Soluções Cross Dados - R3 S3 2025

Seção: Ferramenta Única de Ocorrências - COPF - MVP

História: [Foundation][Dados] Carga da Base 360 e Base de Equipamentos no RDS da COPF

- **Visão de Produto:**

Acreditamos que, ao **carregar no RDS da COPF as bases complementares — Base 360 (IBS 360) e a Base de Equipamentos**, com estrutura padronizada e pronta para cruzamento com a base de ocorrências pendentes, **garantiremos a continuidade do MVP mesmo antes da democratização oficial dessas bases**, permitindo que as funcionalidades do produto sejam sustentadas com dados confiáveis, mesmo que ainda sob carga manual.

Saberemos que isso é verdade quando **as tabelas estiverem disponíveis no RDS da COPF, com os campos necessários para enriquecer as análises e alimentar as regras de negócio do produto**.

- **Descrição:**

Como time de dados da COPF, queremos **carregar e manter atualizadas no RDS as duas bases complementares fundamentais para o funcionamento do produto**:

- **Base 360**: Traz contexto sobre os pontos de atendimento do banco.
- **Base de Equipamentos**: Fornecida com apoio do time do Painel Operacional, é essencial para entender os equipamentos vinculados às ocorrências.

Essas duas bases permitirão o enriquecimento da base de ocorrências pendentes, viabilizando a priorização e análise com base em atributos complementares.

- **Dados Necessários:**

1. **Base 360**:
    - CD_PONTO
    - Nome do ponto (agência)
    - Estado (UF)
    - Município
    - DINEG
    - SUPT
    - Tipo do Ponto (ex: Agência, Espaço Itaú, PAB, PAE)
    - Categoria para o COPF:
        - Convencional (AG)
        - Convencional (PAB)
        - Terceirizada (Espaço Itaú)
        - Terceirizada (PAB)
        - Terceirizada (PAE)
        - Terceirizada (Phygital)
2. **Base de Equipamentos**:
    - Identificador único do equipamento
    - Nome do equipamento

- **Visão do Usuário:**

Usuários do COPF, incluindo **analistas de operação e gestores de produto**, precisam de dados enriquecidos sobre agências e equipamentos para:

- Priorizar ocorrências críticas com base no tipo de ponto e equipamento afetado.
- Diferenciar pontos convencionais e terceirizados, orientando ações de SLA e tratativas com fornecedores.
- Viabilizar análises estruturadas e visualizações agregadas nas telas operacionais.

- **Contexto / Narrativa de Negócio:**

Atualmente, essas bases **não estão democratizadas via catálogo**, o que impede o acesso automatizado e obriga o uso de versões manuais. Carregá-las no RDS da COPF **é uma solução provisória estratégica** que garante a continuidade operacional, testes de funcionalidades e validações de regras de negócio essenciais para o avanço do produto.

- **Premissas:**

1. Os arquivos mais recentes dessas bases estão disponíveis por extração manual junto aos times responsáveis (Base 360 e Painel Operacional).
2. A ingestão no RDS será feita com estrutura padronizada, mesmo que sob atualização manual.
3. Essas cargas são temporárias até a formalização da democratização e ingestão automatizada.

- **Regras de Negócio:**

1. As tabelas carregadas devem conter os campos mínimos necessários para cruzamento com a base de ocorrências.
2. Cada versão carregada deve conter uma coluna `data_base` informando sua referência temporal.
3. A periodicidade mínima de atualização deve ser definida até o processo ser automatizado.

- **Informações Técnicas:**
- Local de ingestão: **RDS da COPF (VO3)**.
- Tabelas:
  - `tbl_agências`
  - `tbl_equipamentos`
- Atualização via script Python com execução manual (inicialmente).

- **Tarefas:**

1. Solicitar e validar os arquivos atualizados com os times do IBS 360 e Painel Operacional.
2. Realizar a carga inicial e validar os dados inseridos.

3. Testar o cruzamento com a base de ocorrências.

- **Cenários para Teste e Homologação:**

1. As tabelas são populados com os campos esperados e tipagem adequada.
2. Os dados cruzam corretamente com a base de ocorrências (testar joins e filtros).
3. O volume e a integridade dos dados são mantidos durante a carga.
4. As colunas `data_base` estão corretamente preenchidas.

- **Impacto Esperado:**
- Continuidade das funcionalidades já construídas no MVP.
- Capacidade de análise enriquecida e orientada por contexto.
- Redução de riscos associados à falta de democratização oficial no curto prazo.

- **Conclusão:**
  - **Início:**
    - **Desejado:** R3 S3 2025
    - **Real:** R3 S3 2025
  - **Fim:**
    - **Desejado:** R3 S3 2025
    - **Real:** *(preencher ao fim da sprint)*
  - **Resultado:** *(preencher ao fim da sprint)*

História: [Delivery][Frontend] Desenvolvimento dos big numbers (cards) da tela de Home Page (dashboard) (Frontend)

- **Visão de Produto:**

Acreditamos que, ao **exibir Big Numbers (indicadores resumidos e de alto impacto visual) na tela inicial do COPF (dashboard)**, página visível apenas para a**nalistas operacionais e gestores (não estará visível para**

fornecedores), resultará em **uma visão imediata da situação geral das ocorrências e permitirá priorizações rápidas e orientadas por dados**.

Saberemos que isso é verdade quando **os usuários conseguirem, com apenas uma olhada na dashboard, entender a gravidade do cenário atual, como número de ocorrências críticas, agências VIPs afetadas e equipamentos inoperantes.**

- **Descrição:**

Como time de frontend da COPF, queremos **desenvolver e disponibilizar na tela inicial do produto os Big Numbers**, com destaque visual e navegabilidade, para que **os times operacionais possam ter uma leitura rápida e objetiva dos principais KPIs da ferramenta**.

