Fundação Escola de Comércio Álvares Penteado

Ciência da Computação 4º Semestre

Aplicação de Design de Software e Diagramas UML

Gabriel Henrique Coelho Marussi - 24026609
Lucas Kenichi Soares – 24026179
Felipe Oluwaseun Santos Ojo - 24026245
Arthur Rodrigues Ferreira – 24026567
Pedro Dimitry Zyrianoff – 24026165

São Paulo – 2025

SUMÁRIO

1. Introdução

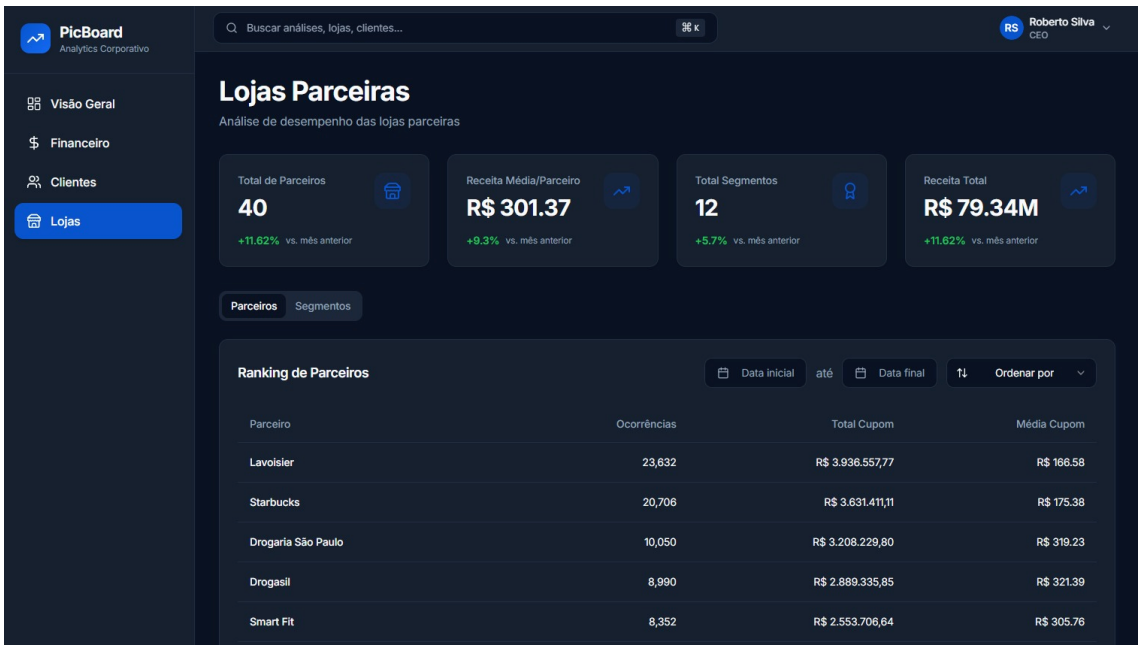
O presente documento detalha o desenvolvimento do **Dashboard da PicMoney**, uma aplicação estratégica desenvolvida para fornecer aos gestores executivos (CEO e CFO) uma plataforma visual e analítica para a tomada de decisões. O projeto foi guiado pela aplicação rigorosa de princípios de Design de Software para garantir a construção de um sistema modular, escalável e de fácil manutenção.

2. Aplicações de Design de Software em nosso Projeto

Durante o desenvolvimento do Dashboard da PicMoney, foram aplicados diversos princípios de Design de Software para garantir um sistema modular, escalável e de fácil manutenção. O projeto foi estruturado com base em uma arquitetura distribuída, integrando diferentes linguagens e tecnologias modernas como React, Node.js, Python, JavaScript e AWS.


