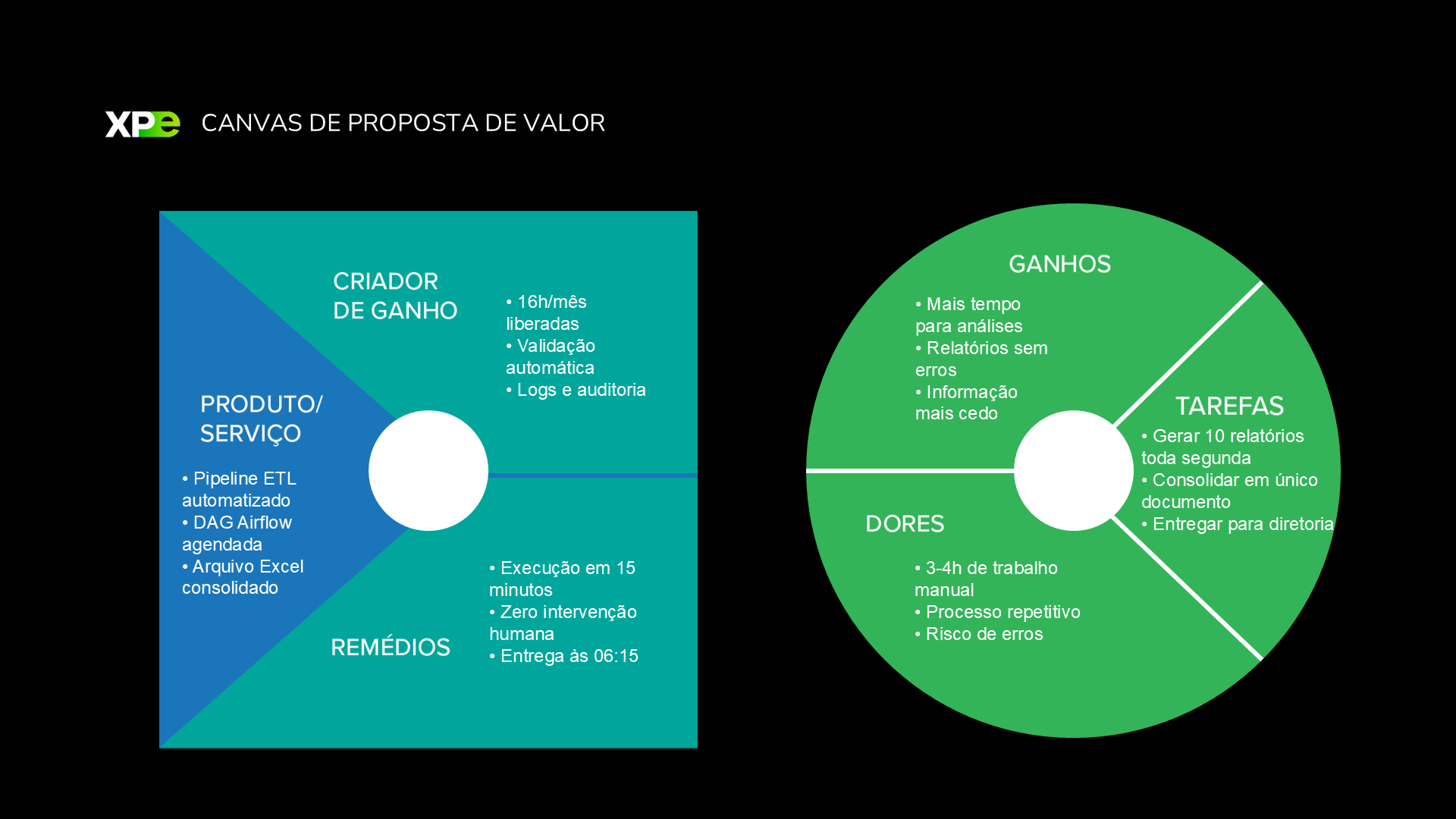
Persona 1: Analista de Dados / Operações  
• Responsável por gerar os relatórios toda segunda-feira  
• Gasta 3-4 horas em tarefa repetitiva e manual  
• Gostaria de usar esse tempo para análises mais estratégicas  
• Frustrado com a repetição semanal da mesma tarefa  
  