Indicadores exibidos:

1. **Ocorrências pendentes**
2. **Ocorrências críticas**
3. **Agências afetadas** (com destaque para VIPs)
4. **Equipamentos inoperantes**

Cada indicador contará com:

- Valor absoluto atualizado a cada vez que o usuário clicar no botão de atualizar.
- Acesso a "Detalhes" com redirecionamento para as telas de listagem e filtros aplicados.

- **Visão do Usuário:**

Usuários operacionais e gestores precisam visualizar os principais indicadores de status logo ao acessar a ferramenta, para tomar decisões rápidas, priorizar ações e distribuir esforços da equipe com base na gravidade do cenário atual.

- **Contexto / Narrativa de Negócio:**

Antes da exibição desses indicadores, os times precisavam abrir várias telas ou consultar painéis externos para entender a situação da rede. Isso dificultava reações rápidas em contextos críticos. Os Big Numbers oferecem uma leitura executiva e operacional da situação em tempo real.

- **Premissas:**

1. Os indicadores são calculados com base em views ou queries já existentes.
2. Os dados utilizados são os mesmos da tela de ocorrências, garantindo consistência.
3. A navegação para "Detalhes" aplica filtros correspondentes ao indicador clicado.

- **Regras de Negócio:**

1. **Ocorrências pendentes**: Total de ocorrências abertas.
    a. Status da ocorrência: A iniciar + Em Andamento + Com impedimentos
2. **Ocorrências críticas**: Subconjunto das pendentes, marcadas como críticas.
    a. Status da ocorrência: A iniciar + Em Andamento + Com impedimentos
    b. Criticidade: Críticas
3. **Agências afetadas**: Contagem de agências com ao menos uma ocorrência pendente; VIPs devem aparecer em destaque.
4. **Equipamentos inoperantes**: Contagem de equipamentos com status de indisponibilidade operacional.
    a. Status da ocorrência: A iniciar + Em Andamento + Com impedimentos
    b. Status do equipamento: Inoperantes

- **Informações Técnicas:**
- Os valores devem ser renderizados com destaque visual (ex: fonte grande, ícones se aplicável).
- O componente de **“Detalhes”** deve permitir redirecionamento com filtros aplicados.
- Os dados devem ser atualizados de forma reativa com base no backend atual (streaming ou polling).

- **Tarefas:**

1. Mapear os dados necessários para cada indicador.
2. Construir os componentes de Big Numbers no front-end.
3. Criar a navegação para as páginas de detalhe com os filtros aplicados.

4. Realizar testes de exibição e performance.

- **Cenários para Teste e Homologação:**

1. A home carrega corretamente os quatro indicadores com dados atualizados.
2. Cada botão "Detalhes" leva à tela correta com os filtros esperados.
3. Layout responsivo e legível em diferentes resoluções.
4. Os dados exibidos batem com as queries de base.

- **Impacto Esperado:**
- Aumento da velocidade de análise e reação dos times operacionais.
- Maior clareza do status geral da operação logo na entrada da ferramenta.
- Redução de cliques e tempo para identificar prioridades.

- **Conclusão:**
  - **Início:**
    - **Desejado:** R3 S3 2025
    - **Real:** R3 S3 2025
  - **Fim:**
    - **Desejado:** R3 S3 2025
    - **Real:** *(preencher ao fim da sprint)*
  - **Resultado:** *(preencher ao fim da sprint)*

História: [Delivery][Backend] Desenvolvimento da Lambda de agregação para Big Numbers da Home Page (dashboard)

- **Visão de Produto:**

Acreditamos que, ao **desenvolver uma Lambda responsável por consolidar e entregar os indicadores que compõem os Big Numbers da Home (Dashboard) do COPF**, será possível **garantir atualização eficiente, performance e consistência dos dados que sustentam a experiência de leitura rápida e tomada de decisão por parte dos usuários**.

Saberemos que isso é verdade quando **os Big Numbers forem exibidos com informações atualizadas, consistentes com os dados operacionais, e com tempo de resposta adequado para uso no dia a dia operacional**.

- **Descrição:**

Como time de engenharia de dados/backend da COPF, queremos **implementar uma função Lambda em AWS que agregue e exponha os dados necessários para os indicadores principais da Home (Big Numbers)**, para que o frontend possa consumi-los de forma simples e performática.

A função deve:

- Ser acionada sob demanda (ex: clique no botão de atualizar).
- Realizar as agregações conforme as regras de negócio.
- Retornar um JSON contendo os valores dos quatro indicadores.
- Ser acessada via API Gateway com autenticação apropriada.

- **Visão do Usuário:**

Embora invisível para o usuário final, essa Lambda viabiliza que os dados dos Big Numbers estejam sempre consistentes com a base atual, com performance adequada e baixo custo de infraestrutura.

- **Contexto / Narrativa de Negócio:**

A Lambda centraliza o processo de agregação com regras de negócio bem definidas, garantindo dados confiáveis e encapsulados para o dashboard.

- **Premissas:**

1. As tabelas e views utilizadas já estão disponíveis no ambiente da VO3 - COPF.
2. A Lambda será protegida por autenticação e autorizada apenas ao front da COPF. Além de validar se quem está chamando, é da operação, pois somente a operação deve conseguir obter esses valores.
3. O deploy será realizado via pipeline automatizado.

- **Regras de Negócio (para cálculo dos indicadores):**

1. **Ocorrências pendentes**: Contar todas com status:

- `A iniciar`, `Em Andamento`, `Com impedimentos`

2. **Ocorrências críticas**: Subconjunto das pendentes com `criticidade = Crítica`.
3. **Agências afetadas**: Contagem de agências com ao menos uma ocorrência pendente. Indicar também quantas são VIPs.
4. **Equipamentos inoperantes**:
    - Ocorrências com status `A iniciar`, `Em Andamento`, `Com impedimentos`
    - E equipamentos com status = `Inoperante`

- **Informações Técnicas:**
- Linguagem sugerida: Python 3.11
- Infra: AWS Lambda + API Gateway
- Output (JSON):

```
{
  "ocorrencias_pendentes": 747,
  "ocorrencias_criticas": 254,
  "agencias_afetadas": {
    "total": 211,
    "vips": 41
  },
  "equipamentos_inoperantes": 326
}
```

- **Tarefas:**

1. Especificar as queries de agregação para cada indicador.
2. Implementar a Lambda com controle de erros e logging.
3. Realizar testes unitários e de performance.

