AVALIAÇÃO PRÁTICA – PARTE I

A primeira parte da avaliação prática deverá estar integralmente implementada até o dia **06/06/2025**. O discente deverá comparecer à avaliação com o projeto funcional e pronto para execução. Na data da avaliação, serão divulgados os requisitos e os critérios de avaliação da segunda parte da prova. A composição da nota final seguirá a seguinte distribuição:

- Parte I: peso de 30%
- Parte II: peso de 60%

A prova prática consiste no desenvolvimento de uma aplicação para gerenciamento de clientes, contas correntes e lançamentos financeiros, conforme os requisitos funcionais e não funcionais estabelecidos na especificação do projeto.

A solução deve ser baseada em uma arquitetura em camadas (Figura 01), contemplando a separação entre as responsabilidades de apresentação (Front-end), lógica de negócios e persistência de dados (Back-end), com uso de frameworks apropriados para cada parte.

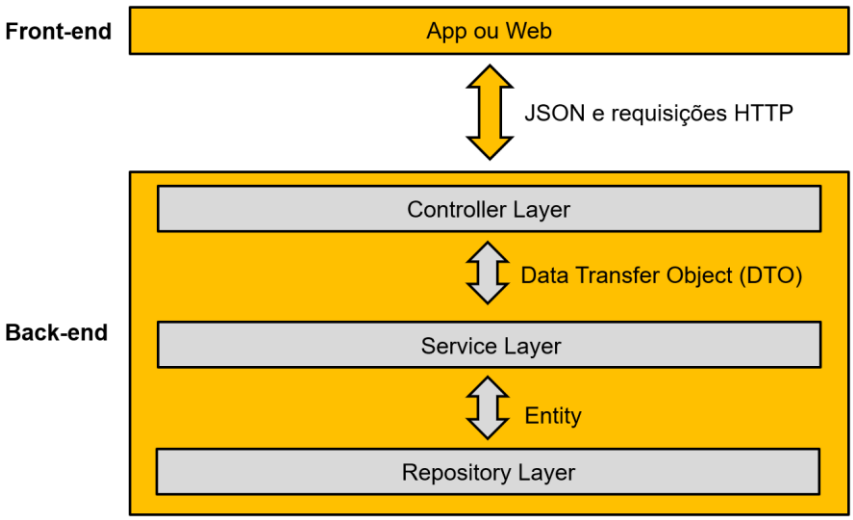

Figura 01: Arquitetura em Camadas

A escolha das tecnologias a serem utilizadas no desenvolvimento do sistema — incluindo frameworks, linguagem de programação e Sistema Gerenciador de Banco de Dados (SGBD) — fica a critério do discente. Abaixo, seguem exemplos de tecnológicas válidas:

- Front-end: Angular, React, Vue.js
- Back-end: Spring Boot (Java), Laravel (PHP), Node.js (JavaScript/TypeScript)
- SGBD: MySQL, PostgreSQL, MongoDB

Observação: O discente deve assegurar que as tecnologias escolhidas sejam compatíveis com os requisitos da avaliação e permitam a implementação das funcionalidades esperadas.

Projeto da Aplicação Back-end

O desenvolvimento da aplicação back-end deve atender aos requisitos funcionais e não funcionais descritos a seguir:

- **Modelo de Dados**
  O diagrama do banco de dados (Figura 02) é composto por três tabelas que modelam as entidades principais do sistema: clientes, contas correntes e lançamentos financeiros. Essa estrutura pode ser estendida pelo discente conforme a necessidade da solução proposta. A adição de novas tabelas ou campos é permitida e, em alguns casos, necessária para garantir a aderência aos requisitos do sistema.