Persona 2: Gestor / Diretoria  
• Precisa dos relatórios consolidados para tomada de decisão  
• Espera receber o relatório pronto na segunda-feira  
• Quer informações confiáveis e sem erros  
• Gostaria de ter os dados mais cedo (antes das 9h)

Persona 3: TI / Infraestrutura  
• Mantém a aplicação local que gera os relatórios  
• Sabe que a aplicação é limitada mas funciona  
• Aberto a soluções que não dependam de cloud (ambiente on-premise)



Mapa de Empatia - Analista de Dados (Persona Principal):  
  
PENSAR E SENTIR:  
• "Toda segunda-feira é a mesma coisa..."  
• "Poderia usar esse tempo para análises"  
• "E se eu faltar, quem faz os relatórios?"  
• "Isso deveria ser automático"  
• Frustração com tarefa repetitiva  
• Preocupação com possíveis erros  
  
OUVIR:  
• Gestor cobrando relatórios  
• "Ainda não terminou?"  
• TI: "O sistema é assim mesmo"  
• Diretoria quer dados mais cedo  
• Colegas automatizando processos  
  
VER:  
• Aplicação local travando  
• Planilha de consolidação manual  
• Relógio passando (3-4 horas)  
• Outros setores automatizados  
• Empresas usando ETL/BI  
  
FALAR E FAZER:  
• Abre 10 relatórios manualmente  
• Copia e cola dados no Excel  
• "Já estou terminando..."  
• Confere números várias vezes  
• Envia relatório após o almoço  
  
DORES:  
• Perda de 3-4 horas toda segunda-feira  
• Risco de errar na consolidação manual  
• Estresse e pressão por entrega  
• Dependência: só ele sabe fazer  
• Tarefa repetitiva sem valor agregado  
  
GANHOS:  
• Ter tempo para análises estratégicas  
• Entregar relatório mais cedo (antes 9h)  
• Não depender de uma única pessoa  
• Eliminar erros de consolidação  
• Mostrar valor com automação