- **Cenários para Teste e Homologação:**

1. Lambda retorna todos os indicadores com dados consistentes.
2. Manipulação de falhas (ex: erro na query, retorno vazio).
3. Integração validada com o front-end (mock ou ambiente real).

4. Logs gerados para cada execução.

- **Critérios de Aceite:**

☐ A Lambda está disponível em endpoint autenticado.

☐ O retorno segue o padrão de JSON acordado.

☐ Os dados retornados batem com as views de origem.

☐ O código está versionado e integrado no repositório da COPF.

- **Impacto Esperado:**
- Consolidação da lógica de negócio em um ponto único.
- Maior confiabilidade e consistência dos dados apresentados no dashboard.

- **Conclusão:**
  - **Início:**
    - **Desejado:** R3 S3 2025
    - **Real:** R3 S3 2025
  - **Fim:**
    - **Desejado:** R3 S3 2025
    - **Real:** *(preencher ao fim da sprint)*
  - **Resultado:** *(preencher ao fim da sprint)*

História: [Delivery][Frontend] Desenvolvimento dos gráficos da dashboard da Home Page (COPF)

- **Visão de Produto:**

Acreditamos que, ao **exibir gráficos de análise na Home Page (Dashboard) do COPF**, como **status de equipamentos**, **agências mais afetadas**, **aging de ocorrências**, **motivos mais frequentes** e **fornecedores e a relação previsão de atendimento e sla**, **os times operacionais poderão identificar rapidamente padrões, tendências e gargalos**, otimizando decisões e melhorando a eficiência da atuação.

Saberemos que isso é verdade quando **os usuários conseguirem tomar decisões ou priorizar ações com base apenas nos gráficos exibidos, sem**

precisar acessar outras telas ou relatórios externos.

- **Descrição:**

Como time de frontend da COPF, queremos **implementar os gráficos principais da Home Page**, com visual limpo, responsivo e de fácil leitura, para que os usuários tenham uma **visão analítica resumida** das ocorrências e equipamentos monitorados.

Gráficos a serem desenvolvidos:

1. **Status dos Equipamentos**
2. **Top 10 pontos por volume de ocorrências**
3. **Gráfico de Aging**
4. **Long Tail - Sintomas das Ocorrências**
    a. Sem o eixo duplo (gráfico de linha acumulada)
5. **Fornecedor x Previsão x SLA**

Cada gráfico deve:

- Ser reativo à atualização manual da página

- **Visão do Usuário:**

Usuários operacionais e gestores precisam **entender rapidamente os focos de atenção**, como agências mais impactadas, principais motivos de falhas e o tempo médio de resolução, de forma visual e acessível logo na Home (Dashboard).

- **Contexto / Narrativa de Negócio:**

Antes desses gráficos, as análises eram feitas manualmente a partir de planilhas ou painéis separados. Ao integrar os gráficos na Home Page do COPF, tornamos o monitoramento mais inteligente e proativo, especialmente em situações críticas.

- **Premissas:**

1. As bibliotecas gráficas utilizadas são compatíveis com o framework atual do COPF.
2. Os dados são consistentes com os já utilizados em outras telas da plataforma.

- **Regras de Negócio:**
  - **Status dos Equipamentos:**
    - Agrupar por tipo de equipamento.
    - Contabilizar status "operante" e "inoperante".
  1. **Top 10 pontos por volume de ocorrências**
     - Listar as 10 agências com maior volume de ocorrências pendentes.
     - Incluir marcador para VIPs.
  2. **Aging das Ocorrências:** Agrupar ocorrências abertas pelas faixas de dias:
     - 0 – 1 dia
     - 1 – 2 dias
     - 2 – 4 dias
     - ≥ 5 dias
  3. **Long Tail – Motivos:** Agrupar ocorrências por motivo/sintoma e ordenar por frequência decrescente.
  4. **Fornecedor x Previsão x SLA:**
     - Agrupar ocorrências por fornecedor.
     - Categorizar conforme:
       - SLA vencido
       - Previsão maior que SLA
       - Sem previsão

- **Tarefas:**

1. Implementar componentes de gráficos no frontend.
2. Integrar com a API de backend (Lambda).
3. Garantir layout responsivo e acessibilidade.
4. Realizar testes de usabilidade e performance.

- **Cenários para Teste e Homologação:**

1. Os gráficos carregam corretamente com os dados da API.
2. Os dados são atualizados ao clicar em “Atualizar”.
3. Os gráficos são responsivos em diferentes dispositivos.
4. O comportamento visual é consistente com os padrões da plataforma.

- **Impacto Esperado:**
- Redução do tempo de análise das equipes.
- Aumento da assertividade nas decisões operacionais.
- Proatividade no tratamento de falhas e gargalos.

- **Conclusão:**
- **Início:**
  - **Desejado:** R3 S3 2025
  - **Real:** R3 S3 2025
- **Fim:**
  - **Desejado:** R3 S3 2025
  - **Real:** *(preencher ao fim da sprint)*
- **Resultado:** *(preencher ao fim da sprint)*

História: [Delivery][Backend] Desenvolvimento da Lambda de agregação para os gráficos da Home Page (COPF)

- **Visão de Produto:**

Acreditamos que, ao desenvolver uma **função Lambda dedicada à agregação e disponibilização dos dados para os gráficos do dashboard do COPF**, entregaremos **rapidez, consistência e facilidade de manutenção**, permitindo que o frontend se concentre apenas na visualização.

