Engenharia de Software e Arquitetura de Sistemas

Design de Software

Introdução

Esta entrega apresenta a aplicação dos princípios de design de software e a modelagem UML do Dashboard Interativo desenvolvido para a PicMoney. O sistema fornece uma plataforma de análise de dados estratégicos, operacionais e financeiros para executivos C-Level (CEO, CFO, CTO).

Princípios – Design de Software

Princípios fundamentais:

Coesão - Cada componente do sistema possui uma responsabilidade bem definida:

- DashboardController: Gerencia exclusivamente a lógica de apresentação dos KPIs
- AlertService: Cuida exclusivamente da geração e notificação de alertas
- AuthenticationService: Gerencia autenticação e permissões de usuários

Acoplamento - Minimizamos as dependências entre módulos através de:

- Injeção de Dependência: componentes recebem suas dependências externamente
- Interfaces: uso de contratos bem definidos entre camadas
- API REST: comunicação entre frontend e backend via endpoints padronizados

Encapsulamento - Proteção de dados internos implementada em:

- Atributos privados nas classes de modelo (User, KPI, Report)
- Métodos getter/setter para acesso controlado aos dados
- Validação de dados na camada de serviço antes de persistência

Reuso - Componentes reutilizáveis criados para evitar duplicação:

- ChartComponent: Componente genérico de visualização de gráficos

- FilterComponent: Filtros dinâmicos aplicáveis a diferentes visualizações
- ExportService: Serviço genérico de exportação (PDF/Excel) utilizado em múltiplos contextos

Simplicidade – Código Limpo e Direto

- Nomenclatura clara e descritiva
- Funções pequenas com responsabilidade única
- Comentários apenas quando necessário para lógica complexa

Princípios SOLID:

- S – Single Responsibility Principle: No Dashboard PicMoney, cada classe possui apenas uma responsabilidade bem definida. A classe UserService gerencia exclusivamente operações de usuários, KPICalculator calcula indicadores de desempenho, ReportGenerator gera relatórios exportáveis, e AlertDetector detecta anomalias nos dados. Esta separação clara facilita a manutenção e torna o sistema mais resiliente a mudanças.
- O – Open/Closed Principle: O sistema está aberto para extensão, mas fechado para modificação. Utilizamos o Strategy Pattern para cálculo de KPIs, onde todas as classes implementam a interface IKPICalculator. Novos tipos de indicadores podem ser adicionados criando novas classes sem alterar o código existente.
- L – Liskov Substitution Principle: Os diferentes perfis executivos (CEO, CFO, CTO) herdam da classe base ExecutiveUser e podem substituí-la sem quebrar o sistema. Todos os executivos podem fazer login e visualizar o dashboard, mas cada um acessa KPIs específicos ao seu perfil, mantendo o comportamento esperado do tipo base.
- I - Interface Segregation Principle: Criamos interfaces específicas e coesas ao invés de genéricas. IDataExporter contém apenas métodos de exportação, IAlertNotifier apenas notificações, e IDataFilter apenas filtragem. Esta abordagem evita forçar classes a implementar métodos não utilizados, resultando em código mais limpo e manutenível.
- D - Dependency Inversion Principle: Os módulos de alto nível dependem de abstrações, não de implementações concretas. O DashboardController depende da interface IKPIService, não da classe KPIServiceImpl. Esta inversão facilita testes unitários com mocks, permite substituição de implementações e aumenta a flexibilidade do sistema.

