Desafio Técnico Nível 3: JUSCASH
Documentação Técnica Consolidada
Candidato: Vinicius Oliveira Dos Santos, Data: 05/11/2026

Sumário Executivo

- **Total de Bugs Corrigidos:** 9 Obrigatórios + 9 Bônus
- **Tempo Total Gasto no Case**: Aprox. 7 horas
- **Principais Desafios Enfrentados:** O principal desafio foi o diagnóstico do Bug Crítico do Python (Chamado #2847). O chamado indicava uma "falha ao salvar o arquivo Excel" , mas a análise sistemática dos logs provou que a causa raiz era, na verdade, um bug no Scheduler (agendador) que impedia a execução em produção fora do horário estipulado. Isso exigiu diagnosticar que a premissa do próprio chamado estava incorreta.
- **Lições Aprendidas:** O case reforçou a importância do tratamento robusto de exceções (usar throw nos blocos catch do Node.js em vez de apenas logar), o acesso seguro a propriedades aninhadas (optional chaining no Next.js para evitar crashes de renderização) e a diferença crítica entre código assíncrono e síncrono (writeFile vs writeFileSync no Node.js)

Índice

CASE 1: Sistema de Coleta de Dados (Python)

Bug #1 - Falha ao Salvar Arquivo Excel - Stack: Python

- **Arquivo:** app/scheduler.py
- **Linha:** 18 - pode_executar()
- **Severidade:** CRÍTICA
- **Problema:** O módulo de agendamento “app/scheduler.py” continha duas filhas críticas que impediam a execução

  1 - Modo Developement(Bug A): Esse modo que deveria ignorar o horário não estava funcionando. O código não verifica “settings.APP_ENV”, fazendo com que a execução falhasse mesmo com “settings.APP_ENV=development”

- **Como Identifiquei:**

  1- Bug A: Tentei rodar “APP_ENV=development python app/main.py”. O log mostrou que o ambiente foi detectado, mas a verificação de horário falhou, provando que o if() de verificação de ambiente estava ausente.

  2- Bug B: Tentei rodar “APP_ENV=production python app/main.py”. O log mostrou que o script parou no "PASSO 1" por falha na verificação de horário, identificando esta como a causa raiz da "Produção parada" .

- **Correção Aplicada:**

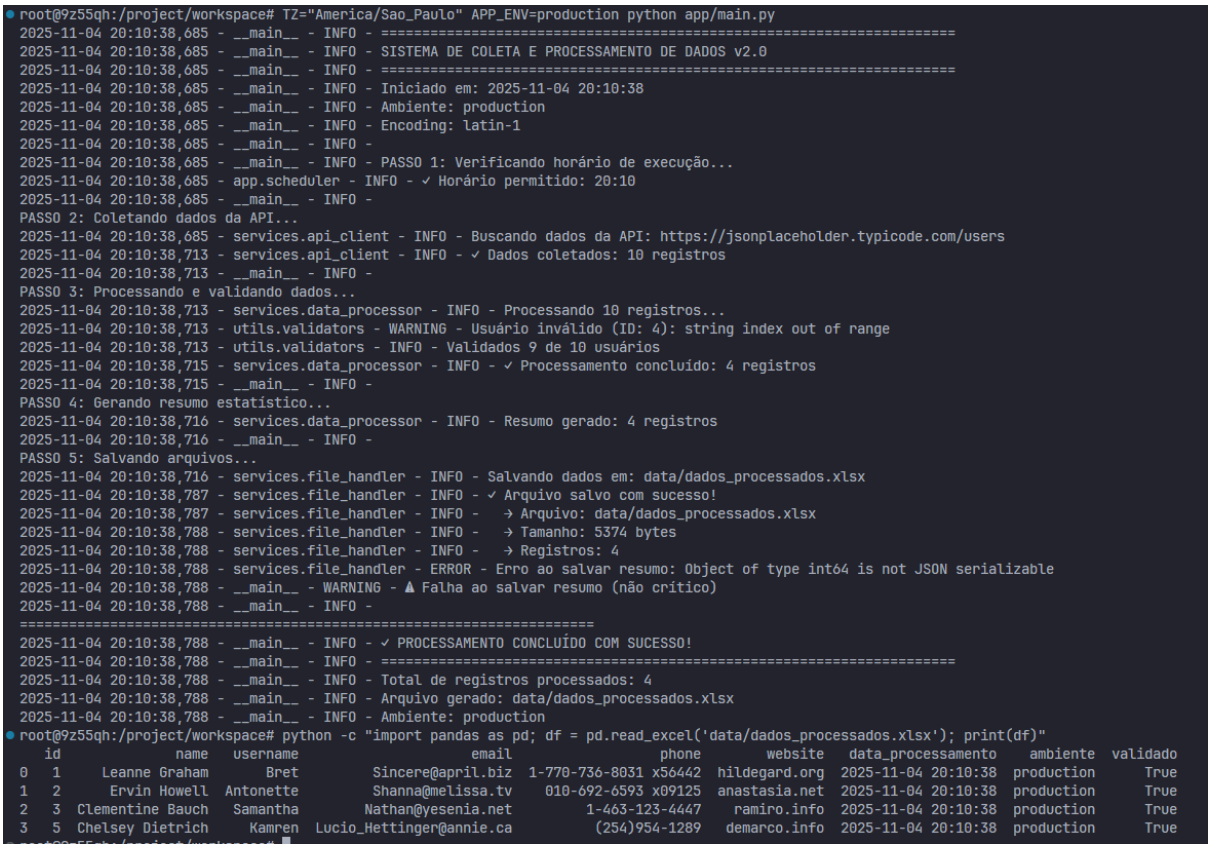
  1- Bug A: Adicionei a lógica de verificação do ambiente no início da função pode_executar()

  2 - Bug B: A correção/proposta seria remover toda a lógica de verificação de horário de “app/scheduler.py”, pois a aplicação não deve ser responsável pelo seu próprio agendamento. Para teste alterei o “HORARIO_EXECUCAO” em “config/settings.py” para permitir a execução.