Saberemos que isso é verdade quando **os gráficos da Home Page forem carregados com alta performance, refletindo dados consistentes e atualizados, sem depender de múltiplas queries dispersas ou lógicas fragmentadas.**

- **Descrição:**

Como time de backend da COPF, queremos **implementar uma Lambda que agregue e entregue os dados necessários para a renderização dos gráficos da Home Page**, consumida via API Gateway autenticada.

Essa Lambda será responsável por consolidar as regras de negócio dos seguintes gráficos:

1. **Status dos Equipamentos**
2. **Top 10 pontos por volume de ocorrências**
3. **Gráfico de Aging**
4. **Long Tail - Sintomas das Ocorrências**
    a. Sem o eixo duplo (gráfico de linha acumulada)
5. **Fornecedor x Previsão x SLA**

A Lambda:

- Será acionada sob demanda (via frontend).
- Retornará os dados já agrupados, ordenados e no formato pronto para visualização.
- Será protegida com autenticação e autorizada via API Gateway.
- Permitirá desacoplamento da lógica de agregação do frontend.

- **Visão do Usuário:**

Embora os usuários não acessem diretamente a Lambda, ela garante **a entrega confiável dos dados** que alimentam os gráficos analíticos da Home Page, **proporcionando uma experiência ágil e fluida** na navegação e análise dos dados.

- **Premissas:**

1. Todos os dados necessários estão disponíveis no banco de dados do VO3 - COPF no RDS.
2. A Lambda será deployada com suporte ao pipeline CI/CD do COPF.
3. O contrato de resposta da API será definido em conjunto com o time de frontend.
4. O controle de acesso será feito via autenticação no API Gateway.

- **Regras de Negócio:**

1. **Status dos Equipamentos:**
    - Agrupar por tipo de equipamento.
    - Contabilizar status "operante" e "inoperante".
2. **Top 10 pontos por volume de ocorrências**
    - Listar as 10 agências com maior volume de ocorrências pendentes.
    - Incluir marcador para VIPs.
3. **Aging das Ocorrências:** Agrupar ocorrências abertas pelas faixas de dias:
    - 0 – 1 dia
    - 1 – 2 dias
    - 2 – 4 dias
    - ≥ 5 dias
4. **Long Tail – Motivos:** Agrupar ocorrências por motivo/sintoma e ordenar por frequência decrescente.
5. **Fornecedor x Previsão x SLA:**
    - Agrupar ocorrências por fornecedor.
    - Categorizar conforme:
        - SLA vencido
        - Previsão maior que SLA
        - Sem previsão

- **Informações Técnicas:**
- Linguagem: Python
- Infraestrutura: AWS Lambda + API Gateway
- Monitoramento: Datadog com rastreamento de erro e performance
- Controle de versão: Repositório COPF no GitHub

- **Exemplo de Output:**

```
{
  "aging": [
    { "faixa": "0-1 dias", "quantidade": 32 },
    { "faixa": "1-2 dias", "quantidade": 21 },
    { "faixa": "2-4 dias", "quantidade": 18 },
    { "faixa": "≥5 dias", "quantidade": 9 }
  ],
  "motivos": [
    { "motivo": "ATM Quebrado", "quantidade": 103 },
    { "motivo": "Rede Inoperante", "quantidade": 87 }
  ],
  "status_equipamentos": [
    { "tipo": "Notebook", "operantes": 120, "inoperantes": 12 },
    { "tipo": "ATM", "operantes": 50, "inoperantes": 8 }
  ],
  "top_agencias": [
    { "agencia": "1234", "ocorrencias": 17, "vip": true },
    { "agencia": "5678", "ocorrencias": 15, "vip": false }
  ],
  "fornecedor_sla": [
    { "fornecedor": "ABC", "sla_vencido": 10, "previsao_maior_sla": 5, "sem_previsao": 3 }
  ]
}
```

- **Tarefas:**

1. Definir queries SQL ou views para cada tipo de agregação.
2. Implementar a lógica de agregação na Lambda.
3. Garantir logs detalhados para troubleshooting.

- **Cenários para Teste e Homologação:**

1. Lambda retorna os dados esperados em todos os formatos.
2. Todos os gráficos do frontend carregam corretamente com os dados da API.

3. Logs de erro e sucesso são registrados corretamente.
4. Segurança e autenticação funcionam conforme esperado.

- **Critérios de Aceite:**

☐ Lambda responde com JSON formatado corretamente.

☐ Dados consistem com o que é exibido nas telas atuais.

☐ Segurança via API Gateway implementada.

☐ Código versionado no repositório COPF.

☐ CI/CD funcionando com sucesso.

- **Impacto Esperado:**
- Redução da complexidade no frontend.
- Aumento da performance e responsividade do dashboard.
- Padronização e governança da lógica de agregação.
- Redução de retrabalho em futuras evoluções dos gráficos.

- **Conclusão:**
  - **Início:**
    - **Desejado:** R3 S3 2025
    - **Real:** R3 S3 2025
  - **Fim:**
    - **Desejado:** R3 S3 2025
    - **Real:** *(preencher ao fim da sprint)*
  - **Resultado:** *(preencher ao fim da sprint)*

História: [Delivery][Backend] Lambda de cálculo de prioridade de ocorrências (COPF)

- **Visão de Produto:**

Acreditamos que, ao **automatizar o cálculo de prioridade das ocorrências via Lambda**, permitindo atribuir uma pontuação baseada em critérios objetivos e

estratégicos, **as áreas operacionais poderão priorizar de forma mais eficaz suas ações**, concentrando esforços onde há maior impacto ou urgência.

Saberemos que isso é verdade quando **as ocorrências forem exibidas com uma pontuação de prioridade clara, consistente e alinhada às regras de negócio definidas — incluindo a classificação de criticidade (Crítica, Alta, Média, Baixa)**.

- **Descrição:**

Como time de backend da COPF, queremos **desenvolver uma função Lambda que atribui uma pontuação de prioridade para cada ocorrência**, com base em critérios como localização geográfica, pontos na lista de críticos para o negócio, aging e impacto na operação do ponto.