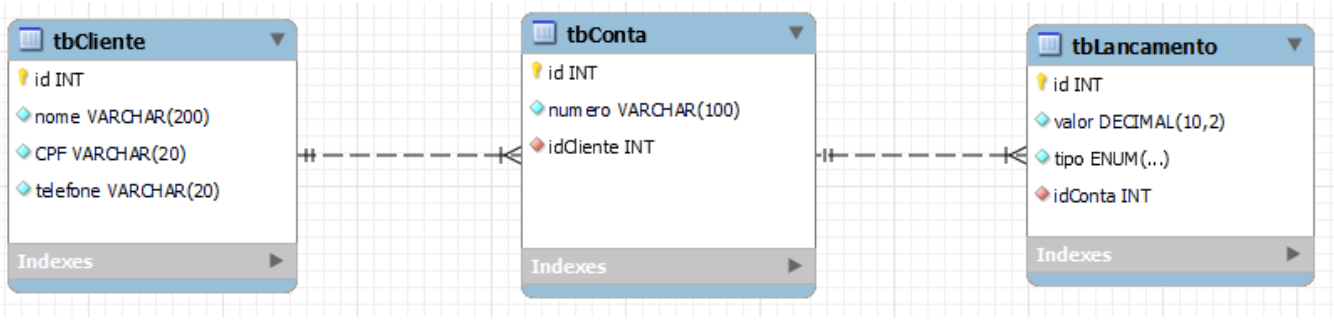

Figura 02: Modelo de Dados

- **tbCliente**: tabela responsável por armazenar os dados dos clientes do banco.
  - Todos os campos devem ser obrigatórios.
  - O campo **CPF** deve possuir valor único (restrição de unicidade).
- **tbConta**: tabela que armazena as informações das contas correntes.
  - Todos os campos devem ser obrigatórios.
  - O campo **número** da conta deve ser único.
  - Cada conta deve estar associada a um cliente, por meio de chave estrangeira.
- **tbLancamento**: tabela destinada ao registro dos lançamentos financeiros das contas correntes.
  - Todos os campos devem ser obrigatórios.
  - O campo **valor** não pode assumir valores negativos.
  - O campo **tipo** deve aceitar apenas os valores 'credito' e 'debito' (pode ser implementado com tipo ENUM).
  - Cada lançamento deve estar vinculado a uma conta corrente, por meio de chave estrangeira.

Mapeamento objeto-relacional (ORM)

O discente deve seguir a arquitetura proposta para o desenvolvimento da aplicação, respeitando a separação entre as camadas da solução. É obrigatório realizar o ORM entre o banco de dados e as classes que representam as entidades do domínio, garantindo a integridade dos dados e a consistência entre os modelos. Por exemplo:

```
public class Cliente {
   private Long id;
   private String nome;
   private String cpf;
   private String telefone;
}
```

```
public class Conta {
   private Long id;
   private String numero;
   private Cliente cliente;
}
```

```
public class Lancamento {
   private Long id;
   private Double valor;
   private Conta conta;
   private TipoLancamento tipo;
}

public enum TipoLancamento {
    CREDITO,
    DEBITO
}
```

Data Transfer Objects (DTOs)

O padrão DTO (Data Transfer Object) é utilizado para transportar dados entre as camadas da aplicação. Sua principal função é encapsular os dados que realmente precisam ser expostos, evitando o envio de informações desnecessárias ou sensíveis. No contexto deste projeto, o uso de DTOs pode contribui para:

- Maior segurança, ao evitar a exposição de atributos sensíveis das entidades.
- Melhor desempenho, ao transmitir apenas os dados relevantes.
- Maior controle sobre o formato de saída da API, desacoplando a estrutura da base de dados da estrutura das respostas.

A entidade **Conta** está relacionada com a entidade **Cliente**, e o objetivo é expor apenas o número da conta e CPF do cliente, ocultando dados sensíveis como o nome e o telefone do cliente. Nesse caso, podemos criar uma classe DTO da seguinte forma:

```
public class ContaDTO {
    private Long id;
    private String numero;
    private String cpf;

    public ContaDTO(){

    }

    public ContaDTO(Conta conta) {
        this.id = conta.getId();
        this.numero = conta.getNumero();
        this.cpf = conta.getCliente().getId();
    }

    // Getters
}
```

Dessa forma, o discente deve obrigatoriamente utilizar o padrão DTO para garantir que informações sensíveis, como nome e telefone do cliente, não sejam expostas nas respostas que envolvem dados de contas correntes e lançamentos financeiros.