- **Teste (Evidências):**

  1 - Após a correção A: rodei “APP_ENV=development python app/main.py” e o log mostrou o script prosseguindo

  2- Após contornar o Bug B alterando o horário, rodei “APP_ENV=production python app/main.py” e o log mostrou o script executando o fluxo completo e salvando o Excel



- **Tempo Gasto:** 1 hora e 30 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-python/commit/bddbe723d757c79b2c055f7c9c512cf806f4247f

---

Bug #2 - Dados Faltando no Relatório - Stack: Python

- **Arquivo:** utils/validators.py
- **Linha:** N/A
- **Severidade:** ALTA
- **Problema:** Reportava que o relatório final continha apenas 4 usuários em vez de 5, com suspeita no ID 7.
- **Como Identifiquei:**

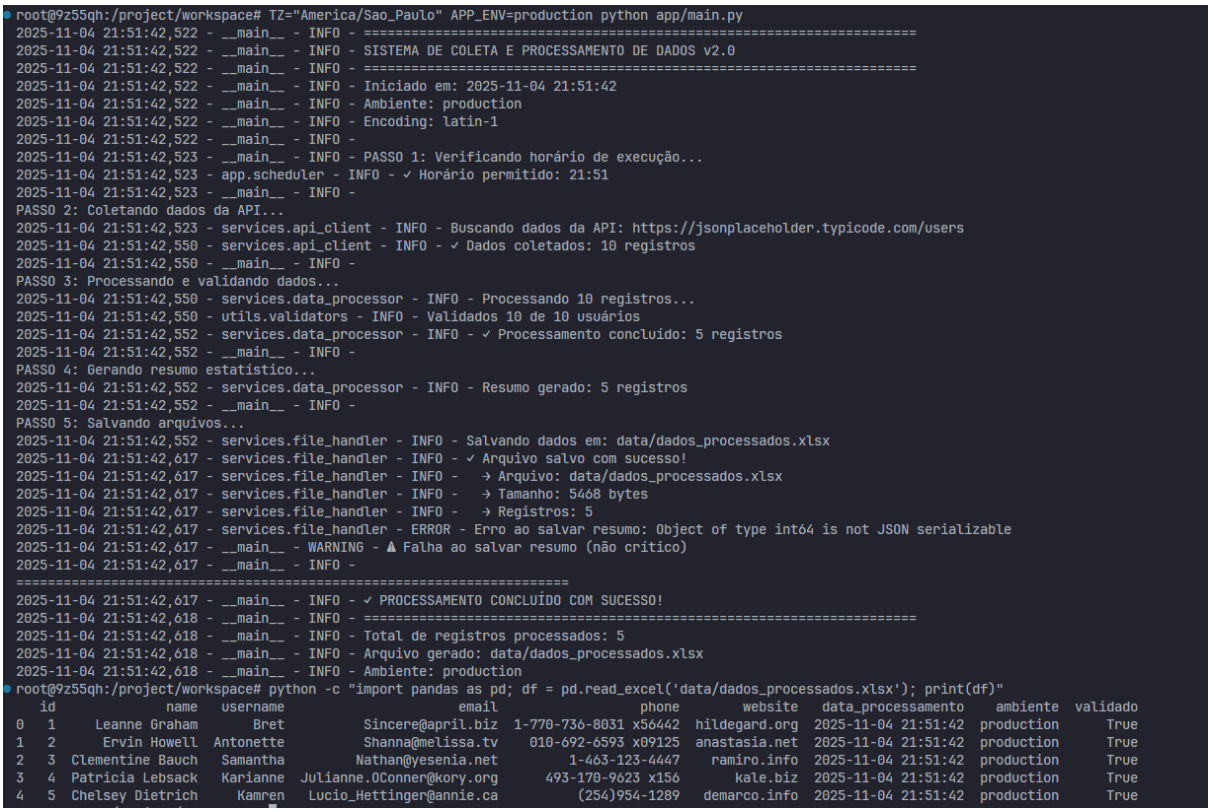
  O log de validação “Usuário inválido (ID: 4)” provou que a causa raiz era o Bug #3, que estava descartando o ID 4.

- **Correção Aplicada:**

Ao corrigir o Bug #3 (erro de validação), O usuário 4 deixou de ser descartado, resolvendo automaticamente esse Bug