PicBoard
Analytics Corporativo
Buscar análises, lojas, clientes...
⌘K
RS
Roberto Silva
CEO
Visão Geral
Financeiro
Clientes
Lojas
Clientes
Análise de comportamento e métricas de clientes
Total de Usuários
9,011
+119.08% vs. mês anterior
Ticket Médio
R$ 267.77
+7.8% vs. mês anterior
Taxa de Retenção
47.5%
+3.2% vs. mês anterior
Usuários Ativos
6,018
+119.08% vs. mês anterior
Visão Geral
Localização
Usuários por Zona
Distribuição de usuários por região (ordenado por quantidade)
Usuários
2800
2100
1400
700
0
2704
2687
2681
2653
2650
Centro
Oeste
Leste
Sul
Norte
Top 10 Bairros por Usuários
Clique nas abas para alternar entre residencial, trabalho e escola
Residencial
Trabalho
Escola
Usuários
1000
750
500
250
0
968
943
936
928
922
905
859
853
20
20
Liberdade
Pinheiros
Vila Mariana
Jardins
Tatuapé
Santana
Bela Vista
Moema
Higienópolis
Penha


PicBoard
Analytics Corporativo
Buscar análises, lojas, clientes...
⌘K
RS
Roberto Silva
CEO
Visão Geral
Financeiro
Clientes
Lojas
Dashboard Executivo
Análise completa de performance e métricas financeiras
Receita Total
R$ 79.34M
+11.62% vs. mês anterior
Receita Líquida
R$ 69.15M
+11.62% vs. mês anterior
Margem Operacional
87.2%
+2.3% vs. mês anterior
Total de Usuários
9,011
+119.08% vs. mês anterior
Visão Geral
Análise Temporal
Segmentos
Receita por Segmento
Clique na legenda para mostrar/ocultar séries
R$
20000k
R$ 15000k
R$ 10000k
R$ 5000k
R$ 0k
Restaurante
Farmácia
Moda
Supermercado & Conveniência
Esporte & Fitness
Saúde
Clube | Cultura & Esporte
Eletro & Móveis
Cafeteria
Papelaria
Receita
Cupons



2.1 Arquitetura e Estrutura do Sistema

A aplicação seguiu o padrão MVC (Model-View-Controller):

- Frontend (View): desenvolvido em React, proporcionando uma interface moderna, dinâmica e responsiva, com componentes reutilizáveis e atualizações em tempo real.
- Backend (Controller): construído em Node.js com JavaScript, responsável pelas rotas, autenticação e comunicação com o banco de dados.
- Camada de dados (Model): implementada em Python, responsável por análise de dados e geração de relatórios automatizados.

Os serviços foram hospedados na AWS (Amazon Web Services), garantindo disponibilidade, segurança e escalabilidade da aplicação.

2.2 Princípios e Boas Práticas de Design

O desenvolvimento seguiu os princípios SOLID e boas práticas de programação orientada a objetos:

- Responsabilidade única: cada módulo cumpre uma função específica (ex.: autenticação, relatórios, análise).
- Reuso e extensibilidade: o sistema permite adicionar novas métricas e gráficos sem alterar a base existente.
- Modularidade: o código foi dividido em componentes e APIs independentes, facilitando o trabalho colaborativo entre as equipes.

2.3 Padrões de Projeto Utilizados

Para garantir um código limpo e de fácil evolução, foram aplicados padrões de projeto como:

- MVC, para separação entre interface, controle e lógica.
- Observer, usado em React para atualização automática dos componentes do dashboard.
- DAO (Data Access Object), para abstração do acesso aos dados.
- Singleton, garantindo uma única instância da conexão com os serviços da AWS.

2.4 Integração e Deploy

A AWS foi utilizada para hospedar tanto o frontend quanto o backend, além de armazenar dados e relatórios. Essa infraestrutura em nuvem trouxe:

- Escalabilidade automática conforme o número de acessos.
- Alta disponibilidade e segurança das informações.
- Facilidade de integração contínua (CI/CD) para novas versões do sistema.