Modelo de Design

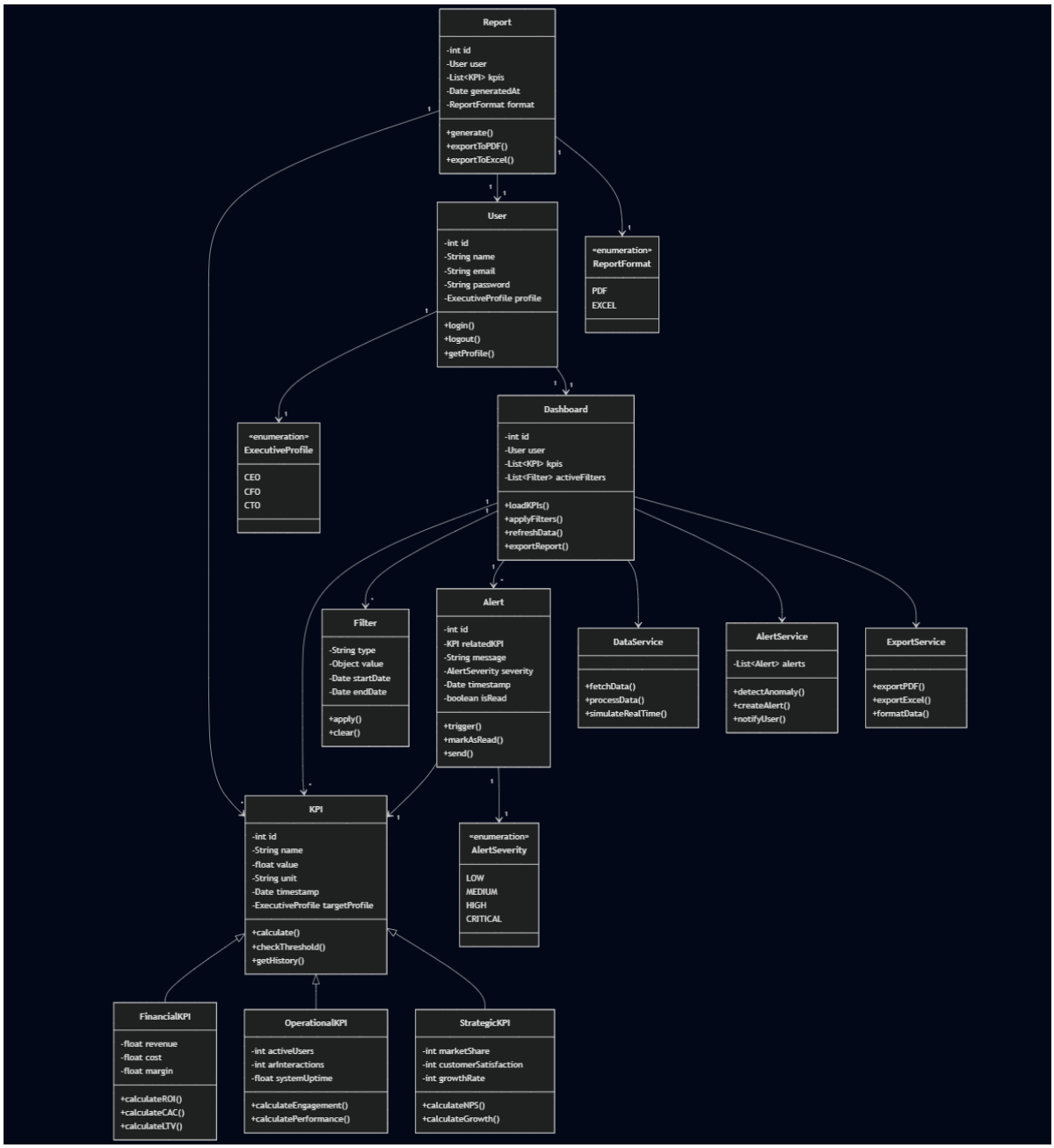
Arquitetura em Camadas

A Arquitetura em Camadas foi escolhida por proporcionar separação clara de responsabilidades, facilitando a manutenção e permitindo que alterações em uma camada não impactem as demais. A testabilidade é melhorada significativamente, pois cada camada pode ser testada isoladamente. Além disso, a arquitetura permite escalabilidade independente conforme a demanda e sendo um padrão amplamente conhecido pela equipe e documentado na literatura de engenharia de software.

Diagramas UML

Diagrama de Classes

O diagrama de classes representa a estrutura estática do sistema, mostrando as principais classes, atributos, métodos e relacionamentos.

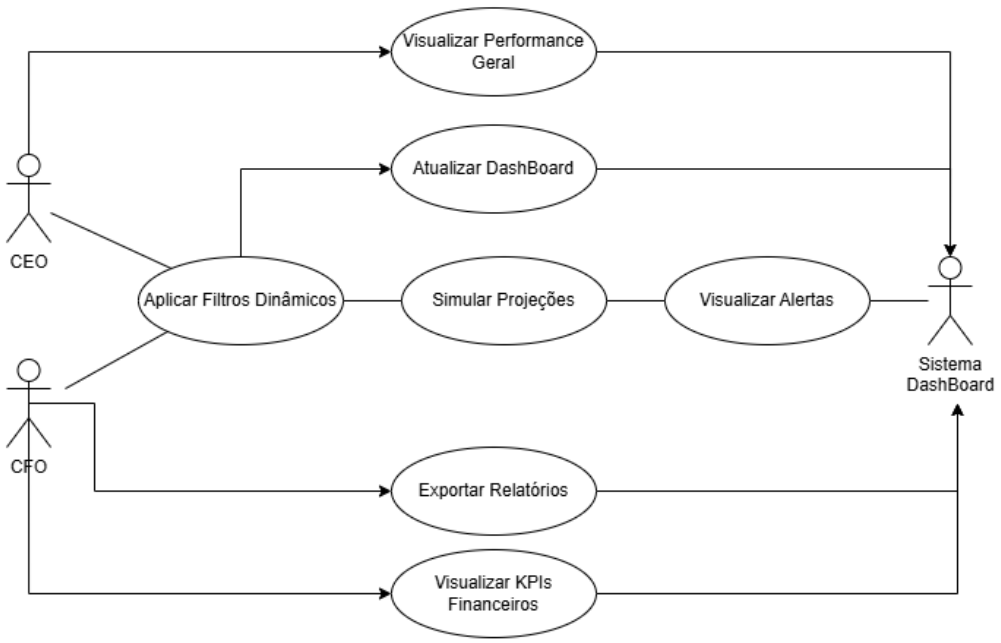


Descrição do Diagrama de Classes:

- **User:** Representa quem usa o sistema.
- **Dashboard:** Classe central que gerencia a visualização e interação.
- **KPI:** Classe abstrata base para todos os indicadores.
- **FinancialKPI / OperationalKPI / StrategicKPI:** Especializações de KPI por perfil.
- **Filter:** Implementa os filtros dinâmicos do sistema.
- **Alert:** Gerencia alertas inteligentes e notificações.
- **Report:** Responsável pela geração e exportação de relatórios.
- **Services:** Camada de serviços para lógica de negócio.