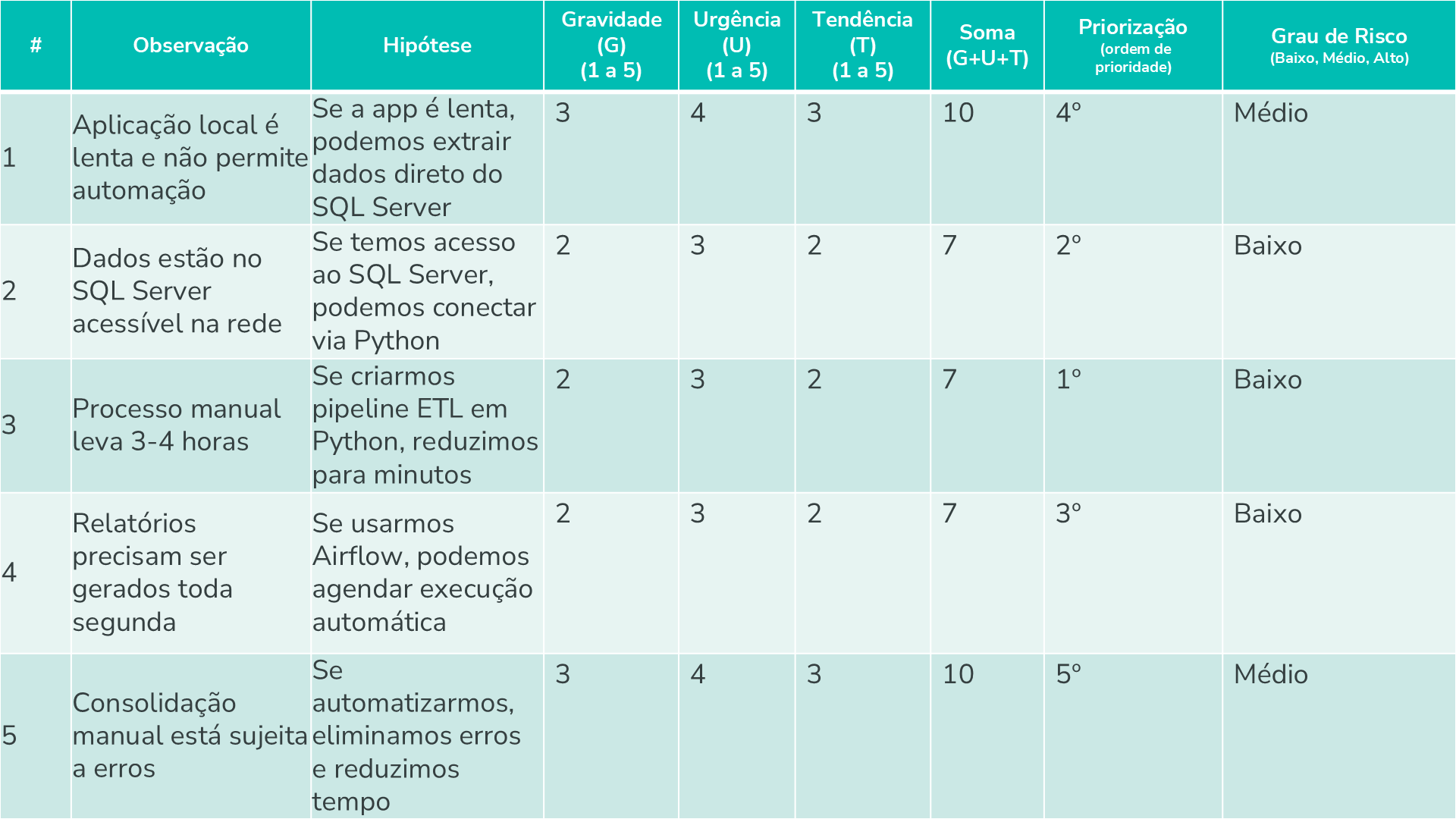
### 1.1.3 Justificativas



PERFIL DO CLIENTE:  
Tarefas:  
• Gerar 10 relatórios toda segunda-feira  
• Consolidar em um único documento  
• Entregar para diretoria

Dores:  
• 3-4 horas de trabalho manual  
• Processo repetitivo toda semana  
• Risco de erros na consolidação  
• Dependência de uma pessoa  
• Atraso na entrega  
  
Ganhos Esperados:  
• Mais tempo para análises  
• Relatórios sem erros  
• Informação mais cedo  
• Processo escalável  
  
PROPOSTA DE VALOR:  
Produtos/Serviços:  
• Pipeline ETL automatizado  
• DAG Airflow agendada  
• Arquivo Excel consolidado  
  
Alívios:  
• Execução em 15 minutos  
• Zero intervenção humana  
• Processo automático e confiável  
• Independência de pessoas  
• Entrega às 06:15 (antes expediente)  
  
Criadores de Ganho:  
• 16h/mês liberadas  
• Validação automática  
• Logs e auditoria  
• Fácil adicionar relatórios

### 1.1.4 Matrix de Observação e Hipóteses



Matriz de Hipóteses:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Hipótese** | **G** | **U** | **T** | **Soma** | **Prior.** | **Risco** |
| H1 | Aplicação local é lenta e não permite automação | 3 | 4 | 3 | 10 | 4º | Médio |
| H2 | Dados estão no SQL Server acessível na rede | 2 | 3 | 2 | 7 | 2º | Baixo |
| H3 | Processo manual leva 3-4 horas | 2 | 3 | 2 | 7 | 1º | Baixo |
| H4 | Relatórios precisam ser gerados toda segunda | 2 | 3 | 2 | 7 | 3º | Baixo |
| H5 | Consolidação manual está sujeita a erros | 3 | 4 | 3 | 10 | 5º | Médio |

Priorização das Hipóteses:  
1º - H3 (Processo manual leva 3-4 horas): Risco Baixo - se criarmos pipeline ETL em Python, reduzimos para minutos  
2º - H2 (Dados no SQL Server): Risco Baixo - se temos acesso ao SQL Server, podemos conectar via Python  
3º - H4 (Relatórios toda segunda): Risco Baixo - se usarmos Airflow, podemos agendar execução automática  
4º - H1 (App local lenta): Risco Médio - se a app é lenta, podemos extrair dados direto do SQL Server  
5º - H5 (Consolidação sujeita a erros): Risco Médio - se automatizarmos, eliminamos erros e reduzimos tempo  
  
