Beatriz Auer Mariano, João Marcos Pimentel, Kleber André Picoli, Lara Aguilar de Amorim,

Manoel Rodrigues Loureiro

Desenvolvimento Orientado a Objetos

Relatório do trabalho: Watchman (Nota Fiscal)

O objetivo deste trabalho é desenvolver um *job* que, ao identificar uma nova nota fiscal no Minio, realize a coleta das informações relevantes e salve-as no banco de dados. São consideradas informações relevantes: tipo de compra (serviço, produto e respectiva discriminação), valor, imposto, dados da emitente e do comprador. As notas fiscais devem estar vinculadas aos Projetos do Conecta. Espera-se, ao final do desenvolvimento, que um *dashboard* possa ser desenvolvido para obter um mapa visual de compras e gastos por projeto. É desejável que o sistema seja capaz de validar a nota fiscal recebida. Esse relatório explora os padrões de projetos aplicados, ou não, no projeto, e as justificativas.

REPOSITÓRIOS DO PROJETO

https://github.com/joaomrpimentel/watchman

https://github.com/laraguilar/watchman-front

PADRÕES CRIACIONAIS

Não aplicados.

Justificativa: o próprio modelo de execução e arquitetura do Airflow não favorece ou exige o uso de padrões comportamentais. A arquitetura do Airflow é orientada ao fluxo de dados, não à criação de objetos. Além disso, as *tasks* são simples funções executadas em contêineres isolados. Portanto, não há contexto onde a criação de objetos é complexa o bastante para justificar a aplicação de padrões criacionais.

PADRÕES ESTRUTURAIS

Aplicou-se o padrão **Facade**.

Justificativa: O padrão Facade foi aplicado para fornecer uma interface simples para a *view* do front-end. O Service da API é responsável por lidar com diferentes tipos de respostas, erros de rede, JSON e da API. Porém, a *view* não precisa saber desses detalhes de implementação, apenas chamar a função para salvar a nota fiscal.

Benefícios: Ao aplicar esse padrão, o código do front-end fica isolado da complexidade do serviço, e o uso da API se torna apenas uma chamada de função.

Comparação com código anterior: como não havia código anterior, não há base para comparações.

Os demais padrões comportamentais não foram aplicados neste projeto porque, na mesma linha da justificativa dos padrões criacionais, as *tasks* do projeto no back-end possuem responsabilidades únicas e limitadas, e não compõem estruturas complexas de objetos. Portanto, torna-se desnecessária a aplicação de padrões estruturais voltados à composição, abstração ou adaptação entre classes. O mesmo se aplica para o front-end, que, por enquanto, é apenas um portal de submissão de documentos.

PADRÕES COMPORTAMENTAIS

Aplicou-se o padrão **Strategy**.

Justificativa: O padrão Strategy foi aplicado para encapsular o comportamento de salvamento de arquivos, permitindo que diferentes tipos de documentos tenham sua própria lógica de persistência. Atualmente, existem apenas notas fiscais no formato XML. Porém, a aplicação pode passar a receber novos tipos e formatos de documentos no futuro. Dessa forma, o código principal fica desacoplado da lógica de persistência.

Benefícios: Ao aplicar esse padrão, o projeto ganha flexibilidade e extensibilidade, permitindo a inclusão de novos tipos de documentos com diferentes regras de salvamento sem modificar o código existente. Isso torna o sistema mais modular, aderente ao princípio aberto/fechado e facilita a manutenção e evolução da aplicação conforme surgem novos requisitos.

Comparação com código anterior: como não havia código anterior, não há base para comparações.

Os demais padrões comportamentais não foram aplicados neste projeto porque as interações entre os componentes são diretas e simples, sem necessidade de orquestração complexa de comportamentos ou comunicação entre muitos objetos. Padrões como Observer, Mediator ou Chain of Responsibility não se justificam, pois não há eventos encadeados, múltiplos manipuladores, nem objetos que precisem ser desacoplados em suas interações. Cada componente executa uma tarefa específica com fluxo de controle claro e linear, tornando desnecessária a complexidade adicional que esses padrões trariam.