- **Teste (Evidências):**

  O log final mostra “Total de registros processados: 5”. O comando de validação “python -c "import pandas..."” também imprimiu o Data Frame completo com os IDs 1, 2, 3, 4 e 5.



```
root@9z55qh:/project/workspace# TZ="America/Sao_Paulo" APP_ENV=production python app/main.py
2025-11-04 21:51:42,522 - __main__ - INFO - ======================================================================
2025-11-04 21:51:42,522 - __main__ - INFO - SISTEMA DE COLETA E PROCESSAMENTO DE DADOS v2.0
2025-11-04 21:51:42,522 - __main__ - INFO - ======================================================================
2025-11-04 21:51:42,522 - __main__ - INFO - Iniciado em: 2025-11-04 21:51:42
2025-11-04 21:51:42,522 - __main__ - INFO - Ambiente: production
2025-11-04 21:51:42,522 - __main__ - INFO - Encoding: latin-1
2025-11-04 21:51:42,522 - __main__ - INFO -
2025-11-04 21:51:42,523 - __main__ - INFO - PASSO 1: Verificando horário de execução...
2025-11-04 21:51:42,523 - app.scheduler - INFO - ✓ Horário permitido: 21:51
2025-11-04 21:51:42,523 - __main__ - INFO -
PASSO 2: Coletando dados da API...
2025-11-04 21:51:42,523 - services.api_client - INFO - Buscando dados da API: https://jsonplaceholder.typicode.com/users
2025-11-04 21:51:42,550 - services.api_client - INFO - ✓ Dados coletados: 10 registros
2025-11-04 21:51:42,550 - __main__ - INFO -
PASSO 3: Processando e validando dados...
2025-11-04 21:51:42,550 - services.data_processor - INFO - Processando 10 registros...
2025-11-04 21:51:42,550 - utils.validators - INFO - Validados 10 de 10 usuários
2025-11-04 21:51:42,552 - services.data_processor - INFO - ✓ Processamento concluído: 5 registros
2025-11-04 21:51:42,552 - __main__ - INFO -
PASSO 4: Gerando resumo estatístico...
2025-11-04 21:51:42,552 - services.data_processor - INFO - Resumo gerado: 5 registros
2025-11-04 21:51:42,552 - __main__ - INFO -
PASSO 5: Salvando arquivos...
2025-11-04 21:51:42,552 - services.file_handler - INFO - Salvando dados em: data/dados_processados.xlsx
2025-11-04 21:51:42,617 - services.file_handler - INFO - ✓ Arquivo salvo com sucesso!
2025-11-04 21:51:42,617 - services.file_handler - INFO -   → Arquivo: data/dados_processados.xlsx
2025-11-04 21:51:42,617 - services.file_handler - INFO -   → Tamanho: 5468 bytes
2025-11-04 21:51:42,617 - services.file_handler - INFO -   → Registros: 5
2025-11-04 21:51:42,617 - services.file_handler - ERROR - Erro ao salvar resumo: Object of type int64 is not JSON serializable
2025-11-04 21:51:42,617 - __main__ - WARNING - ⚠ Falha ao salvar resumo (não crítico)
2025-11-04 21:51:42,617 - __main__ - INFO -
============================================================
2025-11-04 21:51:42,617 - __main__ - INFO - ✓ PROCESSAMENTO CONCLUÍDO COM SUCESSO!
2025-11-04 21:51:42,618 - __main__ - INFO - ======================================================================
2025-11-04 21:51:42,618 - __main__ - INFO - Total de registros processados: 5
2025-11-04 21:51:42,618 - __main__ - INFO - Arquivo gerado: data/dados_processados.xlsx
2025-11-04 21:51:42,618 - __main__ - INFO - Ambiente: production
root@9z55qh:/project/workspace# python -c "import pandas as pd; df = pd.read_excel('data/dados_processados.xlsx'); print(df)"
   id              name   username                email                phone        website   data_processamento    ambiente  validado
0   1     Leanne Graham       Bret          Sincere@april.biz  1-770-736-8031 x56442  hildegard.org  2025-11-04 21:51:42  production      True
1   2      Ervin Howell  Antonette          Shanna@melissa.tv    010-692-6593 x09125  anastasia.net  2025-11-04 21:51:42  production      True
2   3  Clementine Bauch   Samantha         Nathan@yesenia.net          1-463-123-4447    ramiro.info  2025-11-04 21:51:42  production      True
3   4  Patricia Lebsack   Karianne  Julianne.OConner@kory.org      493-170-9623 x156       kale.biz  2025-11-04 21:51:42  production      True
4   5  Chelsey Dietrich     Kamren   Lucio_Hettinger@annie.ca          (254)954-1289   demarco.info  2025-11-04 21:51:42  production      True
```

- **Tempo Gasto:** 45 minutos (junto com Bug #3)
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-python/commit/2d8a55802154ba8de5fc44332fa84ca43abf1ed0

Bug #3 - Erro Intermitente na Validação - Stack: Python

- **Arquivo:** utils/validators.py
- **Linha:** linha 50 - “validação adicional de email”
- **Severidade:** MÉDIA
- **Problema:**

Reportava um “KeyError” aleatório, logs mostrou que o erro de validação era na real um “string index out of range” ao processar o usuário com ID 4

- **Como Identifiquei:**

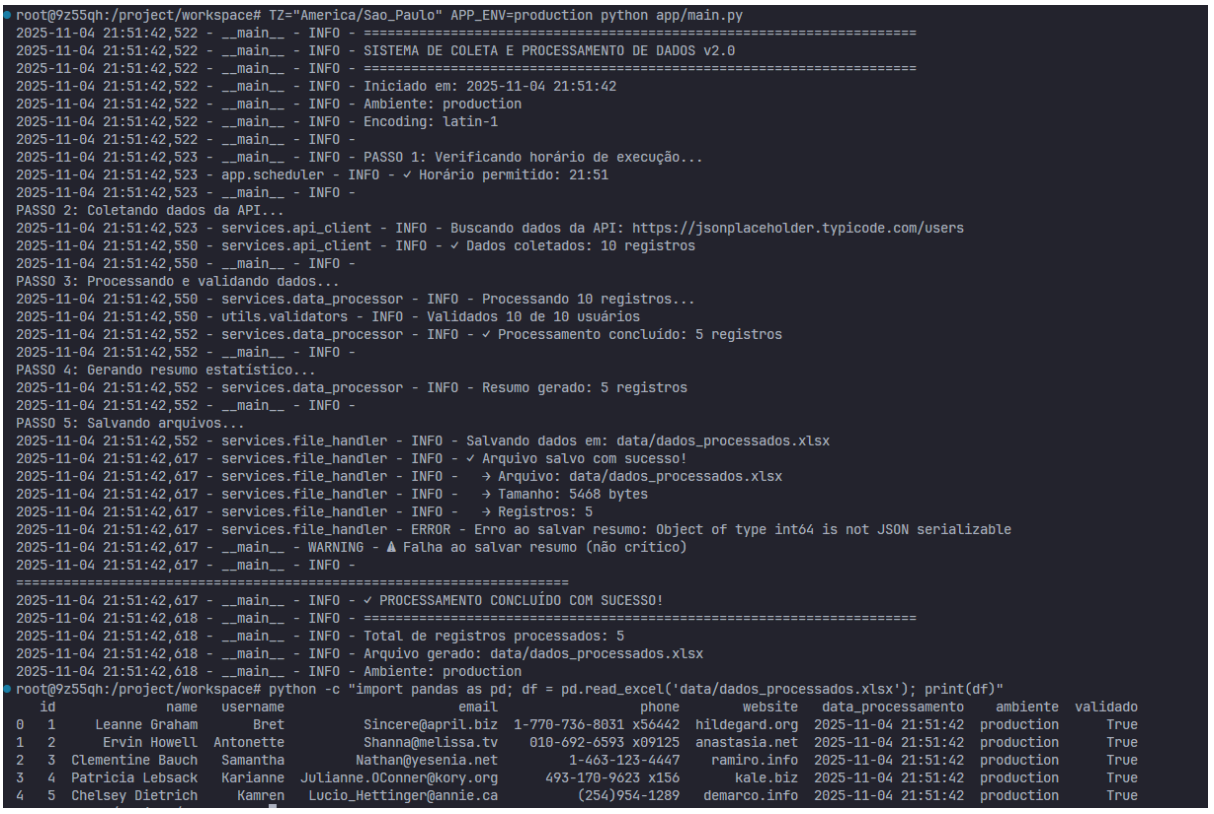
  O log mostrou claramente “utils.validators - WARNING - Usuário inválido (ID: 4): string index out of range”, analisando o arquivo “utils/validators.py”, encontrei um bloco de código que tentava acessar “email[100] em um e-mail que para o ID 4 tinha 25 caracteres