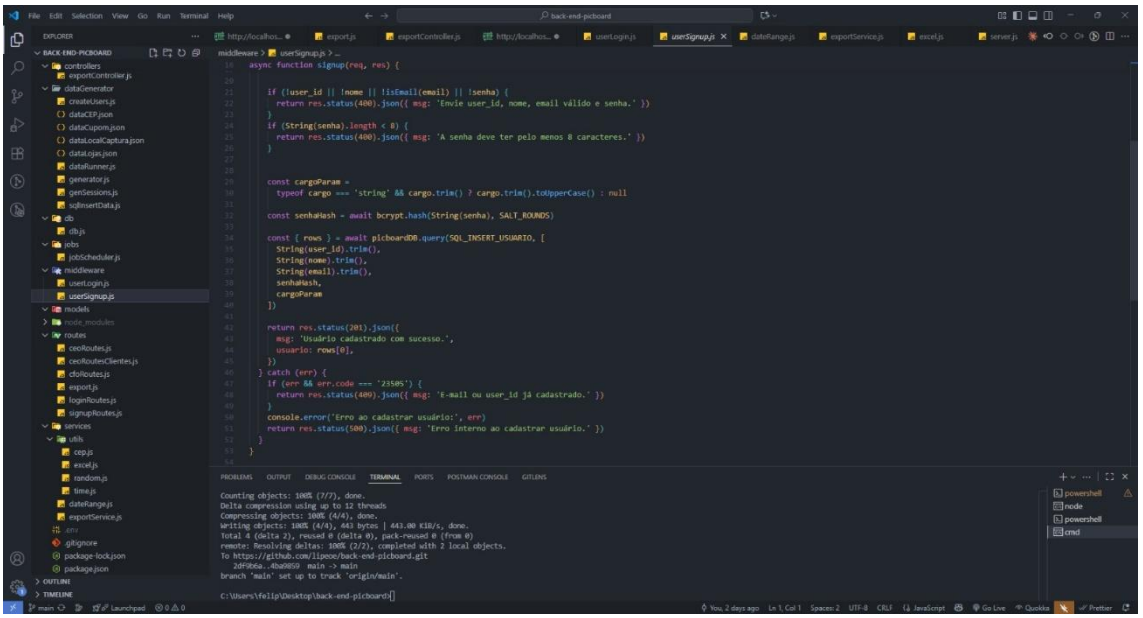
2.5 Resultados e Benefícios

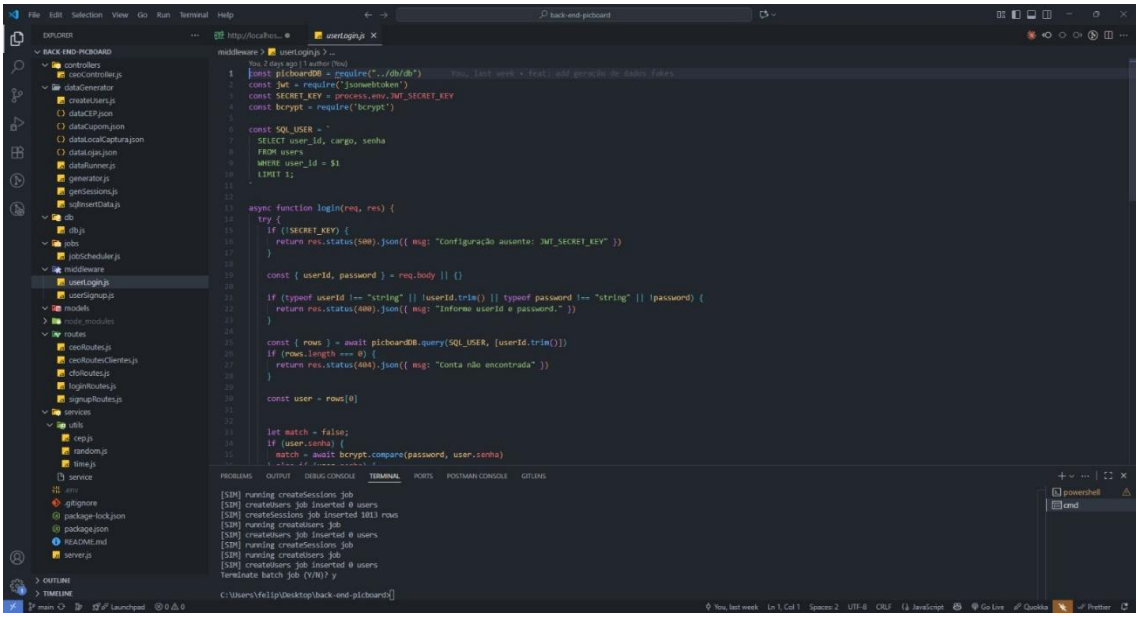
A aplicação de design de software resultou em um sistema:

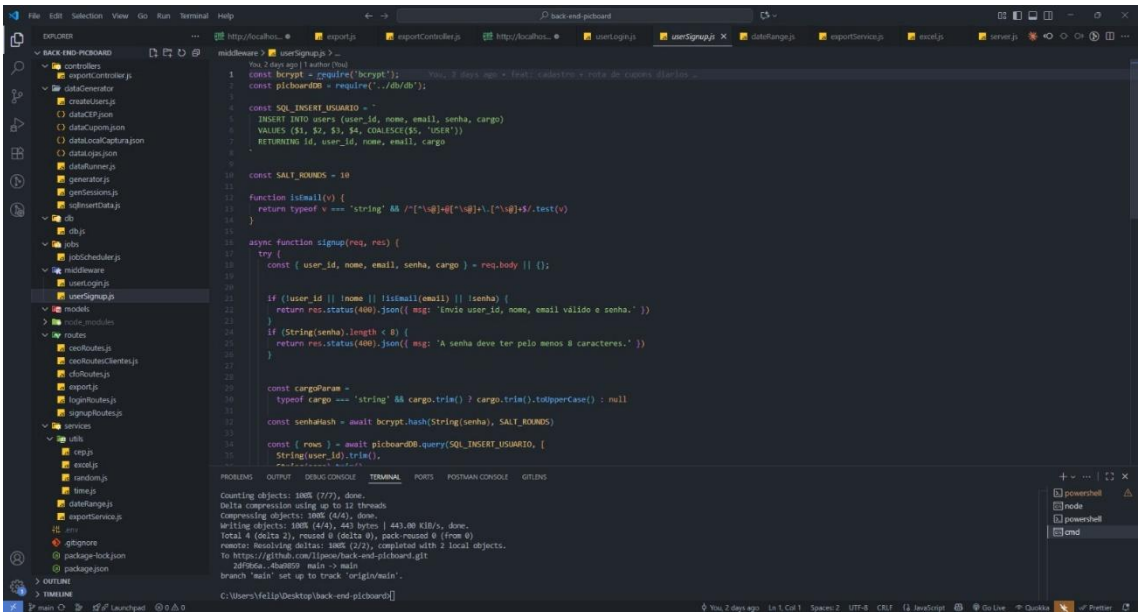
- Eficiente e modular, com código reutilizável e organizado;
- Rápido e responsivo, graças ao React e Node.js;
- Seguro e escalável, por meio da infraestrutura AWS;
- Flexível, permitindo integração futura com inteligência artificial para previsões e recomendações financeiras.

3. Arquitetura em Camadas do Sistema PicMoney Dashboard

O Dashboard PicMoney foi desenvolvido com base em uma arquitetura em camadas, que separa de forma organizada as responsabilidades do sistema, garantindo eficiência, segurança e facilidade de manutenção.