Camadas da Aplicação

A aplicação deve seguir a arquitetura proposta (Figura 01), que favorece a separação de responsabilidades, a manutenção do código e a escalabilidade da solução. Essa abordagem organiza o sistema em três camadas principais, cada uma com uma função bem definida:

- **Repositório**: camada responsável pelo acesso e manipulação dos dados persistidos no banco de dados. Nela, são definidas as interfaces ou classes que realizam operações de consulta, inserção, atualização e remoção de registros.
- **Serviço**: camada intermediária onde reside a lógica de negócio da aplicação. É responsável por processar as regras do sistema, coordenar operações entre diferentes entidades e garantir a integridade das ações realizadas.
- **Controle**: camada encarregada de expor a interface da aplicação por meio de endpoints, normalmente seguindo o padrão REST. Essa camada recebe as requisições do cliente, encaminha-as ao serviço adequado e retorna as respostas apropriadas.

Endpoints da API

A aplicação back-end deve ser implementada como uma API REST, responsável por disponibilizar os dados e funcionalidades do sistema por meio de endpoints acessíveis ao front-end. Essa estrutura permite a comunicação entre cliente e servidor de forma padronizada, utilizando o protocolo HTTP.

Um endpoint é uma URL que representa um recurso ou uma ação dentro da API. Ele define um ponto de acesso a determinada funcionalidade do sistema, como consultar, criar, atualizar ou remover dados. Cada endpoint é associado a um verbo HTTP específico (GET, POST, PUT, DELETE, etc.), refletindo a operação a ser realizada. Por exemplo:

Método	URL	Descrição
GET	http://localhost:8080/clientes	Retorna a lista de todos os clientes cadastrados.
GET	http://localhost:8080/clientes /{id}	Retorna os dados de cliente específico pelo ID.
POST	http://localhost:8080/clientes	Cadastra um novo cliente
PUT	http://localhost:8080/clientes /{id}	Atualiza os dados de um cliente existente.
DELETE	http://localhost:8080/clientes /{id}	Remove um cliente.
GET	http://localhost:8080/clientes /{id}/contas	Lista as contas associadas ao cliente identificado por {id}

Esses endpoints devem seguir boas práticas de design RESTful, utilizando nomes no plural para representar coleções de recursos e empregando rotas claras e consistentes. A implementação correta desses endpoints garante que o front-end consiga consumir os dados da aplicação de forma segura, organizada e eficiente.

Segurança e controle de acesso

No desenvolvimento da API do sistema bancário, o discente deve implementar mecanismos de segurança e controle de acesso, protegendo todos os endpoints, exceto os de cadastro de nvoos clientes e autenticação, que devem ser de acesso público. A autenticação deve utilizar um esquema seguro de geração e validação de tokens, garantindo que apenas usuários autenticados acessem os recursos protegidos.

Projeto da Aplicação Front-end

O desenvolvimento da aplicação front-end deve atender aos requisitos funcionais e não funcionais descritos a seguir. Os protótipos apresentados têm caráter ilustrativo, representando possíveis interfaces do sistema. O discente tem autonomia para definir o design e o layout que considerar mais adequados às necessidades do projeto.

- **Cadastro e autenticação de cliente**

  O sistema deve oferecer funcionalidade para o cadastro de novos clientes, exigindo, obrigatoriamente, os seguintes dados: nome, CPF, senha e telefone (Figura 03). O CPF deve ser único, não sendo permitido o registro de clientes com CPF duplicado. O acesso ao sistema deve ser restrito aos usuários previamente cadastrados na base de dados.