- **Correção Aplicada:**

  Apenas comentei o código defeituoso, pois ele não representava nenhuma regra de negócio válida

- **Teste (Evidências):**

  Após a correção, log não mostrou mais o erro e todos os usuários foram validados



- **Tempo Gasto:** 45 minutos (junto com Bug #2)
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-python/commit/2d8a55802154ba8de5fc44332fa84ca43abf1ed0

Bug #4 - BÔNUS - Erro 'int64 is not JSON serializable' - Stack: Python

- **Arquivo:** services/file_handler.py
- **Linha**: save_summary() e a adição da classe NpEncoder
- **Severidade:** BÔNUS
- **Problema:**

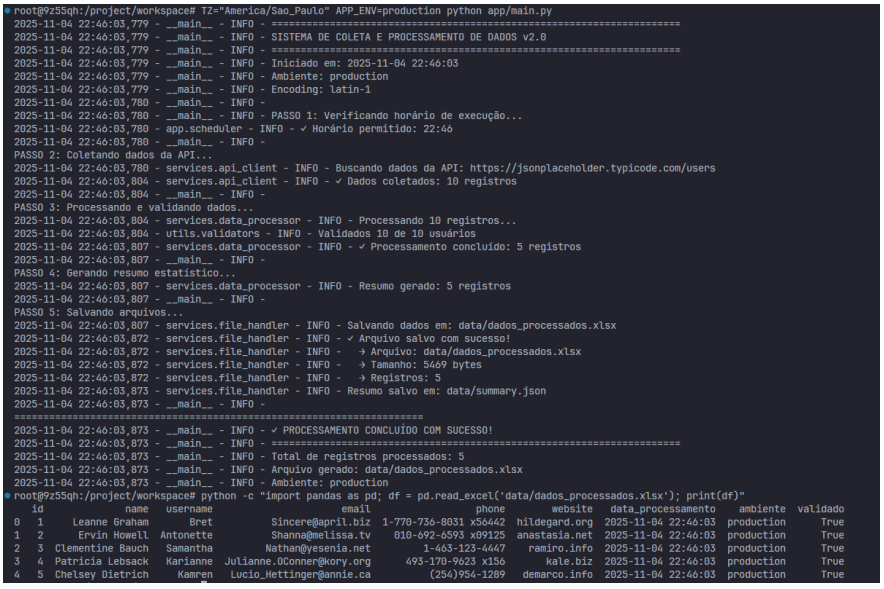
  No log mostrava “ERROR - Erro ao salvar resumo: Object of type int64 is not JSON serializable”. Isso ocorria porque o script tentava salvar o summary.json com um tipo de dado numérico do Numpy (o np.int64), que a biblioteca json padrão do Python não reconhece.

- **Como Identifiquei:**

  O erro ficava bem evidente no log após os registros salvos no “PASSO 5”

- **Correção Aplicada:**
  1. Importei numpy as np e json no services/file_handler.py.
  2. Criei uma classe NpEncoder customizada que herda de json.JSONEncoder.
  3. Dentro dela, implementei um método default que usa isinstance(obj, np.int64) para detectar o tipo problemático e o converte para um int() padrão do Python.
  4. Passei esta classe para a função json.dump() usando o argumento cls=NpEncoder.
- **Teste (Evidências):**

  Executei em modo produção e o erro JSON serializable desapareceu, indicando que o summary.json foi salvo com sucesso.



- **Tempo Gasto:** 50 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-python/commit/15b725805189c70cc8d357c1b6229d576ec128f4

CASE 2: Sistema de Webhooks (Node.js)

Bug #4 - Aplicação Não Salva Arquivo JSON - Stack: Node.js

- **Arquivo:** utils/fileWriter.ts
- **Linha:** linha 45, dentro da função saveToJson
- **Severidade:** CRÍTICA
- **Problema:**

  Reportava que a aplicação falhava ao salvar o arquivo JSON, impedindo a execução.

- **Como Identifiquei:**

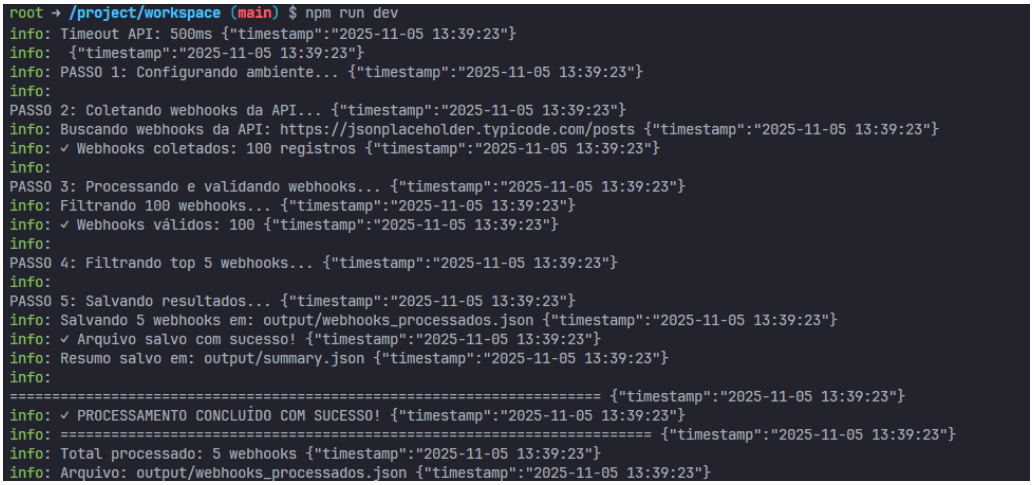
  O log mostrou o erro exato “TypeError: promises_1.default.writeFileSync is not a function”. Analisando src/index.ts, vi que o "PASSO 5: Salvando resultados..." chamava “this.fileWriter.saveToJson”. Dentro de utils/fileWriter.ts, o código importava fs de "fs/promises" (assíncrono), mas tentava usar a função síncrona fs.writeFileSync, que não existe nesse módulo.