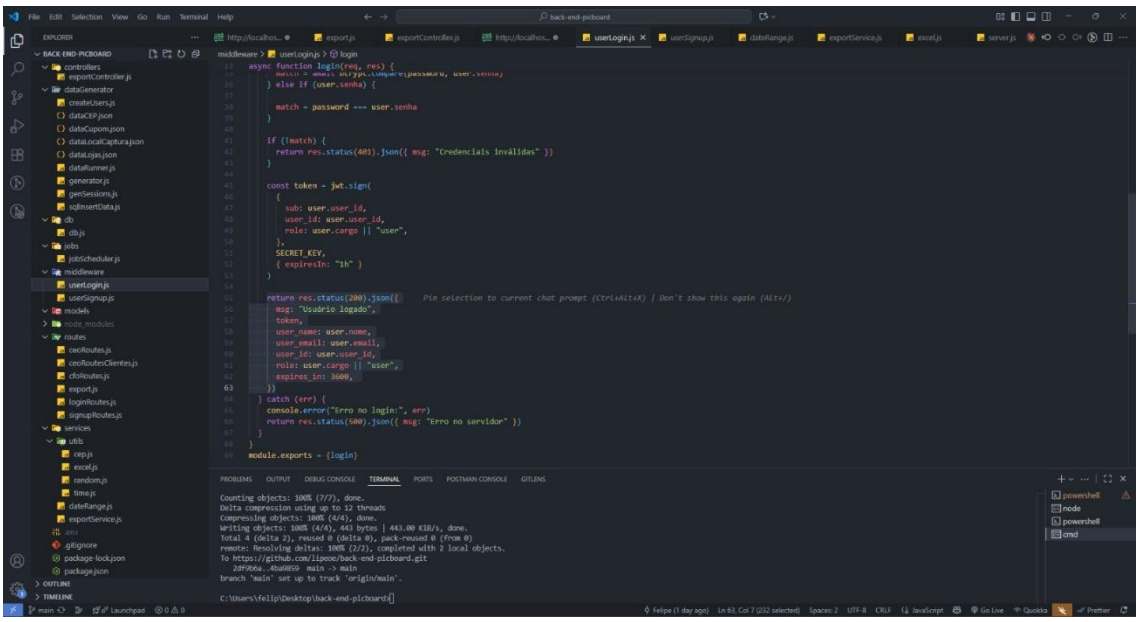




```
middleware > userSignup.js > ...
  You, 2 days ago | 1 author (You)
1 const bcrypt = require('bcrypt');        You, 2 days ago • feat: cadastro + rota de cupons diarios ...
2 const picboardDB = require('../db/db');
3
4 const SQL_INSERT_USUARIO = `
5   INSERT INTO users (user_id, nome, email, senha, cargo)
6   VALUES ($1, $2, $3, $4, COALESCE($5, 'USER'))
7   RETURNING id, user_id, nome, email, cargo
8 `
9
10 const SALT_ROUNDS = 10
11
12 function isEmail(v) {
13   return typeof v === 'string' && /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(v)
14 }
15
16 async function signup(req, res) {
17   try {
18     const { user_id, nome, email, senha, cargo } = req.body || {};
19
20
21     if (!user_id || !nome || !isEmail(email) || !senha) {
22       return res.status(400).json({ msg: 'Envie user_id, nome, email válido e senha.' })
23     }
24     if (String(senha).length < 8) {
25       return res.status(400).json({ msg: 'A senha deve ter pelo menos 8 caracteres.' })
26     }
27
28
29     const cargoParam =
30       typeof cargo === 'string' && cargo.trim() ? cargo.trim().toUpperCase() : null
31
32     const senhaHash = await bcrypt.hash(String(senha), SALT_ROUNDS)
33
34     const { rows } = await picboardDB.query(SQL_INSERT_USUARIO, [
35       String(user_id).trim(),
```

```
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 443 bytes | 443.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/lipeoo/back-end-picboard.git
   2df9b6a..4ba0859  main -> main
branch 'main' set up to track 'origin/main'.

C:\Users\felip\Desktop\back-end-picboard>
```



```
middleware > userLogin.js > login
13 async function login(req, res) {
14     match = await bcrypt.compare(password, user.senha)
36     } else if (user.senha) {
37
38       match = password === user.senha
39     }
40
41     if (!match) {
42       return res.status(401).json({ msg: "Credenciais inválidas" })
43     }
44
45     const token = jwt.sign(
46       {
47         sub: user.user_id,
48         user_id: user.user_id,
49         role: user.cargo || "user",
50       },
51       SECRET_KEY,
52       { expiresIn: "1h" }
53     )
54
55     return res.status(200).json({     Pin selection to current chat prompt (Ctrl+Alt+X) | Don't show this again (Alt+/)
56       msg: "Usuário logado",
57       token,
58       user_name: user.nome,
59       user_email: user.email,
60       user_id: user.user_id,
61       role: user.cargo || "user",
62       expires_in: 3600,
63     })
64   } catch (err) {
65     console.error("Erro no login:", err)
66     return res.status(500).json({ msg: "Erro no servidor" })
67   }
68 }
69 module.exports = {login}
```

```
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 443 bytes | 443.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/lipeoo/back-end-picboard.git
   2df9b6a..4ba0859  main -> main
branch 'main' set up to track 'origin/main'.

C:\Users\felip\Desktop\back-end-picboard>
```