Essa Lambda será invocada em uma step function que contempla atualmente duas lambdas:

- Lambda de ocorrências pendentes
- Lambda para calculo de priorização

Calcular a **prioridade (score entre 0 e 999)**.

- Atribuir uma **criticidade categórica (Crítica, Alta, Média, Baixa)** com base no score final.
- Armazenar e retornar esse valor junto à ocorrência.
- A API ser exposta via API Gateway com autenticação.

- **Pontos de atenção:**
- A primeira versão dessa lambda, será apenas para ocorrências do segmento AA.
- Para ocorrências do segmento AB, os valores retornamos devem ser nulos.
    - ranking = null
    - criticidade = null

- **Visão do Usuário:**

A prioridade e a criticidade calculadas serão usadas para ordenar (ranking), filtrar (criticidade) e destacar ocorrências, permitindo que analistas e

gestores ajam primeiro nas situações mais críticas com base em critérios padronizados e transparentes.

- **Contexto / Narrativa de Negócio:**

Com essa Lambda, trazemos critérios claros e objetivos que permitem **priorização inteligente e categorizada**, mesmo antes de qualquer ação humana. Para o fornecedor, a criticidade será fundamental para determinar quais ocorrências ele irá priorizar.

- **Premissas:**

1. As bases auxiliares (ex: Base 360 e Ceis_Válidos) estão disponíveis para consulta.
2. As regras de negócio se baseiam no motor de priorização já desenvolvido pela operação da Gimea.

- **Regras de Negócio:**

A pontuação final será a soma de critérios (pontuação total entre 0 e 999):

Exemplo de critérios:

Critério	Descrição	Pontuação
UF = Maranhão	A ocorrência está em agência localizada no Maranhão	+55
Agência VIP	A agência é marcada como VIP	+20
Aging > 5 dias	A ocorrência está aberta há mais de 5 dias	+40
Impacto > 50%	Mais de 50% dos equipamentos da agência estão inoperantes	+30

Cálculo do Impacto do Ponto:

- `impacto (%) = (qtde equipamentos impactados / qtde total equipamentos da agência)`
- A quantidade total de equipamentos será consultada na base **CEIS_Válidos**.
- A quantidade de impactados virá da base atual de ocorrências.

- **Critérios para classificação de criticidade:**

Score de Prioridade	Criticidade
800 a 999	Crítica

Score de Prioridade	Criticidade
600 a 799	Alta
400 a 599	Média
0 a 399	Baixa

- **Informações Técnicas:**
- **Linguagem:** Python 3.11
- **Serviços AWS:** Lambda + API Gateway + DataDog
- **Bases envolvidas:** VO3 (COPF), Ceis_Válidos, Base 360
- **Formato de retorno esperado:**

```
{
  "ocorrencia_id": "abc123",
  "prioridade": 145,
  "criticidade": "Média",
  "componentes": {
    "ma": 55,
    "vip": 20,
    "aging": 40,
    "impacto": 30
  }
}
```

- **Tarefas:**

1. Validar regras de pontuação e faixas de criticidade com PM.
2. Codificar a função Lambda.
3. Implementar cálculo de impacto.
4. Expor endpoint via API Gateway.
5. Testar com ocorrências reais e validar respostas.

- **Cenários para Teste e Homologação:**
- Lambda retorna score correto com base nas características.
- Lambda atribui **criticidade correta conforme faixa de score**.

- Cálculo de impacto da agência retorna valor esperado.
- API responde em tempo adequado e com segurança aplicada.

- **Critérios de Aceite:**

✅ Score de 0 a 999 corretamente calculado.

✅ Criticidade classificada corretamente com base no score.

✅ Endpoint funcional e integrado ao step function.

✅ Código versionado, com logs e testes unitários aplicados.

- **Impacto Esperado:**
- Aumento da assertividade nas ações operacionais.
- Priorização objetiva e alinhada à estratégia do COPF.
- Redução de subjetividade e retrabalho em tratativas.

- **Conclusão:**
- **Início:**
  - **Desejado:** R3 S3 2025
  - **Real:** R3 S3 2025
- **Fim:**
  - **Desejado:** R3 S3 2025
  - **Real:** *(preencher ao fim da sprint)*
- **Resultado:** *(preencher ao fim da sprint)*

História: [Delivery][Backend] Separação de código e nome do ponto + carregamento de código e descrição do sintoma via ETL e contrato da lambda

---

- **Visão de Produto:**

Acreditamos que, ao **separar as informações de código e nome do ponto, bem como código e descrição do sintoma da ocorrência**, poderemos **exibir essas informações com maior clareza e flexibilidade visual no front-end do COPF**, tornando a interface mais limpa, navegável e útil para usuários operacionais.

Saberemos que isso é verdade quando **os usuários visualizarem os dados corretamente segmentados (ex: código do ponto em destaque, nome ao lado; código do sintoma em cinza e descrição expandida), facilitando a leitura, priorização e identificação de ocorrências**.

- **Descrição:**

Como time de backend e dados do COPF, queremos:

1. **Separar o campo de código do ponto do nome do ponto**, para que o front-end possa renderizar esses dados separadamente.
2. **Modificar a estrutura da carga via ETL**, de forma que o sintoma da ocorrência traga **o código do sintoma (ex: "E102")** e sua **descrição (ex: "Equipamento inoperante por falha elétrica")** de forma separada.
3. **Alterar a lambda de detalhe da ocorrência**, que será responsável por retornar o `codigo_sintoma` e `descricao_sintoma` de forma estruturada.
4. **Atualizar o contrato da API no API Gateway**, refletindo as novas propriedades da resposta JSON da lambda.

- **Visão do Usuário:**

Usuários operacionais e gestores precisam visualizar rapidamente **qual ponto está afetado (via código) e onde (nome da agência)**, bem como **entender rapidamente o tipo de sintoma**, sem depender de concatenações confusas ou de múltiplos cliques. A clareza na apresentação dessas informações acelera a análise e tomada de decisão.