- **Correção Aplicada:**

  Na função “saveToJson”, substituí a chamada incorreta await “fs.writeFileSync(...)” pela função assíncrona correta: “await fs.writeFile(...)”.

- Teste (Evidências):

  Após a correção, rodei npm run dev novamente. O script executou até o fim, sem TypeError, e gerou o arquivo “output/webhooks_processados.json”.



- Tempo Gasto: 30 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/e8425a34fd48f13b733a6fa62cfba970fb5a0eff

---

Bug #5 - Webhooks Errados no Arquivo - Stack: Node.js

- **Arquivo:** src/services/webhookService.ts
- **Linha:** 132, a função getTopWebhooks
- **Severidade:** ALTA
- **Problema:**

  Reportava que o arquivo salvo continha os webhooks errados (IDs 96-100) em vez dos 5 primeiros (IDs 1-5).

- **Como Identifiquei:**

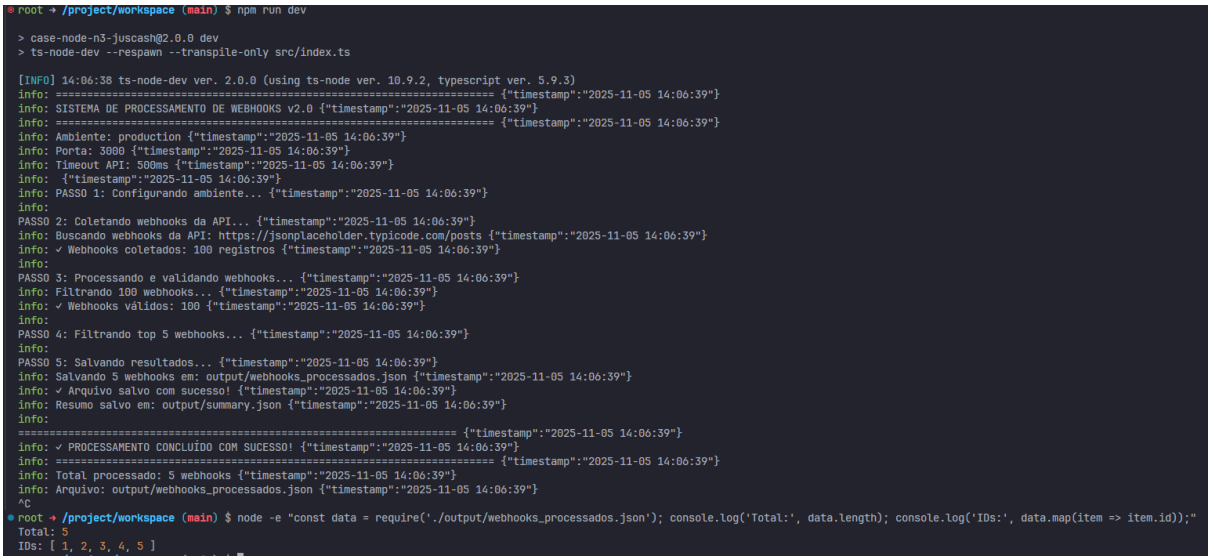
  1.Executei o comando de validação: node -e "const data = require('./output/webhooks_processados.json'); console.log('IDs:', data.map(item => item.id));". A saída foi IDs: [96, 97, 98, 99, 100]

  2.Analisando “src/services/webhookService.ts”, encontrei na função “getTopWebhooks” uma lógica de ordenação invertida: “webhooks.sort((a, b) => b.id - a.id)”, que ordenava por ID decrescente.

- **Correção Aplicada:**

  Inverti a lógica de ordenação para crescente: “webhooks.sort((a, b) => a.id - b.id)”. Isso garante que o .slice(0, 5) pegue os 5 primeiros IDs (1-5).

- **Teste (Evidências):** A saída agora foi a correta: Total: 5 IDs: [1, 2, 3, 4, 5]



- **Tempo Gasto:** 25 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/88afbdc17d6963e21def8fa348ed9c6687a06e68

Bug #6 - Timeouts Frequentes em Produção - Stack: Node.js

- **Arquivo:** src/config/settings.ts
- **Linha:** 39, na função getTimeOut
- **Severidade:** MÉDIA
- **Problema:**

  Reportava que a aplicação funcionava em dev, mas sofria com timeouts (ECONNABORTED) no ambiente de production.