3.1 Camada de Apresentação (Frontend)

A camada de apresentação foi construída em React, responsável pela interface visual acessada pelos gestores da PicMoney, como o CEO e o CFO.
Essa camada exibe gráficos interativos e painéis dinâmicos que permitem:

- Visualizar tendências financeiras e indicadores de desempenho da empresa;
- Filtrar resultados por período, região e categoria de investimento;
- Acompanhar métricas em tempo real de receita, cashback, clientes ativos e volume de transações.

3.2 Camada de Lógica de Negócio (Backend)

A camada intermediária, desenvolvida em Node.js e Python, concentra toda a lógica de negócio e regras operacionais.
Ela é responsável por:

Processar as requisições vindas do dashboard;

Calcular indicadores e gerar relatórios automáticos;

Aplicar filtros inteligentes e análises comparativas de desempenho;

Comunicar-se com a base de dados de forma segura e eficiente.

Essa separação permite que o backend funcione como um núcleo inteligente, capaz de realizar cálculos complexos e análises avançadas sem comprometer a performance da interface.

3.3 Camada de Dados (Database e Integração em Nuvem)

A camada de dados foi estruturada com banco relacional (PostgreSQL) e hospedada na AWS.
Essa camada é responsável por:

Armazenar informações de usuários, relatórios, cupons e desempenho;

Garantir integridade e segurança por meio de autenticação e criptografia;

Permitir escalabilidade automática conforme o volume de dados cresce.

A integração com a nuvem AWS também oferece backups automáticos, monitoramento e alta disponibilidade, assegurando que os dados financeiros da PicMoney estejam sempre acessíveis e protegidos.

3.4 Valor Estratégico para o CFO e o CEO

A arquitetura em camadas foi projetada especialmente para atender as demandas executivas da PicMoney.

