UNICURITIBA
Ciência da Computação

Gabriel Zaioncz dos Santos
Gabrielle Luisa Horiuchi
Gustavo de Freitas Crepaldi
João Victor Lellis Butzen

RELATÓRIO DA AVALIAÇÃO A3

Curitiba
2025

Gabriel Zaioncz dos Santos - 172416227
Gabrielle Luisa Horiuchi - 172416615
Gustavo de Freitas Crepaldi - 172415709
João Victor Lellis Butzen - 172410826

RELATÓRIO DA AVALIAÇÃO A3

Relatório sobre o trabalho acadêmico apresentado à Universidade UNICURITIBA, como requisito parcial para a avaliação da disciplina Sistemas Distribuídos e Mobile, do curso de Ciência da Computação.

DOCENTE: Prof. Flávio Henrique da Silva.

Curitiba
2025

SUMÁRIO

1 INTRODUÇÃO

No cenário atual de intensificação das transações digitais, os casos de consumidores que não reconhecem compras realizadas com seus cartões têm se tornado cada vez mais frequentes. Essa situação, além de gerar preocupação e insegurança para os clientes, representa um desafio operacional significativo para os bancos, que precisam agir com rapidez e precisão para evitar fraudes, prejuízos e desgaste na relação com o cliente.

Pensando nesse problema, a partir de uma parceria entre a instituição UNICURITIBA e o Banco Bradesco, foi proposto aos alunos o desenvolvimento de soluções tecnológicas para desafios reais enfrentados pelo setor bancário. O desafio escolhido pelo grupo foi: *"*vítima não reconhece uma compra com seu cartão*"*.

Como solução, o grupo desenvolveu uma aplicação com foco na validação de compras por meio da confirmação via e-mail. A aplicação realiza uma verificação em duas etapas, permitindo que o cliente confirme a legitimidade da transação antes de sua efetivação, o que contribui para a prevenção de fraudes e melhora a experiência do usuário.

2 DESENVOLVIMENTO

O presente projeto foi desenvolvido com o objetivo de propor uma solução para um dos desafios propostos pelo Banco Bradesco em parceria com a UNICURITIBA. A problemática selecionada pelo grupo consistiu na situação em que o cliente não reconhece uma compra efetuada com seu cartão. Diante disso, foi idealizada e implementada uma aplicação que realiza a validação de compras em duas etapas, por meio de uma API que utiliza autenticação por *token* e confirmação via e-mail.

O desenvolvimento da aplicação contemplou desde a criação da lógica de funcionamento da API até a elaboração de uma interface para a confirmação da compra, além da documentação técnica do projeto. O documento encontra-se organizado em seções específicas que descrevem a divisão de tarefas entre os membros do grupo, a estrutura técnica do projeto, o funcionamento da aplicação, as instruções para execução e o repositório onde o projeto está hospedado.

2.1 DIVISÃO DAS TAREFAS

A divisão de tarefas entre os integrantes do grupo foi realizada de forma a otimizar as habilidades individuais de cada membro, garantindo uma execução eficiente do projeto. A seguir, apresenta-se a contribuição de cada participante:

João foi o principal responsável pelo desenvolvimento da lógica de funcionamento da API, concentrando-se na criação do arquivo main.py e na configuração das bibliotecas no arquivo requirements.txt. Sua atuação foi fundamental para a construção do *backend* da aplicação.

Gustavo colaborou diretamente com João no desenvolvimento do *backend*, auxiliando na implementação das rotas da API, inclusão da geração de logs e na integração com o serviço de envio de e-mails para confirmação de compra.

Gabrielle ficou encarregada da criação da interface visual do sistema, desenvolvendo a estrutura HTML no arquivo confirma.html e aplicando o design necessário por meio do arquivo de estilos mail.css.

Gabriel foi o responsável pela elaboração do relatório técnico, estruturando o conteúdo do documento de forma clara e objetiva. Sua tarefa consistiu em organizar as informações sobre o projeto para apresentação e avaliação final.

2.2 ESTRUTURA DO PROJETO

A estrutura do projeto foi concebida para oferecer uma solução funcional e segura para a validação de compras com cartão, utilizando confirmação via e-mail. A aplicação foi desenvolvida com a linguagem de programação Python, e seu núcleo técnico baseia-se no framework FastAPI, escolhido pela sua leveza, desempenho e compatibilidade com aplicações assíncronas.

O projeto é composto pelos seguintes arquivos principais:

- main.py: concentra toda a lógica do *backend*, incluindo rotas da API, geração e validação de *tokens*, envio de e-mails e registros de log.
- requirements.txt: contém as dependências essenciais para a execução da aplicação.
- confirma.html: representa a interface visual acessada pelo cliente ao clicar no link de confirmação.
- mail.css: define o estilo da interface HTML enviada por e-mail.

As bibliotecas utilizadas na aplicação incluem:

- FastAPI (framework principal da API).
- Uvicorn (servidor ASGI para execução da aplicação).
- Pydantic (validação e tipagem de dados).
- PyJWT (geração e verificação de *tokens* JWT).
- Cachetools (armazenamento temporário de dados em cache).
- Smtplib e Email (envio de mensagens por protocolo SMTP).
- Logging (registro de logs e mensagens de sistema).
- OS (manipulação de diretórios e arquivos do sistema).

A API disponibiliza cinco rotas principais:

- /solicita_compra: recebe os dados da transação e gera um *token* JWT com validade de uma hora.
- /obter_email/{token}: vincula o e-mail do cliente ao *token* e envia o link de confirmação.
- /valida_compra/{token}: processa a confirmação do cliente e move o *token* para o cache de confirmações válidas.