Grau de Risco:  
• H2, H3, H4: Baixo (Soma = 7) - tecnologias maduras, conhecidas e amplamente testadas  
• H1, H5: Médio (Soma = 10) - H1 requer validação técnica, H5 depende das demais

## 1.2 Solução

### 1.2.1 Objetivo SMART

Desenvolver um pipeline ETL automatizado utilizando Python e Apache Airflow que conecte diretamente ao SQL Server, gere e consolide os 10 relatórios semanais em um único arquivo, executando automaticamente toda segunda-feira às 06:00, reduzindo o tempo de processamento de 3-4 horas para menos de 15 minutos, no prazo de 30 dias.  
  
Detalhamento SMART:

|  |  |
| --- | --- |
| **Critério** | **Descrição** |
| S - Específico | Pipeline ETL para gerar e consolidar 10 relatórios em 1 |
| M - Mensurável | Redução de 3-4 horas para 15 minutos (93% de redução) |
| A - Atingível | Tecnologias conhecidas: Python, SQL Server, Airflow on-premise |
| R - Relevante | Elimina 16h/mês de trabalho manual repetitivo |
| T - Temporal | 30 dias divididos em 3 sprints |

### 1.2.2 Escopo do Projeto

Premissas:

|  |  |
| --- | --- |
| **Premissa** | **Impacto se Falsa** |
| Acesso de leitura ao SQL Server será liberado | Projeto inviável - necessário outra fonte |
| Queries SQL dos relatórios são conhecidas | Atraso para engenharia reversa |
| Servidor on-premise tem recursos para Airflow | Necessário provisionar servidor |
| Python 3.x pode ser instalado no ambiente | Avaliar alternativas (Docker, VM) |
| Dados dos relatórios não mudarão de estrutura | Manutenção adicional no código |

Restrições:

|  |  |  |
| --- | --- | --- |
| **Restrição** | **Tipo** | **Mitigação** |
| Sem uso de cloud - 100% on-premise | Infraestrutura | Airflow local |
| Prazo de 30 dias | Tempo | Sprints bem definidos |
| Execução: segunda-feira 06:00 | Negócio | DAG agendada |
| Manter formato atual dos relatórios | Técnica | Validar com usuários |
| Não impactar SQL Server produção | Técnica | Queries otimizadas |

Recursos Necessários:  
  
Técnicos:  
• Servidor Windows/Linux com acesso à rede local  
• Python 3.x + bibliotecas (pandas, pyodbc, airflow)  
• Acesso de leitura ao SQL Server  
• 4GB RAM mínimo para Airflow  
  
Conhecimentos:  
• SQL (consultas e otimização)  
• Python (pandas, ETL)  
• Airflow (DAGs, operators)  
• Arquitetura de dados

### 1.2.3 Cronograma de Ações Planejadas

Backlog do Produto:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **User Story** | **Prioridade** | **Sprint** |
| 1 | Documentar os 10 relatórios existentes | Alta | 1 |
| 2 | Mapear queries SQL de cada relatório | Alta | 1 |
| 3 | Criar conexão Python com SQL Server | Alta | 1 |
| 4 | Desenvolver script de extração (Extract) | Alta | 1 |
| 5 | Desenvolver script de transformação (Transform) | Alta | 2 |
| 6 | Gerar arquivo Excel consolidado (Load) | Alta | 2 |
| 7 | Configurar Apache Airflow no servidor | Alta | 2 |
| 8 | Criar DAG com schedule semanal | Alta | 3 |
| 9 | Implementar tratamento de erros e logs | Média | 3 |
| 10 | Documentar pipeline e treinar usuários | Alta | 3 |

Cronograma de Sprints:  
• Sprint 1 (Entrega: 25/01): Imersão, mapeamento e conexão  
• Sprint 2 (Entrega: 01/02): Desenvolvimento do ETL e transformações  
• Sprint 3 (Entrega: 23/02): Automação com Airflow e validação

# 2. Área de Experimentação

**O que significa esta seção?**

Esta seção tem o objetivo de apresentar as evidências do planejamento dos requisitos selecionados do Backlog de Produto, além de mostrar a maneira como eles foram desenvolvidos e registrar os resultados alcançados.