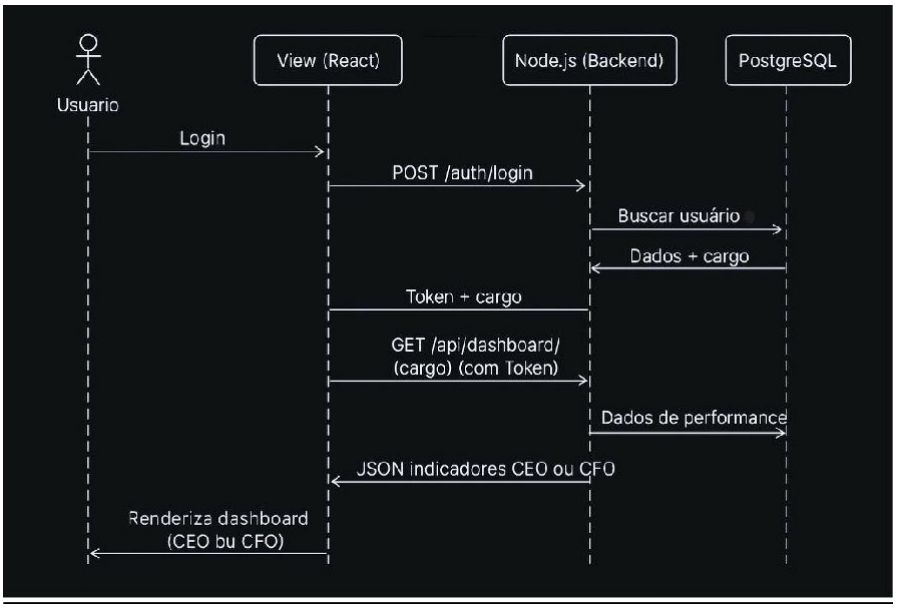
O CFO (Diretor Financeiro) utiliza o dashboard para analisar fluxos de caixa, margens de lucro e desempenho por categoria, com gráficos de barras e linhas comparando diferentes períodos.

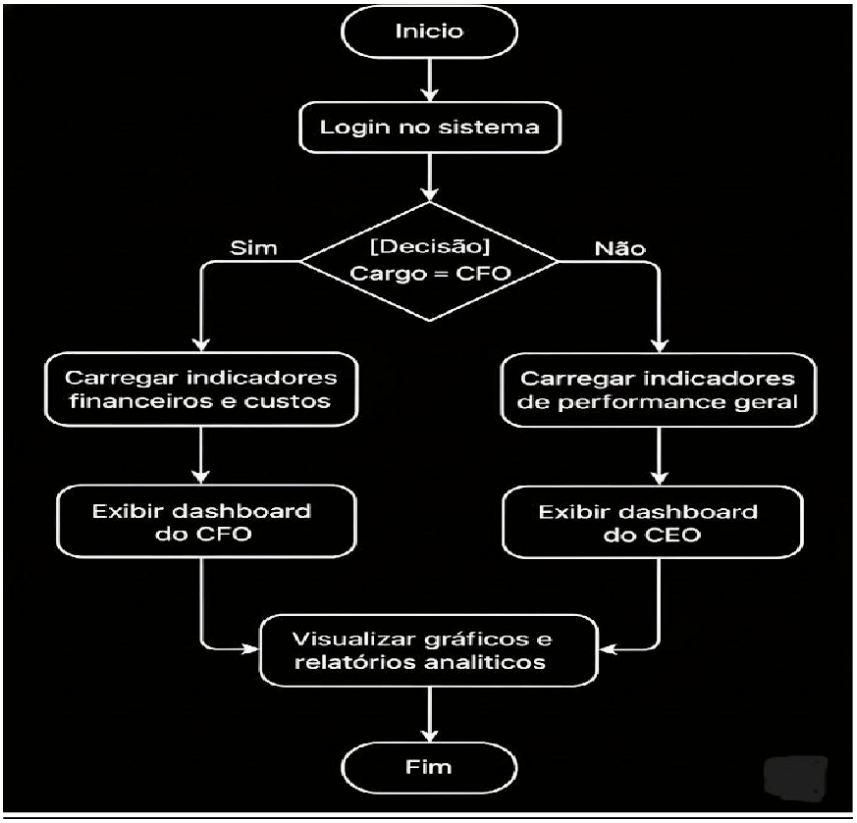
O CEO (Diretor Executivo) tem acesso a visões consolidadas de crescimento, metas atingidas e evolução da base de clientes, por meio de gráficos de KPI e dashboards executivos.

Essas visualizações estratégicas permitem tomadas de decisão rápidas e baseadas em dados, transformando o dashboard em uma ferramenta de gestão e planejamento corporativo.