- /tokens_pendentes: retorna os *tokens* ainda não confirmados.
- /tokens_validos: retorna os *tokens* confirmados com sucesso.

A estrutura adotada visa facilitar a compreensão dos processos, garantir a segurança na manipulação de dados sensíveis e possibilitar a execução eficiente da aplicação.

2.3 EXPLICAÇÃO DA APLICAÇÃO/SOFTWARE

A aplicação desenvolvida consiste em uma API que realiza a validação de compras com cartão por meio de um processo em duas etapas. Inicialmente, a API recebe os dados da compra — incluindo o número do cartão, o valor e o número de parcelas — por meio de uma das cinco rotas principais implementadas.

Após o recebimento dessas informações, a aplicação processa a compra e gera um *token* exclusivo de validação vinculado ao cliente. Em seguida, a API solicita o e-mail do titular do cartão e envia uma mensagem com um link de confirmação. Esse link permite que o cliente valide ou conteste a transação realizada.

Quando o cliente acessa o link de confirmação enviado por e-mail, a API atualiza os dados da transação com base na resposta do usuário. Caso a confirmação não ocorra dentro de um período máximo de uma hora, o *token* expira automaticamente, invalidando o processo de validação e, consequentemente, impedindo a conclusão da compra.

Por fim, os dados validados — ou o status da transação — são disponibilizados ao banco por meio de uma rota específica. Esse fluxo visa aumentar a segurança nas transações e garantir maior controle por parte do cliente em casos de compras não reconhecidas.

2.4 ORIENTAÇÕES DE EXECUÇÃO DA APLICAÇÃO/SOFTWARE

Para executar a aplicação, o primeiro passo consiste em salvar todos os arquivos do repositório em uma pasta separada. Em seguida, é necessário criar um ambiente virtual utilizando o comando 'python -m venv api'.

Após a criação do ambiente, deve-se ativá-lo no terminal: no sistema operacional Windows, utiliza-se o comando 'api\Scripts\activate'; em sistemas Linux ou macOS, o comando correspondente é 'source api/bin/activate'.

Com o ambiente virtual ativado, o próximo passo é instalar as dependências da aplicação utilizando o comando 'pip install -r requirements.txt'.

Finalizada a instalação, a API pode ser inicializada com o comando 'uvicorn main:app --host 0.0.0.0 --port 5000 --reload', sendo possível modificar a porta conforme necessário.

2.5 REPOSITÓRIO

- https://github.com/JVLB06/API_salva_cartao

3 CONCLUSÃO

O presente trabalho teve como objetivo propor uma solução tecnológica eficaz para o problema de compras não reconhecidas por clientes, conforme desafio proposto pelo Banco Bradesco em parceria com a UNICURITIBA. A aplicação desenvolvida demonstrou viabilidade prática ao permitir que o cliente valide, por meio de confirmação via e-mail, transações realizadas com seu cartão, promovendo maior segurança e transparência no processo.

Durante o desenvolvimento do projeto, a equipe enfrentou desafios relacionados ao gerenciamento do tempo de expiração dos tokens e à integração assíncrona das rotas da API. A resolução dessas dificuldades exigiu pesquisa, testes e revisão conjunta do código, contribuindo para o aprendizado coletivo dos integrantes.

A utilização de tecnologias como FastAPI, tokens JWT, armazenamento temporário em cache e envio automatizado de e-mails permitiu a construção de uma API robusta, funcional e de fácil manutenção. Além disso, a divisão clara de tarefas entre os integrantes do grupo contribuiu para o cumprimento dos prazos e para o desenvolvimento colaborativo da solução.

Acredita-se que, com adaptações e integração a sistemas bancários reais, a aplicação pode oferecer uma contribuição relevante para a prevenção de fraudes e para a melhoria da experiência do cliente em instituições financeiras.

REFERÊNCIAS

CACHETOOLS. *Cachetools – Extensible memoizing collections and decorators*. Disponível em: https://cachetools.readthedocs.io/en/stable/. Acesso em: 05 jun. 2025.

FASTAPI. *FastAPI: modern, fast (high-performance) web framework for building APIs with Python 3.6+ based on standard Python type hints*. Disponível em: https://fastapi.tiangolo.com/. Acesso em: 05 jun. 2025.

PYDANTIC. *Data validation and settings management using Python type annotations*. Disponível em: https://docs.pydantic.dev/. Acesso em: 05 jun. 2025.

PYJWT. *PyJWT documentation*. Disponível em: https://pyjwt.readthedocs.io/en/latest/. Acesso em: 05 jun. 2025.

PYTHON SOFTWARE FOUNDATION. *email — An email and MIME handling package*. Disponível em: https://docs.python.org/3/library/email.html. Acesso em: 05 jun. 2025.

PYTHON SOFTWARE FOUNDATION. *logging — Logging facility for Python*. Disponível em: https://docs.python.org/3/library/logging.html. Acesso em: 05 jun. 2025.

PYTHON SOFTWARE FOUNDATION. *os — Miscellaneous operating system interfaces*. Disponível em: https://docs.python.org/3/library/os.html. Acesso em: 05 jun. 2025.

PYTHON SOFTWARE FOUNDATION. *smtplib — SMTP protocol client*. Disponível em: https://docs.python.org/3/library/smtplib.html. Acesso em: 05 jun. 2025.

PYTHON SOFTWARE FOUNDATION. *Welcome to Python.org*. Disponível em: https://www.python.org/. Acesso em: 05 jun. 2025.

UVICORN. *Uvicorn: lightning-fast ASGI server implementation, using uvloop and httptools*. Disponível em: https://www.uvicorn.org/. Acesso em: 05 jun. 2025.