Diagrama de Caso de Uso

O diagrama de casos de uso representa as principais interações entre os atores (CEO, CFO, CTO) e o sistema Dashboard PicMoney.

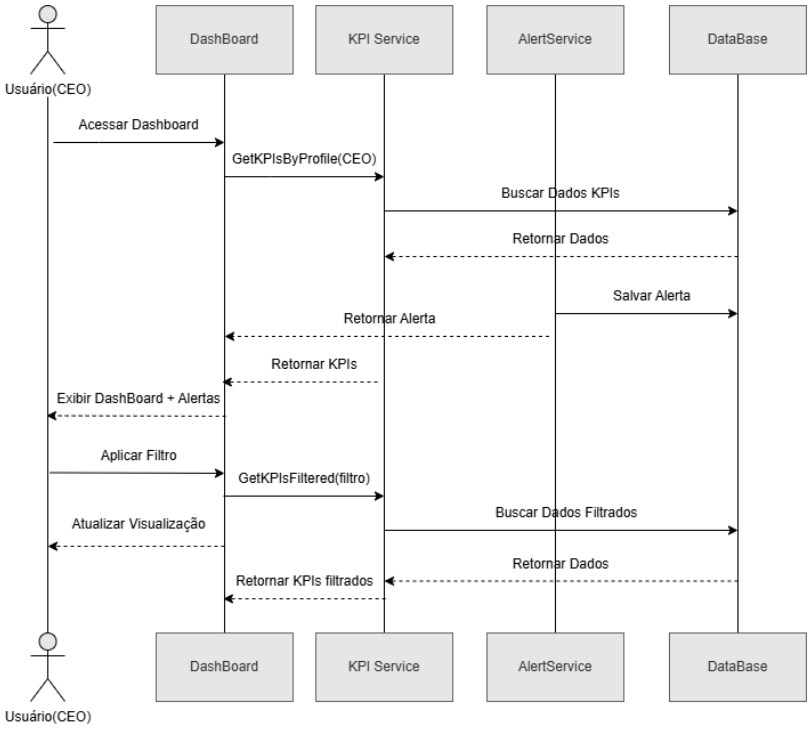


Descrição do Diagrama de Caso de Uso:

- **Visualizar Performance Geral**: CEO visualiza indicadores estratégicos consolidados.
- **Aplicar Filtros Dinâmicos**: Filtrar dados por período, região, campanha, parceiro.
- **Visualizar Alertas**: Receber notificações de anomalias nos KPIs.
- **Visualizar KPIs Financeiros**: CFO acessa indicadores de receita, custo, margem.
- **Exportar Relatórios**: Gerar relatórios em PDF ou Excel
- **Visualizar KPIs Operacionais**: CTO monitora métricas técnicas e de sistema.
- **Monitorar Dados Técnicos**: Acesso a dados de AR, geolocalização, performance.
- **Atualizar Dashboard**: Sistema atualiza dados em tempo real (simulado).
- **Simular Projeções**: Área de simulação financeira e operacional.

Diagrama de Sequência

O diagrama de sequência ilustra o fluxo de interação quando um executivo acessa o dashboard e o sistema detecta uma anomalia.



Descrição do Diagrama de Sequência:

- **Autenticação**: Usuário acessa e sistema valida credenciais.
- **Carregamento de KPIs**: Sistema busca e calcula KPIs específicos do perfil.
- **Notificação**: Se anomalia detectada, alerta é gerado e exibido.
- **Filtragem**: Usuário aplica filtros e dashboard atualiza.
- **Exportação**: Usuário solicita relatório e sistema gera arquivo.

Conclusão

A aplicação dos princípios de design de software e a modelagem UML foram fundamentais para estruturar o Dashboard PicMoney de forma robusta, escalável e manutenível. A escolha da Arquitetura em Camadas, combinada com a aplicação dos Princípios SOLID e Padrões de Projeto adequados, resultou em um sistema modular com componentes independentes e reutilizáveis.

O sistema está preparado para crescimento e implementação de novas funcionalidades, com código organizado e documentado que facilita a manutenção contínua. Os diagramas UML servem como documentação viva do sistema e facilitam a comunicação entre a equipe, tudo isso refletindo as melhores práticas de engenharia de software.