4. Diagramas de UML

Diagrama de Fluxo do Usuário no Site da PicMoney





4.1 Estrutura geral e classes principais

O sistema foi dividido em entidades funcionais, cada uma representando uma parte essencial do fluxo do usuário:

- Classe Usuario: representa o cliente ou gestor que acessa o sistema. Contém atributos como idUsuario, nome, email, senha e local. Possui métodos como login(), visualizarCupons() e acessarDashboard(), que iniciam o fluxo de navegação.
- Classe Dashboard: é o centro do sistema, responsável por exibir gráficos e métricas. Possui métodos como exibirGraficos(), atualizarDados() e filtrarPorPeriodo(), que permitem ao usuário interagir com os dados.
- Classe Relatorio: representa os relatórios financeiros e de desempenho. Contém métodos como gerarPdf(), exportarRelatorio() e analisarDadosDoDashboard().
- Classe Cupom: armazena informações de cashback e promoções, com métodos para calcularTotalPorUsuario() e filtrarPorTipo().
- Classe Estabelecimento: vincula os cupons a empresas parceiras, com métodos de listagem e cálculo do total de cupons.
- Classe Administrador: herda de Usuario e possui funções adicionais, como gerarRelatorios(), gerenciarUsuarios() e visualizarDashboardGeral().

4.2 Fluxo do usuário no sistema

O fluxo de navegação do usuário segue uma sequência lógica representada pelas relações entre as classes:

1. O usuário acessa o site e executa o método login(), validando suas credenciais.
2. Após a autenticação, ele é direcionado ao Dashboard, onde pode visualizar e interagir com os gráficos.
3. Dentro do dashboard, o usuário pode:
    - Filtrar dados por período, região ou tipo de transação;
    - Gerar relatórios detalhados a partir dos dados do sistema;
    - Visualizar cupons disponíveis e seus respectivos estabelecimentos.
4. O Administrador, por sua vez, tem acesso ampliado ao dashboard e pode gerenciar usuários, atualizar dados e emitir relatórios consolidados.
5. O Relatório é gerado automaticamente e pode ser exportado em PDF para análise pelo CFO e CEO da empresa.

4.3 Relacionamentos e multiplicidades

O diagrama mostra relações bem definidas entre as classes:

- Um usuário pode ter vários cupons (1..n), e cada cupom pertence a um único usuário.
- Um usuário pode acessar múltiplos dashboards, cada um contendo vários relatórios.
- Cada relatório está ligado a apenas um dashboard, garantindo integridade das análises.
- Um estabelecimento pode oferecer vários cupons, mas cada cupom pertence a um único estabelecimento.
- O Administrador é uma subclasse de Usuario, herdando seus atributos e métodos, mas com permissões adicionais.

4.4 Boas práticas de design aplicadas

O diagrama foi construído aplicando conceitos de coesão e baixo acoplamento, mantendo cada classe com uma responsabilidade clara.
Foram utilizados princípios do SOLID e boas práticas de POO, como:

- Herança: o Administrador estende o comportamento do Usuario.
- Encapsulamento: atributos privados e métodos públicos de acesso.
- Reuso e modularidade: as classes Cupom e Estabelecimento podem ser reutilizadas em outros módulos do sistema.

5. Conclusão

Esse diagrama garante uma visão clara da estrutura interna do sistema, facilitando a comunicação entre desenvolvedores e stakeholders.
Além disso, ele orienta o desenvolvimento das funcionalidades voltadas para o CFO e o CEO, como geração de relatórios, comparação de desempenho e visualização de indicadores financeiros estratégicos.

O diagrama de classes UML do PicMoney descreve de forma organizada o fluxo do usuário dentro do sistema, conectando as ações de login, navegação no dashboard, geração de relatórios e interação com cupons e estabelecimentos.
Essa estrutura orientada a objetos permitiu criar um site modular, seguro e escalável, garantindo que o CEO e o CFO tenham acesso a informações precisas e atualizadas em um ambiente visual intuitivo e profissional.