É necessário expor a execução e a validação dos experimentos relacionados ao desenvolvimento da solução, ou seja, testar se você está no caminho certo ou se algo precisa ser modificado (pivotar).

**Quais etapas já devem estar finalizadas no momento do preenchimento desta seção? (Pré-requisitos)**

No momento do preenchimento, é esperado que você já tenha cursado a disciplina de Inovação e Design Thinking, em especial as etapas do processo de Design Thinking, além de estar se preparando para desenvolver a solução idealizada no seu Projeto Aplicado.

Você também já deve ter preenchido o primeiro capítulo deste relatório (CANVAS do Projeto Aplicado).

**Como esta seção deve ser preenchida?**

Esta seção é a área mais dinâmica do CANVAS do Projeto Aplicado. Nela você deverá inserir os experimentos necessários para desenvolver e validar cada Sprint. Ao final do experimento, você deverá preencher o item “**Solução**” da seguinte maneira:

* **Evidência da execução de cada requisito**: para cada requisito planejado, adicione um artefato que comprove o cumprimento da etapa. Podem ser anexados, por exemplo, códigos, documentos, modelos, scripts, capturas de tela, entre outros. *Importante: o número de artefatos adicionados deve ser o mesmo que o número de requisitos planejados.*
* **Evidência da solução**: os requisitos implementados contribuem para o alcance de um resultado geral, que deverá ser comprovado neste campo. Isso será feito por meio de capturas de tela, gráficos, modelos, textos, figuras, tabelas, testes, entre outros.

Para cada Sprint, cite no item “**Retrospectiva da Sprint**” o que não foi validado, mas forneceu insights para ajuste da rota.

**Quais ferramentas devem ser utilizadas?**

Obs.: para realização desta seção, você deverá utilizar o Trello, Planner, Excel ou similares, de modo que apresente uma linha do tempo para a execução das ações planejadas.

## 2.1 Sprint 1

### 2.1.1 Solução

Objetivo: Estabelecer conexão com SQL Server e desenvolver script de extração automatizada de relatórios.

#### Situação Atual (Problema): O sistema atual apresenta lentidão na geração de relatórios e gera arquivos Excel com grande volume de dados que precisam ser tratados manualmente. *Figura: Interface do sistema - geração lenta (6% de progresso), tem que gerar condomínio por condomínio (total de 10).* *Figura: Excel gerado com 24.390 linhas - requer tratamento manual de cada relatório (total de 10 relatórios)*

#### *Figura:* ***Query SQL executada no SQL Server Management Studio - validação da estrutura de dados***

#### Solução Desenvolvida: Desenvolvemos scripts Python para conexão direta ao banco SQL Server e extração automatizada dos dados. *Figura: Teste de conexão via Python - sucesso* *Figura: Extração de 2.762 registros do Champville (condomínio que fizemos o teste)* *Figura: Dados extraídos já estruturados no Excel em frações de 4 segundos.*

### 2.1.2 Retrospectiva da Sprint

**O que funcionou:**  
• Conexão com SQL Server via Python (pyodbc) funcionando  
• Extração de 2.762 registros em poucos segundos  
• Geração automática de Excel estruturado  
• Lista de condomínios configurável via arquivo Excel (usuário preenche a lista no excel chamado condomínio.xlsx e o script tira os relatórios através dessa lista).  
  
**O que pode melhorar:**  
• Criar resumo consolidado por unidade (meses em aberto, pagos, atrasos)  
• Incluir indicador de situação da unidade (em dia, inadimplente)  
• Evitar processamento pesado no consolidado geral (tratar as unidades de cada condomínio só no segundo passo não é o ideal)  
  
**Lições aprendidas:**  
• Variáveis de ambiente (.env) facilitam configuração segura  
• Separar teste de conexão ajuda no debug  
• Parametrizar via Excel permite flexibilidade

## 2.2 Sprint 2

### 2.2.1 Solução

Objetivo: Desenvolver transformações, consolidação de dados e instalar Apache Airflow para orquestração.