- **Como Identifiquei:**

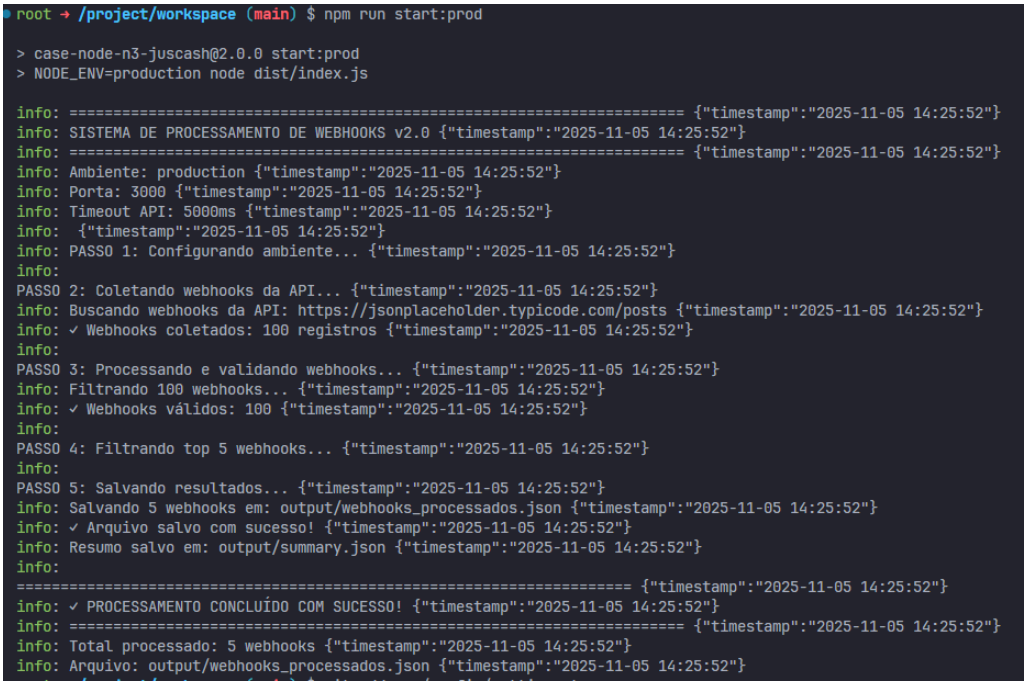
  O log de execução mesmo em dev, mostrava: “Timeout API: 500ms. Analisando src/config/settings.ts”, encontrei a função “getApiTimeout” que continha um bug intencional, ela retornava um valor fixo de 500ms se NODE_ENV fosse production, um valor muito baixo que causaria os timeouts

- **Correção Aplicada:**

  Altei a lógica da função “getApiTimeout” para que ela retorne “this.API_TIMEOUT” (o valor correto de 5000ms) em todos os ambientes, incluindo production.

- **Teste (Evidências):**

  Executei “npm run start:prod” para rodar no cenário de produção. O novo log mostrou “Timeout API: 5000ms” (o valor corrigido).



- **Tempo Gasto:** 30 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/f27173d15b2c2cbbc5f9c4c9fde0bdd2c70b33e6

Bug #7 [BÔNUS - Correção de Múltiplos Bugs Bônus] [Stack: Node.js]

- **Arquivo:** src/services/webhookService.ts, src/config/settings.ts, src/utils/Validators.ts, src/index.ts
- **Severidade:** BÔNUS (MÉDIA/BAIXA)
- **Problema:**

  O código continha múltiplos bugs "bônus", indicam má qualidade de código, riscos futuros e tratamento de erro inadequado

- **Como Identifiquei e Corrigi:**

  Erro de Build (TS6133): O npm run build falhava ao reclamar da variável invalid não utilizada em “webhookService.ts”. **Correção:** O bloco de código foi comentado, limpando o build

Validação de Ambiente: A função “validateEnvironment” em “settings.ts” sempre retornava true. **Correção:** A lógica foi corrigida para return “validEnvs.includes(this.NODE_ENV)”

Falhas de Validação: O arquivo “Validators.ts” continha lógica incorreta para tipos de userId e para o tamanho máximo do "batch". **Correção:** A lógica foi corrigida para “typeof... !== "number"” e para return false se o batch fosse muito grande.

Tratamento de Erro (Swallowing): Várias funções (em fileWriter.ts, index.ts) capturavam erros (catch) mas não os "lançavam" (throw), impedindo a aplicação de parar em caso de falha. **Correção:** Adicionei “throw error” nos blocos catch relevantes e “process.exit(1)” no constructor do index.ts.

- **Teste (Evidências):**

  O comando npm run build passou a ser executado sem nenhum erro. A lógica de validação e tratamento de erros da aplicação agora está robusta e correta.

- **Tempo Gasto:** 35 minutos
- **Link dos Commits:**

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/aa0042c9f41128d5abcb88682c598e5601937ec8

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/9d14da82b0cd1c708677b06bd997b71ea09a2d0d

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/c8b882961ab74d9231de0c1e2d40d591d3aa3cdf

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/3b1bb9f088ad0b9c8e3e834975a4c73455185e7b

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/8649afcde4e0bc89885ea3428956d2e77256321d

  https://github.com/Viniciusoliver8/juscash-desafio-node/commit/55e998e97d4aef6100b113f73b71f9b7f464e416

CASE 3: Dashboard de Usuários (Next.js)

Bug #7 - Dashboard Não Mostra Usuários - Stack: Next.js

- **Arquivo:** src/app/components/UserList.tsx
- **Linha:** 27, na função featchUsers
- **Severidade**: CRÍTICA
- **Problema:**

  Reportava que o dashboard falhava ao carregar usuários. Isso era causado por dois problemas neste arquivo:

  1. (Erro 404): O fetch estava chamando uma URL com erro de digitação (/api/usres em vez de /api/users), causando o erro 404

  2. (Dados Vazios): O código tentava acessar data.users para definir o estado, mas a API retorna os dados dentro de data.data. Isso causaria a tela de "Nenhum usuário encontrado"