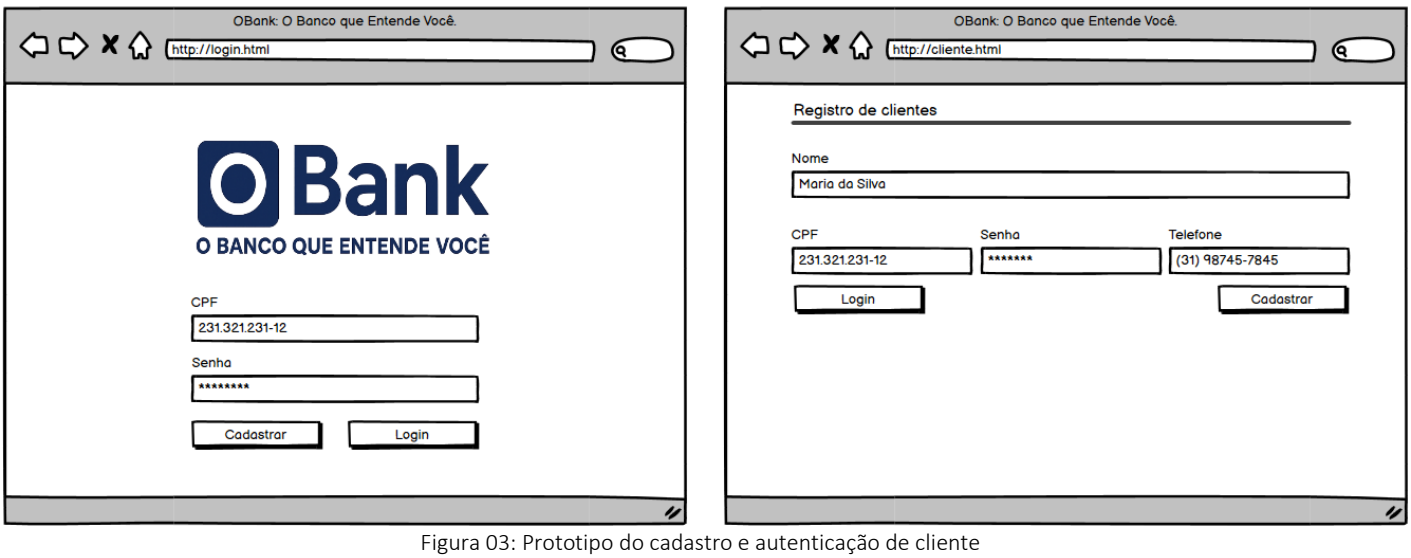

Figura 03: Prototipo do cadastro e autenticação de cliente

- **Menu de acesso**

  Após a autenticação, o sistema deve identificar o cliente e exibir seu primeiro nome na interface, juntamente com as funcionalidades disponíveis. As opções acessíveis ao cliente são:

  - Cadastro de nova conta
  - Realização de saque
  - Realização de depósito
  - Consulta de extrato
  - Encerramento da sessão (Sair)

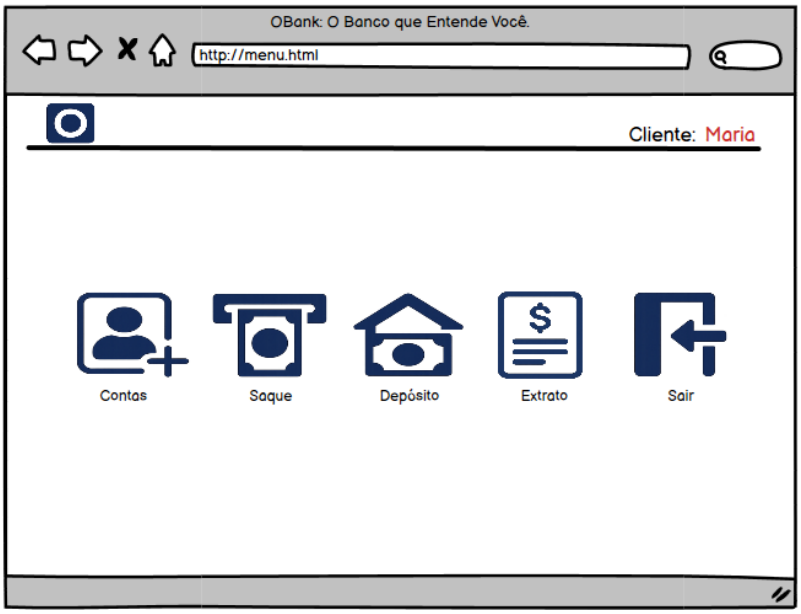

Figura 04: Prototipo menu de acesso

- **Cadastro de nova conta**

  O sistema deve permitir que o cliente visualize suas contas existentes, exibindo o número da conta e o respectivo saldo. Também deve ser possível cadastrar novas contas (Figura 05). O número da conta deve ser gerado automaticamente pelo sistema, seguindo o padrão **AA-999999**, onde:

  - **AA** corresponde às duas primeiras letras do nome do cliente (em maiúsculas);
  - **999999** é um número aleatório de seis dígitos.

  Antes de registrar a nova conta, o sistema deve verificar se o número gerado é único na base de dados. Caso o número já exista, um novo valor deve ser gerado até que a unicidade seja garantida.

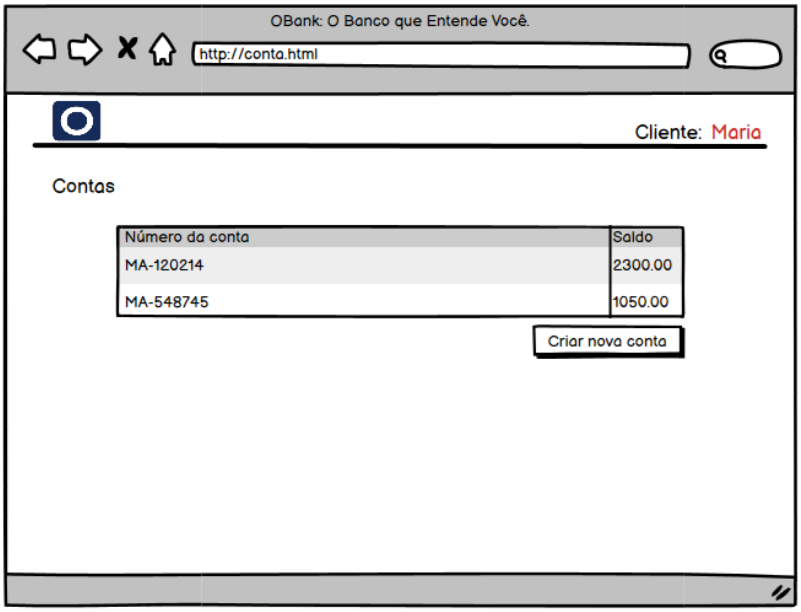

Figura 05: Cadastro de novas contas

- **Operação de Saque**

  O sistema deve permitir que o cliente realize operações de saque. Para isso, o cliente deve selecionar uma de suas contas cadastradas e informar o valor a ser sacado. A operação não deve ser executada caso o saldo da conta seja insuficiente. O sistema deve garantir que o cliente só possa realizar saques em contas de sua titularidade.

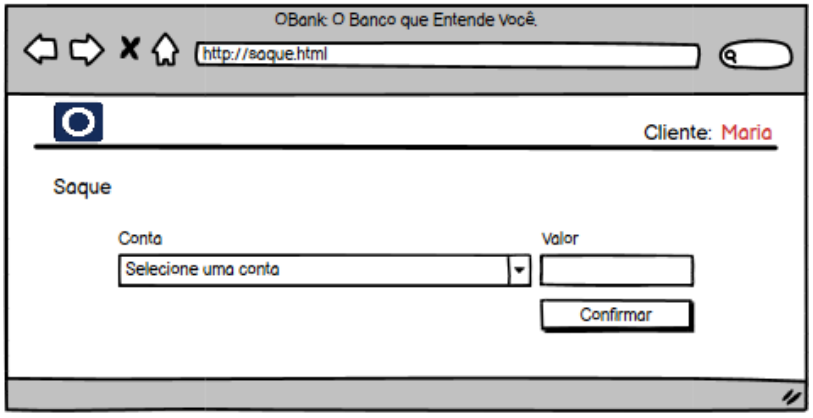

Figura 06: Operação de saque

- **Operação de Depósito**

  O sistema deve permitir que o cliente realize operações de depósito. Para isso, o cliente deve selecionar uma de suas contas cadastradas e informar o valor a ser depositado. Assim como no saque, o sistema deve restringir a operação às contas pertencentes ao cliente autenticado.