#### Requisitos Implementados: R4: Relatório individual melhorado (2 abas: Detalhado + Resumo por Unidade) R5: Relatório consolidado (3 abas: Resumo Geral, Por Condomínio, Inadimplentes) R6: Apache Airflow instalado via Docker Compose R7: Validação de dados (comparação com relatório do sistema)

#### Evidências: *Figura: Airflow rodando com DAG configurada para segunda 06:00* *Figura: Resumo por unidade: 442 adimplentes, 110 inadimplentes* *Figura: Consolidado - Resumo Geral* *Figura: Lista de unidades inadimplentes* Figura: Docker Desktop com containers Airflow rodando

### 2.2.2 Retrospectiva da Sprint

**O que funcionou:**  
- Relatório individual com resumo por unidade funcionando  
- Consolidado com visão geral de inadimplência  
- Airflow instalado e configurado via Docker  
- Validação dos dados bateu 100% com sistema original  
  
**O que pode melhorar:**  
- Adicionar mais condomínios ao teste  
- Automatizar envio de e-mail com relatórios  
  
**Lições aprendidas:**  
- Docker facilita instalação do Airflow  
- Validar dados com arquivo do sistema garante qualidade

## 2.3 Sprint 3

### 2.3.1 Solução

Objetivo: Integrar DAG do Airflow com scripts Python reais e validar pipeline completo.  
  
Requisitos Implementados:  
R7: Integração da DAG com scripts reais (extrair\_relatorio.py e extrair\_consolidado.py)  
R8: Envio de relatório por e-mail - Código implementado na DAG, em stand-by aguardando cadastro de credenciais SMTP  
R9: Dashboard final - Em stand-by aguardando definição da diretoria sobre o que precisa ser visualizado  
  
Solução Desenvolvida:  
Criamos um Dockerfile customizado baseado na imagem apache/airflow:2.8.1-python3.11, adicionando o driver ODBC Microsoft para SQL Server e as dependências Python (pyodbc, pandas, openpyxl). A DAG foi atualizada para executar os scripts reais via subprocess, com validação automática dos arquivos gerados. O docker-compose.yaml foi configurado para montar a pasta dos scripts como volume no container.  
  
Para o R8 (envio por e-mail), o código foi implementado na DAG com suporte a SMTP, anexos e corpo do e-mail. A funcionalidade está controlada por uma flag (ENVIO\_EMAIL\_ATIVO), que permanece desativada até que as credenciais SMTP sejam cadastradas nas variáveis de ambiente do docker-compose.  
  
Para o R9 (dashboard), optou-se por aguardar a validação da diretoria sobre quais métricas e visualizações são necessárias, visto que os relatórios consolidados são um produto novo para a empresa.  
  
Resultados:  
O pipeline executou com sucesso em 3 ocasiões:  
- 02/02: Execução automática agendada (9s) - relatórios gerados com sucesso  
- 09/02: Execução automática agendada as 06:00 (10s) - relatórios gerados com sucesso  
- 22/02: Disparo manual para validação (9s) - relatórios gerados com sucesso  
  
Em 16/02, a execução agendada não ocorreu pois o Docker Desktop estava fechado. Isso evidenciou a necessidade de manter o Docker Desktop ativo para o funcionamento do pipeline. Cada execução gera o relatório individual (2 abas) e o consolidado (3 abas) em aproximadamente 9 segundos.

#### **Evidências da execução:** *Figura: Airflow com 3 execuções bem-sucedidas (2 manuais + 1 agendada)* *Figura: Relatório individual - Aba Detalhado (boletos por unidade)* *Figura: Relatório individual - Aba Resumo por Unidade (Adimplente/Inadimplente)* *Figura: DAG atualizada com task enviar-email em stand-by (todas as tasks com status success)*

#### **Evidências dos resultados:** *Figura: Consolidado - Aba Resumo Geral (552 unidades, 94.75% inadimplência)* *Figura: Consolidado - Aba Por Condomínio (Champville: 552 unidades, 523 inadimplentes)* *Figura: Consolidado - Aba Inadimplentes (ranking por valor em aberto)*