- **Contexto / Narrativa de Negócio:**

Atualmente, os campos de nome do ponto e sintoma vêm concatenados (ex: `"3452 - Agência Paulista"` ou `"E102 - Equipamento inoperante"`), o que **dificulta a renderização customizada e destaca excessivamente textos longos**, prejudicando a leitura.

Essa separação permite uma **melhor experiência visual**, padronização de exibição nos cards, filtros e ordenações mais eficientes.

- **Premissas:**
  - O dado de código e nome do ponto já existe de forma estruturada na **tabela** `Agencias`.
    - Codigo Agencia

        - Nome Agencia
    - O dado de código e descrição do sintoma já existe de forma estruturada na **tabela** `Sintoma Ocorrência`.
        - Codigo Sintoma Ocorrência
        - Descrição Sintoma
    - A **lambda de detalhe da ocorrência** será ajustada para retornar esses campos de forma separada.
    - O contrato da API Gateway está versionado e pode ser atualizado.
    - O front-end está apto a consumir os novos campos (`codigo_ponto`, `nome_ponto`, `codigo_sintoma`, `descricao_sintoma`).

- **Regras de Negócio:**
    - **Código do ponto**: deve conter os dígitos numéricos (ex: "3452").
    - **Nome do ponto**: nome descritivo da agência (ex: "Agência Paulista").
    - **Código do sintoma**: parte inicial padronizada (ex: "E102").
    - **Descrição do sintoma**: parte textual (ex: "Equipamento inoperante por falha elétrica").

- **Informações Técnicas:**
    - **Banco de dados:**
        - A tabela `tbl_agencias` já possui `codigo_ponto` e `nome_ponto`.
        - Garantir que a view/tabela de ocorrências faça o join corretamente para expor esses campos.
    - **ETL / pipeline de ingestão:**
        - Adaptar transformação para popular `codigo_sintoma` e `descricao_sintoma`.
    - **Lambda:**
        - A **lambda de detalhe da ocorrência** será atualizada para retornar os campos:

```
{
  "codigo_sintoma": "E102",
```

```
  "descricao_sintoma": "Equipamento inoperante por falha elétrica"
}
```

- **API Gateway:**
  - O contrato da API que consome a lambda será atualizado para refletir os novos campos na resposta JSON.
  - Será necessário atualizar o **schema de resposta e mapeamentos de integração** no API Gateway.

- **Tarefas:**
  1. Atualizar o modelo de dados e a view de ocorrências para buscar `codigo_ponto` e `nome_ponto` da `tbl_agencias`.
  2. Ajustar a transformação ETL para extrair `codigo_sintoma` e `descricao_sintoma`.
  3. Atualizar a lambda de detalhe da ocorrência para retornar os campos de sintoma separadamente.
  4. Atualizar o contrato da API no API Gateway.
  5. Testar o consumo dos novos campos no front-end.

- **Cenários para Teste e Homologação:**

1. A lambda retorna `codigo_sintoma` e `descricao_sintoma` de forma separada.
2. O front-end recebe `codigo_ponto` e `nome_ponto` separadamente.
3. A resposta da API está conforme o novo contrato do API Gateway.
4. Os dados estão corretos, legíveis e sem quebras de compatibilidade.

- **Critérios de Aceite:**

✅ As respostas da API retornam os novos campos estruturados.

✅ O contrato do API Gateway está atualizado e documentado.

✅ O front-end renderiza os campos separadamente, sem erro.

✅ O processo de ingestão (ETL) popula corretamente os campos de sintoma.

- **Impacto Esperado:**
- **Melhoria na clareza das informações exibidas na ferramenta**.

- **Facilidade de leitura e tomada de decisão por parte dos operadores**.
- **Possibilidade de personalizar visualmente os dados sem depender de parsing em front-end**.

- **Conclusão:**
  - **Início:**
    - **Desejado:** R3 S3 2025
    - **Real:** R3 S3 2025
  - **Fim:**
    - **Desejado:** R3 S3 2025
    - **Real:** *(preencher ao fim da sprint)*
  - **Resultado:** *(preencher ao fim da sprint)*

Tarefa: [Discovery][Backend] Exploração dos dados de ocorrências no Signus

- **Visão de Produto:**

Acreditamos que, ao **explorar os dados brutos de ocorrências disponíveis no Data Mesh (sistema produto: Signus)**, seremos capazes de **estruturar os pipelines de ETL de forma robusta, performática e alinhada com os filtros e KPIs necessários no COPF**, evitando retrabalhos e inconsistências em produção.

Saberemos que isso é verdade quando tivermos clareza sobre as tabelas, campos, relacionamentos e filtros necessários para construir os pipelines de **ocorrências pendentes** e **ocorrências encerradas**.

- **Descrição:**

  Como engenheiros responsáveis pela camada de ingestão e preparação de dados do COPF, queremos:

  - **Analisar os dados do Signus** relacionados a ocorrências, com foco em:
    - Diferenciar claramente as ocorrências **pendentes** das **encerradas**.
    - Identificar as tabelas relevantes, suas **chaves**, **status**, **datas**, **sintomas**, **relacionamento com agências** e outros atributos

críticos.

- Identificar os campos que serão utilizados no ETL para permitir **uso futuro no frontend** e nas lambdas.

- **Visão do Usuário:**

O time técnico precisa ter domínio sobre quais tabelas, filtros e lógicas devem ser utilizadas para construir os pipelines de dados, garantindo consistência e performance na exibição das informações operacionais e históricas no COPF.

- **Premissas:**

Os dados estão disponíveis no Mesh com acesso liberado.

- **Tarefas:**

1. Mapear regras de negócios importantes para o ETL.
2. Identificar necessidade de joins com tabelas auxiliares (agências, sintomas, etc).
3. Documentar resultados da exploração.

- **Critérios de Aceite:**

✅ Mapeamento validado de tabelas, status e regras de negócio.

✅ Documentação compartilhada com o time.