- **Como Identifiquei:**

  1. Rodei “npm run dev” e o "Preview" do navegador mostrou "Erro ao carregar usuários"

  2. Abri o Console do Navegador e identifiquei o GET falhando com 404 (Not Found) para a URL “/api/usres?limit=5:1”

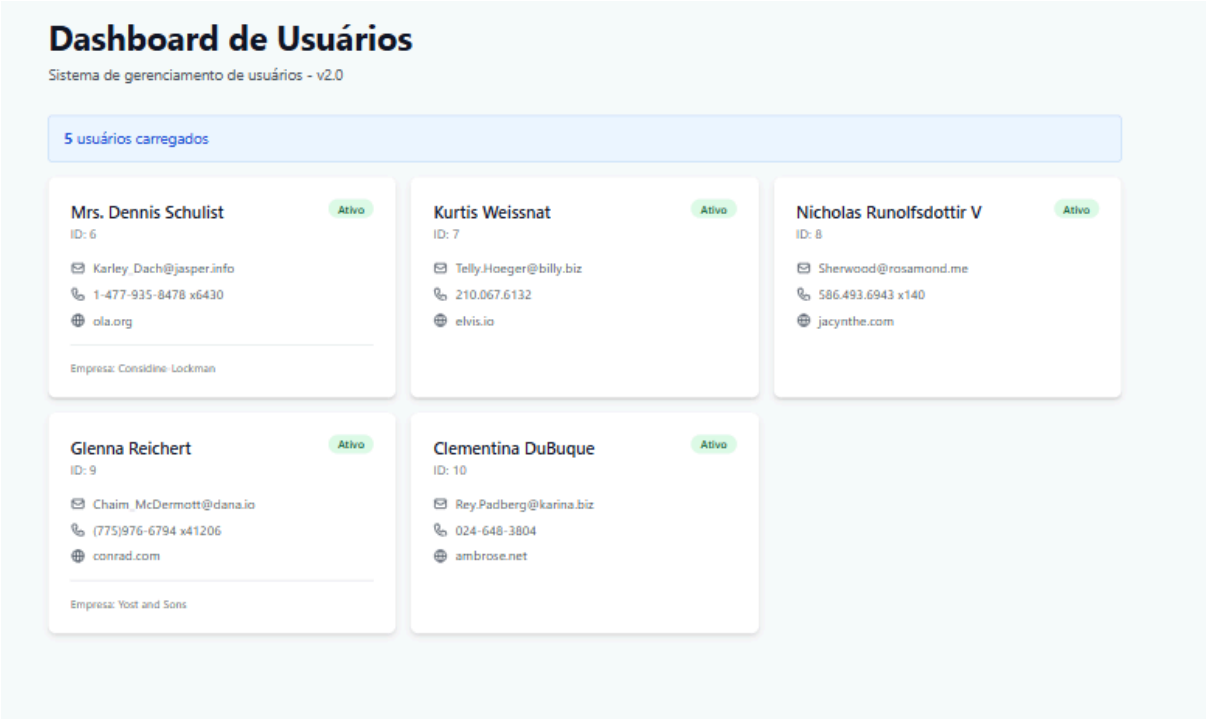
  3. Analisei o UserList.tsx e encontrei tanto a URL incorreta quanto o acesso incorreto aos dados (data.users)

- **Correção Aplicada:**

  Corrigi a URL na função fetch para "/api/users?limit=5"e alterei a função setUsers para ler “data.data || []”

- **Teste (Evidências):**

  Após salvar o arquivo, o "Preview" recarregou automaticamente. O erro desapareceu e o dashboard agora carrega 5 usuários.



- **Tempo Gasto:** 30 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-nextjs/commit/8d666ae5edf014bfc86fe67787d82afae55e03b7

Bug #8 - Usuários Errados Aparecem - Stack: Next.js

- **Arquivo:** src/app/api/users/route.ts
- **Linha:** 29
- **Severidade:** ALTA
- **Problema:**

  O dashboard exibia os usuários errados (IDs 6-10) em vez dos corretos (IDs 1-5) .

- **Como Identifiquei:**

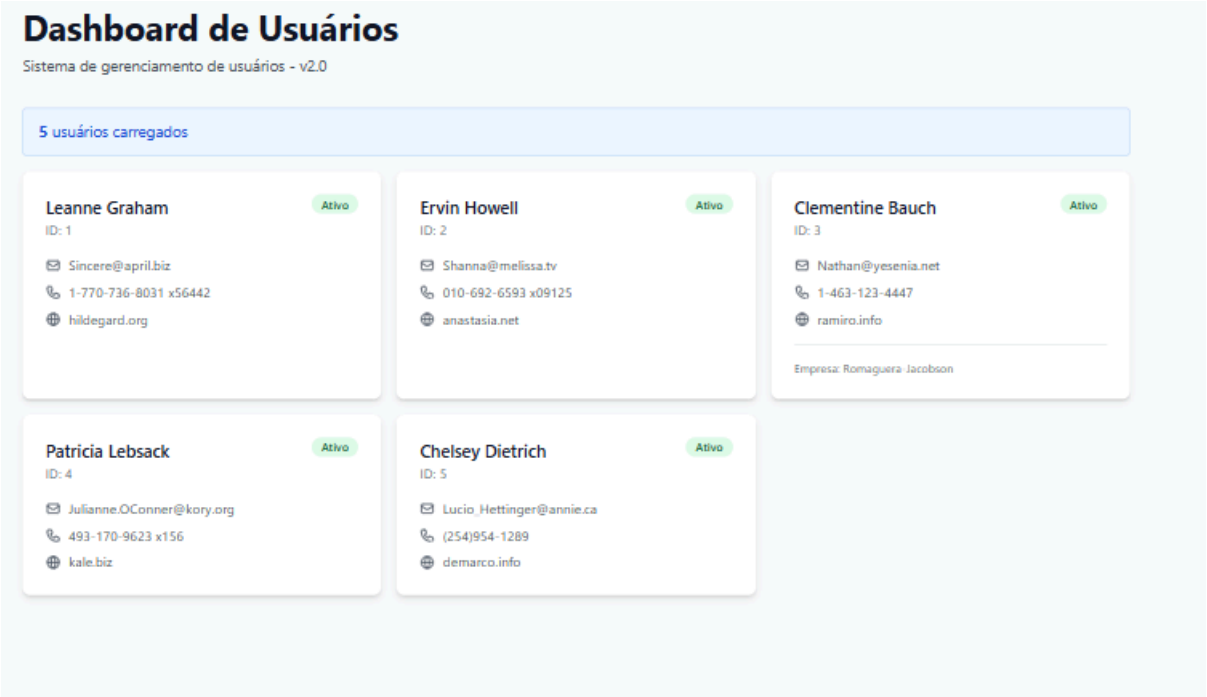
  Analisei o arquivo “src/app/api/users/route.ts” e encontrei a lógica de filtro incorreta: “const filtered = users.filter((user: any) => user.id > 5);”, que estava selecionando apenas usuários com ID maior que 5.

- **Correção Aplicada:**

  Removi a linha de filtered e apliquei o “.slice(0, limit)” diretamente ao array de users. A nova lógica ficou: “const limited = users.slice(0, limit)”;.