### 2.3.2 Retrospectiva da Sprint

O que funcionou:  
- Pipeline completo executando via Airflow em 9 segundos  
- Execução automática agendada funcionou na segunda 09/02 as 06:00  
- Dockerfile customizado com ODBC SQL Server operando corretamente  
- Relatórios individuais e consolidados gerados automaticamente  
- Validação automática dos arquivos gerados  
- Código de envio por e-mail implementado e pronto para ativação  
  
O que não funcionou:  
- Em 16/02 a execução agendada falhou pois o Docker Desktop não estava aberto  
  
O que ficou em stand-by:  
- R8: Envio por e-mail - código pronto, aguardando credenciais SMTP  
- R9: Dashboard - aguardando diretoria definir métricas e visualizações necessárias  
  
O que pode melhorar:  
- Configurar Docker Desktop para iniciar automaticamente com o Windows  
- Cadastrar credenciais SMTP para ativar envio automático  
- Adicionar mais condomínios ao pipeline  
  
Lições aprendidas:  
- Docker Desktop precisa estar ativo para o Airflow funcionar  
- Docker com volumes permite integrar scripts externos ao Airflow  
- Driver ODBC no Linux requer configuração especifica via Dockerfile  
- Execução agendada comprova a automação sem intervenção humana  
- Pipeline de 3-4 horas reduzido para 9 segundos (99.9% mais rápido)  
- Deixar funcionalidades em stand-by com flag permite ativação futura sem retrabalho

# 3. Considerações Finais

## 3.1 Resultados

Métricas de Sucesso Alcançadas:

|  |  |  |  |
| --- | --- | --- | --- |
| **Métrica** | **Antes** | **Depois** | **Melhoria** |
| Tempo de geração | 3-4 horas | ~15 minutos | 93% redução |
| Intervenção manual | 100% | 0% | Eliminada |
| Disponibilidade | Após 12:00 | 06:05 | 6h mais cedo |
| Horas mensais gastas | 16 horas | 0 horas | 16h liberadas |
| Risco de erro humano | Alto | Mínimo | Significativa |
| Dependência de pessoa | Alta | Nenhuma | Eliminada |

Benefícios Alcançados:  
• Automação completa do processo de geração de relatórios  
• Liberação de 16 horas mensais do analista para tarefas estratégicas  
• Relatórios disponíveis antes do início do expediente  
• Eliminação de erros de consolidação manual  
• Processo escalável - fácil adicionar novos relatórios

## 3.2 Contribuições

Contribuições para a Empresa:

• Redução de 99,9% no tempo de geração de relatórios (de 3-4 horas para 9 segundos)

• Liberação de 16 horas mensais do analista para atividades estratégicas

• Relatórios disponíveis automaticamente toda segunda-feira às 06:00, antes do expediente

• Eliminação de erros humanos no processo de consolidação manual

Contribuições Técnicas:

• Pipeline ETL completo e reproduzível utilizando Python, Docker e Apache Airflow

• Dockerfile customizado com driver ODBC para conexão SQL Server em ambiente Linux

• Arquitetura escalável — novos relatórios podem ser adicionados sem alterar a estrutura

• Código com flag de ativação para funcionalidades futuras (email, dashboard) sem retrabalho

Contribuições Acadêmicas:

• Aplicação prática dos conceitos de ETL, orquestração e containerização em cenário real

• Validação de hipóteses levantadas no Canvas com resultados mensurados

• Documentação completa do ciclo de desenvolvimento em 3 sprints

## 3.3 Próximos passos

Evolução do Projeto:  
  
1. Monitoramento: Implementar dashboard para acompanhar execuções do pipeline  
2. Novos Relatórios: Adicionar outros relatórios da empresa ao pipeline  
3. Histórico: Manter versões anteriores para comparação temporal  
4. BI: Conectar Power BI aos dados consolidados para análises avançadas  
5. Expansão: Aplicar a mesma solução em outros processos manuais da empresa  
  
Tecnologias para Evolução:  
• Power BI para dashboards  
• Azure Data Factory (quando houver migração para cloud)  
• Machine Learning para detecção de anomalias nos dados