Banco FMP – Sistema Web de Gestão Bancária Simplificada

Lucas Schutz & Vitor Emanuel

Prática de Linguagem de Programação I – ADS 2ª Fase

Professor: Rafael Rosa

Visão Geral

Projeto e Tecnologias

Objetivo

Simular operações essenciais de um banco digital

Funcionalidades

- Criação de conta
- Login
- Visualização de saldo
- Transferências

Módulo extra: Administração de saldos

Stack Técnico

POO aplicada: Estrutura modular, reuso e evolução

Arquitetura: MVC com Front Controller

Persistência: MySQL via PDO com transações

Validação em múltiplas camadas e integridade de dados como requisitos não-funcionais

POO no Projeto

Aplicação Prática dos Pilares

Classes e Objetos

Usuario, Conta, Transacao representam entidades do domínio; objetos carregam estado e comportamentos específicos

Encapsulamento

Senhas tratadas via métodos dedicados no Model; uso de password_hash na criação; acesso a saldo e operações via métodos controlados

Herança

Base de entidades/Model compartilha utilitários e contratos; Controllers seguem convenção comum para padronizar fluxo

Polimorfismo

Métodos de operação (ex.: realizarTransferencia) flexíveis para diferentes tipos de conta/regra sem mudar a interface de chamada

Arquitetura MVC com Front Controller

Fluxo de Requisição

1

Request

index.php intercepta requisições

2

Front Controller

Resolve rotas e delega ao Controller

3

Controller

Orquestra entradas, validações e respostas (UsuarioController, ContaController)

4

Model

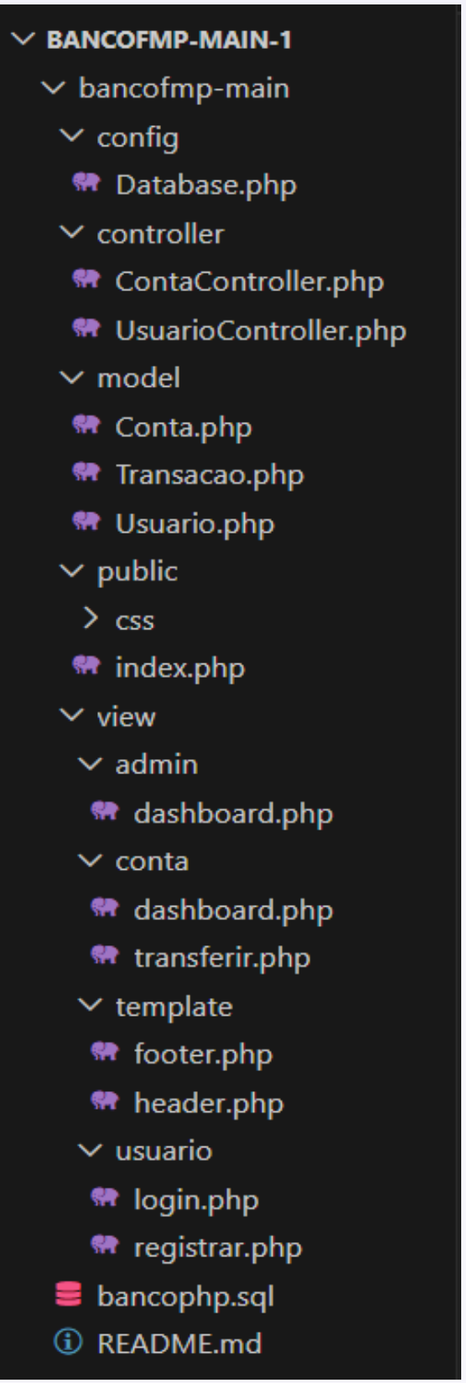
Lógica de negócio + acesso a dados (Usuario, Conta, Transacao)

5

View

Templates PHP/HTML (header.php, footer.php) e páginas dinâmicas

```php
public function transferir() {
    $id_origem = $_SESSION['usuario_id'];
    $saldo = $this->contaModel->getSaldo($id_origem);
    require_once '../view/conta/transferir.php';
}
```



Estrutura de Diretórios

Organização facilita manutenção e onboard da equipe

/config

Database.php (PDO, Singleton, erros)

/controller

UsuarioController (login/registro), ContaController (dashboard/transferência)

/model

Usuario, Conta, Transacao (regras e persistência)

/view

Templates e páginas

/public

index.php, CSS, .htaccess (roteamento)

Segurança e Integridade das Operações

Senhas

password_hash; nada em texto puro

```php
public function registrar($nome, $email, $senha) {
    $hash = password_hash($senha, PASSWORD_DEFAULT);
    $stmt = $this->conn->prepare("INSERT INTO usuarios (nome, email,
senha) VALUES (?, ?, ?)");
    return $stmt->execute([$nome, $email, $hash]);
}

public function checkSenha($id, $senhaDigitada) {
    $usuario = $this->getById($id);
    return password_verify($senhaDigitada, $usuario['senha']);
}
```

Transações Atômicas

beginTransaction, commit, rollBack no realizarTransferencia

Se crédito falha, débito é revertido — invariantes financeiros preservados

Banco de Dados

Tabelas usuarios, contas, transacoes com integridade relacional

Operação atômica: Origem → Destino com transação ACID

Conexão com Banco de Dados (PDO)

Interface PDO

PDO para segurança e flexibilidade

Singleton

Padrão Singleton/estático — uma instância global controlada

Tratamento de Erros

PDO::ERRMODE_EXCEPTION para tratamento e logs

Reduz overhead e melhora observabilidade de falhas

Validação em Camadas

Defense in Depth — cada camada previne classes distintas de erro

Front-end

- HTML5 required
- Tipos (email)
- Regex pattern

Back-end

- FILTER_VALIDATE_EMAIL
- Regras de negócio (valores > 0 e ≤ saldo)

Banco de Dados

- UNIQUE em email
- Constraints para integridade

Desafios e Soluções

Integridade financeira

Uso de transações ACID para evitar inconsistências

Rotas e dependências

Autoloader + roteamento centralizado

Segurança de credenciais

password_hash e boas práticas

Resultados e Aprendizados

Objetivos Atingidos

- Cadastro, login, contas, transferências confiáveis
- Organização por camadas; legível e escalável
- Validações distribuídas; correções rápidas; base para novas features

O que adicionaríamos com a mesma base? Extratos, limites, tarifas, perfis de conta

Conclusão

- POO e MVC aplicados de forma prática, com foco em segurança e integridade
- Estrutura sólida que reduz risco e acelera evolução
- Próximos passos: Relatórios, auditoria de transações, testes automatizados

---

Referências

SILVA, Cleuton. Os 4 Pilares da Programação Orientada a Objetos. Disponível em: https://www.dio.me/articles/os-4-pilares-da-programacao-orientada-a-objetos-SSU4Q9. Último acesso em: 21 de novembro de 2025.

GASPAROTTO, Henrique. POO: Os 4 pilares da Programação Orientada a Objetos. Disponível em: https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264. Último acesso em: 21 de novembro de 2025.

SOUZA, Ângelo. Arquitetura MVC: Entendendo o Modelo-Visão-Controlador. Disponível em: https://www.dio.me/articles/arquitetura-mvc-entendendo-o-modelo-visao-controlador. Último acesso em: 21 de novembro de 2025.

MEDEIROS, Higor. Introdução ao Padrão MVC: Primeiros passos na Arquitetura MVC. Disponível em: https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308. Último acesso em: 21 de novembro de 2025.