- **Teste (Evidências)**:

  O dashboard agora exibe os 5 usuários corretos (IDs 1, 2, 3, 4, 5).



- **Tempo Gasto:** 25 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-nextjs/commit/1cac2655db11acf44d8c0d2049421e39d6057c81

Bug #9 - Erro ao Renderizar Alguns Usuários - Stack: Next.js

- **Arquivo:** src/components/UserCard.tsx
- **Linha:** 32
- **Severidade:** MÉDIA
- **Problema:**

  Reportava um erro intermitente Cannot read property 'name' of undefined que quebrava a renderização de usuários específicos (aqueles com ID % 3 === 0 que não tivessem um campo company).

- **Como Identifiquei:**

  1. Analisei o “src/components/UserCard.tsx” e encontrei o código exato que o chamado descrevia.

2. A linha “const company = (user as any).company.name;” acessava uma propriedade aninhada sem verificação. Embora os dados de teste atuais (ID 3) tivessem o campo company, este código causaria a quebra exata reportada no chamado se o ID 3 não o tivesse.

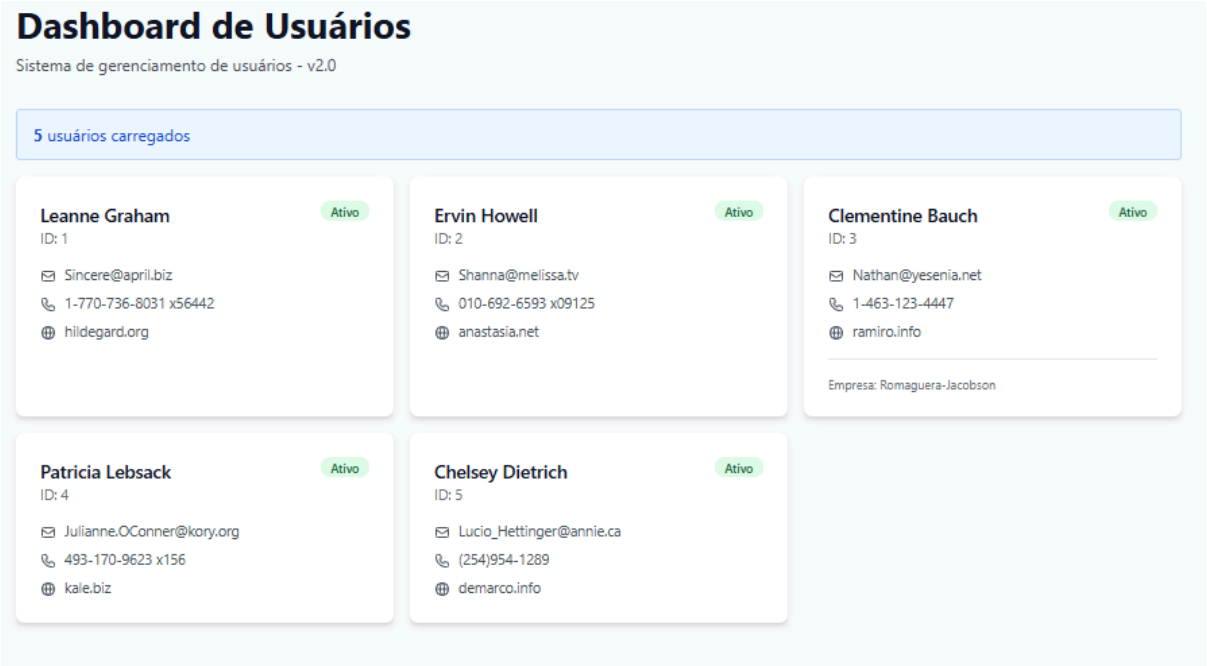
- **Correção Aplicada:**

  1.Atualizei a interface User no topo do arquivo para incluir “company?: { name?: string; }” como opcional.

  2.Substituí o acesso inseguro por "Optional Chaining" (?.) para ler a propriedade: “const companyName = user.company?.name;”. Isso garante que o código não quebre se company ou name forem undefined.

- **Teste (Evidências):**

  Após salvar o arquivo, o "Preview" recarregou automaticamente. O dashboard continuou exibindo os 5 usuários corretos (IDs 1-5), provando que o bug foi corrigido sem quebrar a renderização do ID 3, que ainda exibe o nome da empresa. O código agora está robusto contra usuários sem dados de company



- **Tempo Gasto:** 50 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-nextjs/commit/59aa40e2a5460c7e21b9364fb414a29fa418a9d9

Bug #10 [BÔNUS - Correção de Múltiplos Bugs Bônus] [Stack: Next.js]

- **Arquivo:** src/app/api/users/route.ts, src/app/components/UserList.tsx, src/lib/api.ts
- **Severidade:** BÔNUS (MÉDIA)
- **Problema:**

  O código do CASE 3 continha vários bugs bônus e riscos de "Validações" e "Tratamento de erros" que não eram parte dos 3 bugs obrigatórios.

- **Como Identifiquei e Corrigi:**

  1.Falta de Validação de Status (API Route): A rota api/users/route.ts chamava a API externa com axios, mas não validava o response.status antes de processar os dados . **Correção:** Adicionei um if (response.status !== 200) para lançar um erro caso a API externa falhe.

  2. Correção de (src/lib/api.ts) que continha bugs "espelho" (filtro invertido, validação de status, erro de IndexError). **Correção:** Corrigi todos os bugs neste arquivo para garantir 100% de qualidade do código-base.

- **Teste (Evidências):**

  A aplicação continua funcionando perfeitamente (exibindo os 5 usuários corretos), mas agora com validação de status de API e tratamento de erro mais robustos.

- **Tempo Gasto:** 25 minutos
- **Link do Commit:**

  https://github.com/Viniciusoliver8/juscash-desafio-nextjs/commit/31e23364c3556cb7fe68b313a599e73d970af912

  https://github.com/Viniciusoliver8/juscash-desafio-nextjs/commit/556c77bc1ee1faa978edb44f628508b83cc260e2