EVIDÊNCIAS DO FUNCIONAMENTO DO SISTEMA

Figura 1 - Tela inicial do front-end

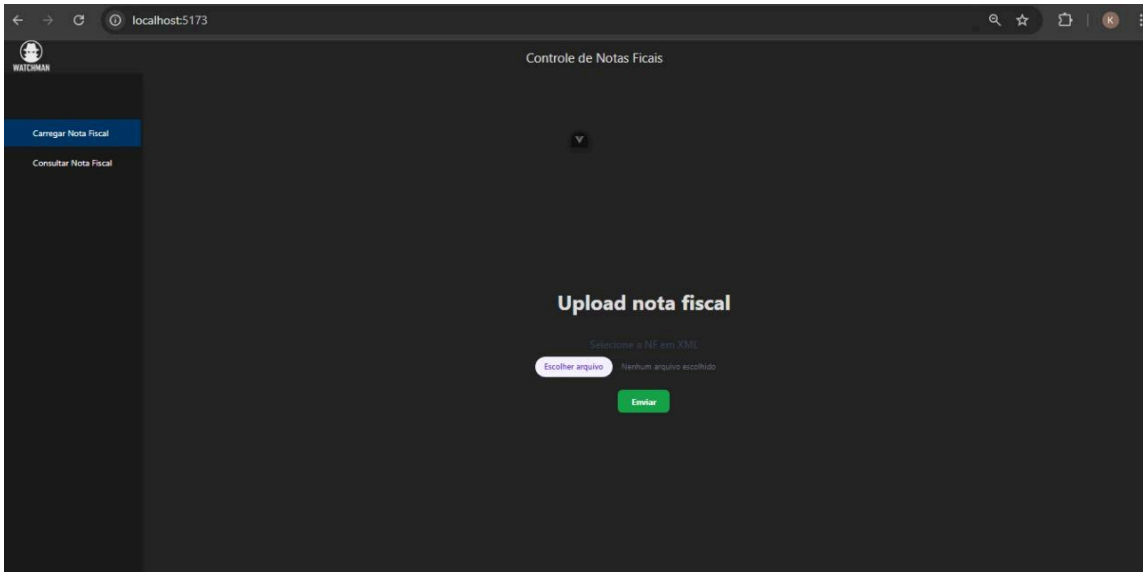


Figura 2 - Seleção de arquivo para *upload*

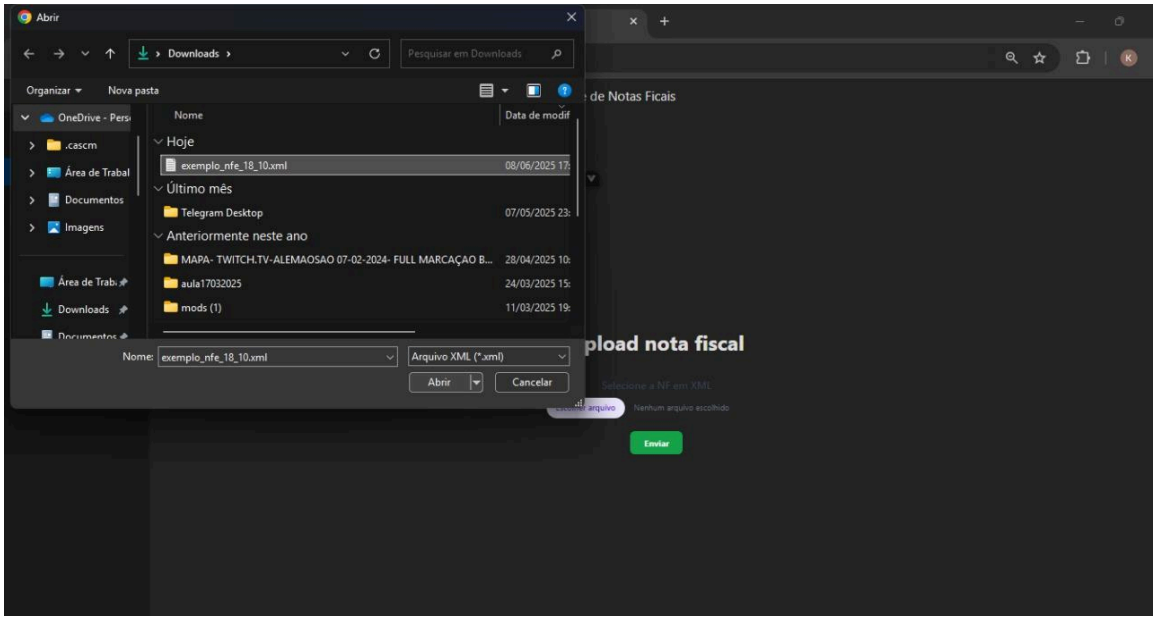


Figura 3 - Mensagem de sucesso no envio do arquivo

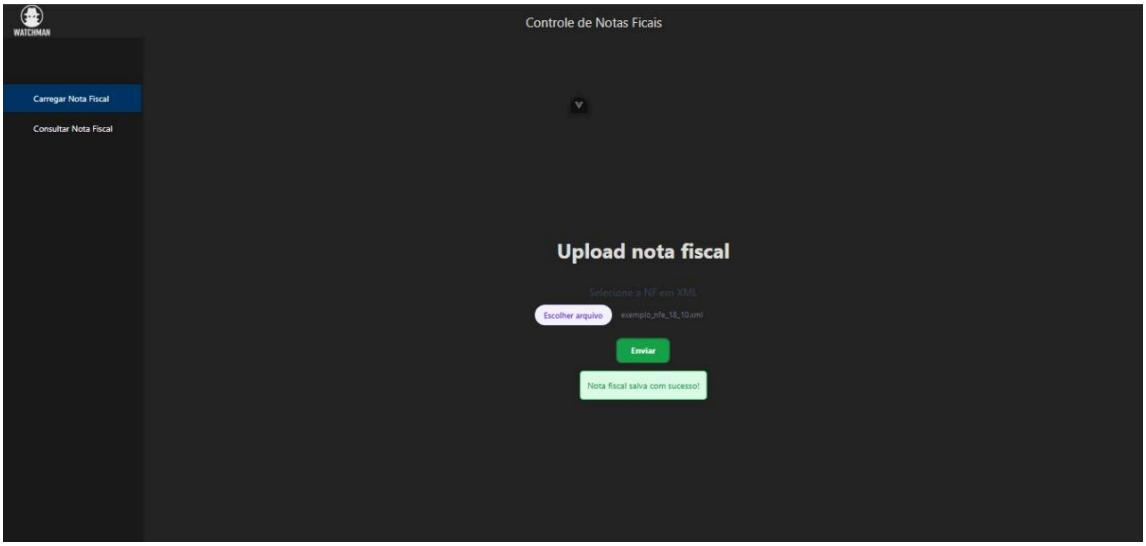
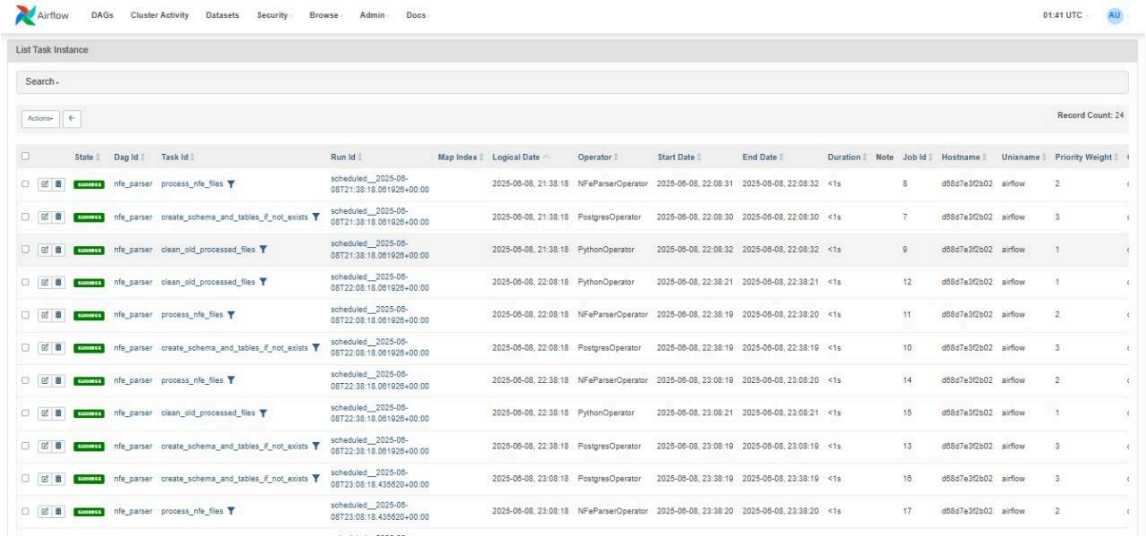


Figura 4 - *Logs* de execução bem sucedidas no Airflow



CONCLUSÃO

Conclui-se que o projeto Watchman atendeu aos objetivos propostos, implementando com sucesso um job que detecta novas notas fiscais no Minio, extrai informações relevantes e as armazena no banco de dados. A aplicação de padrões de projeto, como Facade e Strategy, contribuiu para a organização, modularidade e extensibilidade do sistema, especialmente diante de possíveis evoluções futuras. A ausência de outros padrões foi justificada pela simplicidade e clareza das responsabilidades dos componentes. As evidências apresentadas demonstram o funcionamento correto do sistema, validando a abordagem adotada durante o desenvolvimento.