✅ Lista clara de campos e filtros a serem usados no ETL.

História: [Delivery][Backend] Desenvolvimento do ETL de Ocorrências Pendentes - Insert

- **Visão de Produto**

Ter uma base confiável e atualizada de ocorrências em aberto é essencial para priorização e gestão eficiente. Este ETL garantirá a ingestão incremental das ocorrências pendentes diretamente do Signos.

- **Descrição**

Como engenheiro(a) de dados, queremos construir um ETL que alimente as tabelas `cadastro_ocorrencia` e `acompanhamento_ocorrencia`, com foco exclusivo em ocorrências que ainda estão abertas no Signus.

- **Regras de Negócio**
- Inserir novas ocorrências em `cadastro_ocorrencia` com os campos definidos na modelagem.
- Para cada nova ocorrência, gerar um registro de abertura em `acompanhamento_ocorrencia`, com:
  - `tipo_evento`: "Abertura"
  - `descricao_evento`: "Ocorrência identificada como nova"
  - `data_hora_comunicacao_enviada`: timestamp atual
- O campo `criticidade_ocorrencia` **não permite valores nulos**, conforme a modelagem.
  - Nesta primeira versão, todas as ocorrências pendentes serão inseridas com **criticidade = "Média"**.
  - Utilizar o **código da criticidade "Média"** da tabela `criticidade_ocorrencia` para preencher esse campo.

- **Informações Técnicas**
  - Executado como Lambda.
  - Modelagem conforme imagem anexa:
    - `cadastro_ocorrencia`
    - `acompanhamento_ocorrencia`
  - O processamento incremental será por partição de dados (ano, mes, dia, hora, minuto).
  - Validação de relacionamento com:
    - `agencia`
    - `equipamento`
    - `sintoma_ocorrencia`
    - `criticidade_ocorrencia`

- **Tarefas**
  1. Mapear campos Signus → Modelagem COPF.

2. Criar função Lambda para ingestão incremental.
3. Popular `cadastro_ocorrencia` com novas pendências.
4. Preencher o campo `criticidade_ocorrencia` com código da criticidade "Média" (por que atualmente demanda um valor (futuramente devemos mudar a modelagem para permitir valor nullable).
5. Gerar entrada correspondente em `acompanhamento_ocorrencia`.
6. Validar se dados relacionais (agência, equipamento, sintoma, criticidade) estão consistentes.
7. Testar execução manual.
8. Garantir logging básico e rastreabilidade.

- **Critérios de Aceite**

☐ Ocorrências novas alimentam corretamente a tabela `cadastro_ocorrencia`.

☐ Cada nova ocorrência gera um evento de abertura em `acompanhamento_ocorrencia`.

☐ Não há duplicidade em reprocessamentos.

☐ Campo `criticidade_ocorrencia` é preenchido corretamente com código da criticidade "Média".

☐ Logging simples está ativo.

História: [Delivery][Backend] Desenvolvimento do ETL de Ocorrências Pendentes - Update

---

- **Visão de Produto**

Garantir que os dados das ocorrências em aberto estejam sempre atualizados é essencial para refletir a realidade operacional nas ferramentas de gestão. No ETL de Ocorrências Pendentes, há a capacidade de atualizar periodicamente alguns dos campos das ocorrências já inseridas, assegurando confiabilidade nas análises e nas decisões.

- **Descrição**

Como engenheiro(a) de dados, queremos implementar uma etapa no ETL de Ocorrências Pendentes que atualize os campos relevantes das ocorrências que

continuam abertas no Signus, assegurando que alterações de status, fornecedor, impedimentos, SLA e datas sejam refletidas nas tabelas `cadastro_ocorrencia` e `acompanhamento_ocorrencia`.

- **Regras de Negócio**
  - Atualizar apenas ocorrências que já estejam registradas na tabela `cadastro_ocorrencia`.
  - Atualizar os seguintes campos:
    - `fornecedor_id`
    - `status_ocorrencia`
    - `sla_ocorrencia`
    - `criticidade_ocorrencia` (não nula; manter coerência com classificação atualizada pelo motor de priorização)
    - `previsao_atendimento`
    - `possui_impedimento` e `motivo_impedimento`
  - Atualizações que geram registro na tabela `acompanhamento_ocorrencia`:
    - Alteração de previsão de atendimento
      - `tipo_evento`: "Atualização Previsão de Atendimento"
      - `descricao_evento`: descrição do(s) campo(s) alterado(s)
      - `data_hora_comunicacao_enviada`: timestamp atual
    - Inclusão de impedimento/Alteração de impedimento/Remoção de impedimento
      - `tipo_evento`: "Inclusão de Impedimento"
      - `descricao_evento`: descrição do(s) campo(s) alterado(s). Incluir motivo.
      - `data_hora_comunicacao_enviada`: timestamp atual

- **Informações Técnicas**
  - Executado na Lambda do 'ETL de Ocorrências Pendentes'.
  - Atualização incremental por partição de dados (ano, mês, dia, hora, minuto).

- Leitura comparativa dos dados atuais no banco versus payload do Signus.
- Garantir integridade referencial com:
  - `agencia`
  - `equipamento`
  - `sintoma_ocorrencia`
  - `criticidade_ocorrencia`

- **Tarefas**
  1. Mapear campos do Signus que podem sofrer atualização.
  2. Construir lógica de detecção de alterações campo a campo.
  3. Criar rotina incremental no Lambda com controle por timestamp.
  4. Atualizar os dados em `cadastro_ocorrencia` apenas se houver alteração.
  5. Gerar entrada em `acompanhamento_ocorrencia` com campos alterados.
  6. Validar consistência de chaves estrangeiras.
  7. Implementar logs de alterações para rastreabilidade.
  8. Testar a execução com casos simulados de atualização.

- **Critérios de Aceite**

☐ Apenas ocorrências existentes são atualizadas (sem sobrescrever ocorrências encerradas).

☐ Algumas alteração geram eventos de atualizações em `acompanhamento_ocorrencia`.

☐ Campos atualizados refletem as mudanças no Signus.

☐ Logs permitem rastrear quais ocorrências foram atualizadas e quais campos mudaram.

Tarefa: [Infraestrutura][EventBridge] Parametrização do cron para o gatilho de ocorrências pendentes

Visão de Produto:

Orquestrar execução frequente da função de ingestão de ocorrências em aberto.

Descrição:

Criação de regra de cron do EventBridge para execução a cada 15 minutos da lambda de ocorrências pendentes.

Premissas:

- Lambda já publicada.
- Permissões IAM criadas.

Tarefas:

- Criar regra de cron.
- Apontar para lambda.
- Validar execução automatizada.

Critérios de Aceite:

- Lambda acionada periodicamente.
- Logs de execução disponíveis.

Impacto Esperado:

Atualização constante da base com baixa latência.

Tarefa: [Deploy][Homologacao] Deploy da esteira de ocorrências pendentes

Visão de Produto:

Validar a esteira de ocorrências pendentes em ambiente de homolog para testes integrados.

Descrição:

Publicação de lambdas, variáveis, banco, EventBridge e testes em ambiente de homolog. Garantir que dados estão sendo processados corretamente.

Premissas:

- Todas as etapas anteriores finalizadas.

Tarefas:

- Criar stack de homolog.
- Publicar lambdas e agendadores.
- Validar fluxo de ponta a ponta.

Critérios de Aceite:

- Todos os registros corretamente gerados.
- Logs e auditoria validados.

Impacto Esperado:

Pronto para avaliação do time funcional e stakeholders.

História: [Infraestrutura][Banco de Dados] Criar réplica de leitura para o RDS

Visão de Produto:

Garantir separação entre escrita e leitura para evitar concorrência e gargalos de performance.

Descrição:

Configuração de uma réplica read-only do RDS para consumo pelas lambdas de leitura e análise.

Premissas:

- Utilização de RDS MySQL.
- Permissão de acesso via IAM configurada.

Tarefas:

- Criar réplica.
- Garantir sincronização automatizada.
- Atualizar variáveis de ambiente das lambdas de leitura.

Critérios de Aceite:

- Réplica criada e funcional.
- Queries de leitura usando somente réplica.

Impacto Esperado:

Aumento de performance e estabilidade.

Seção: IBS 360 - Plataforma

Seção: IBS 360 - Frontend

Seção: IBS 360 - Gestão do Parque

Seção: IBS 360 - Gestão do Parque - Acompanhamento de Esteiras

Seção: IBS 360 - Geocompasso

Seção: Radar Imobiliário

Seção: Inteligência Imobiliária

História: [Delivery][Backend] Criação do ETL para Geração da Base Filtrada a partir dos Parâmetros do Planejamento de Negociações

- **Visão de Produto:**

Acreditamos que, ao **criar um processo de ETL para filtrar a base original de dados imobiliários com base nos parâmetros preenchidos no Frontend do Planejamento de Negociações**, para **preparar uma base específica e otimizada para uso pelo time de Advanced Analytics**, resultará em **um processo mais eficiente, seguro e rastreável para execução do modelo de otimização**.

- **Descrição:**

Como time responsável pela engenharia do IBS 360, queremos **desenvolver e operacionalizar um ETL que leia os dados recebidos pelo ETL1 (dados da GPA**

e Osiris), filtre a base imobiliária consolidada no ETL1 e gere **uma base reduzida, segura e rastreável**, que será utilizada nas simulações feitas pelo time de Advanced Analytics.

- **Visão do Usuário:**

O usuário não visualiza diretamente a base filtrada, mas se beneficia de **simulações mais rápidas, coerentes com os filtros aplicados no painel, com rastreabilidade clara entre os dados preenchidos e os resultados retornados**.

- **Contexto/Narrativa de Negócio:**

O ETL criará **bases de trabalho específicas por simulação, permitindo rastreabilidade via UUID e promovendo um fluxo limpo para execução dos modelos**.

- **Premissas:**
    1. A base consolidada de imóveis está disponível via ETL1.
    2. Os filtros preenchidos estão no frontend.
    3. A arquitetura suporta a geração de arquivos versionados por simulação.
- **Regras de Negócio:**
    1. A base deve conter somente os imóveis elegíveis conforme os filtros informados.
    2. O arquivo final deve estar associado a um UUID de simulação.
- **Informações Técnicas:**
    1. Leitura da base ETL1 consolidada.
    2. Aplicação dos filtros preenchidos pelo usuário.
    3. Logging estruturado no CloudWatch.
- **Tarefas:**
    1. Criar script de filtragem.
    2. Escrever base filtrada.
    3. Automatizar a execução com.
    4. Testar fluxos de ponta a ponta com múltiplos cenários.
    5. Documentar estrutura de diretórios, nomenclatura de arquivos e regras aplicadas.

- **Cenários para Teste e Homologação:**
  1. Testar execução com diferentes conjuntos de filtros.
  2. Validar que apenas registros elegíveis estão na base final.
  3. Verificar se o UUID aparece corretamente no nome/pasta.
  4. Confirmar que arquivos são legíveis por processos posteriores (ex: modelos em Python).
- **Impacto Esperado:**
  - Base otimizada e segura para execução de modelos.
  - Redução de tempo e custo de processamento nas simulações.
  - Garantia de rastreabilidade entre preenchimento → base gerada → resultado.
- **Conclusão:**
  - **Início:**
    - **Desejado:** R2 S2 2025
    - **Real:** R2 S2 2025
  - **Fim:**
    - **Desejado:** R2 S2 2025
    - **Real:**
  - **Resultado:**
    - R3 S2 2025 - Desenvolvimento iniciado com a colaboradora Beatriz Moura.

Seção: Score de Agências

Seção: App Planejamento de Pessoas

Seção: Arquitetura Cross

Seção: Monitoramento e Métricas de Produto

Seção: AWSCloudBridge