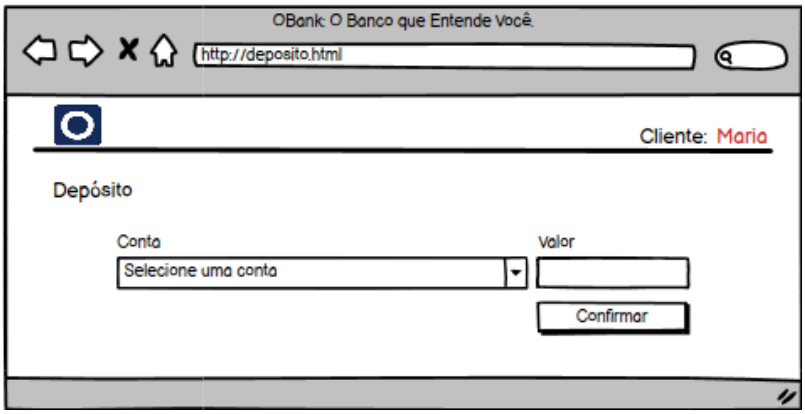

Figura 06: Operação de depósito

- **Consultar Extrato**

  O sistema deve permitir que o cliente consulte o extrato de uma conta. Para isso, o cliente deve selecionar uma de suas contas cadastradas. A consulta deve ser restrita às contas pertencentes ao cliente autenticado.

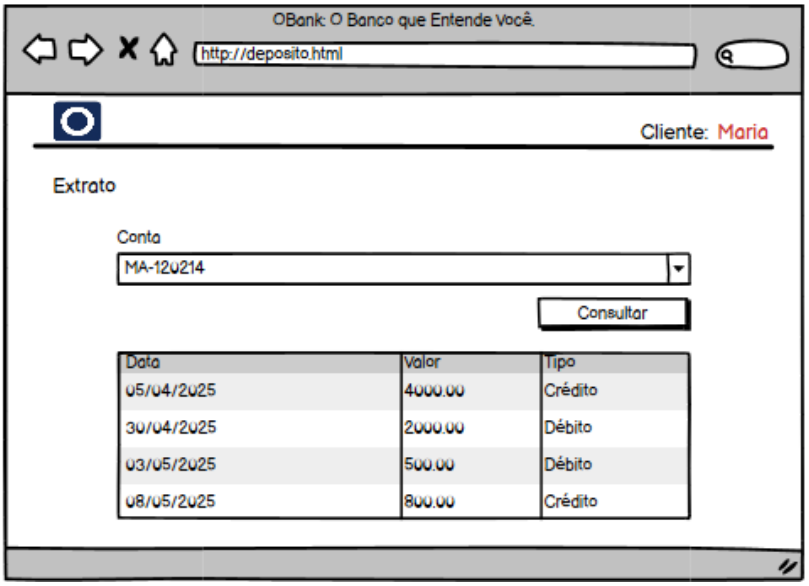

Figura 07: Consulta de extrato

- **Responsividade**

  O sistema deve ser desenvolvido com foco em responsividade, garantindo que a interface se adapte de forma adequada a diferentes tamanhos de tela, como os de computadores, notebooks e smartphones. O discente deve assegurar que todas as funcionalidades estejam acessíveis e visualmente organizadas independentemente da resolução ou tipo de tela utilizada pelo usuário

Critérios de avaliação

Parte I

- Aplicação Back-end **[15 pontos]**
  - A aplicação será avaliada quanto à aplicação da arquitetura em camadas, com separação clara entre camadas de repositório, serviço e controlador. Serão analisados o funcionamento e a consistência dos endpoints da API REST, especialmente no que se refere à realização das operações de CRUD e à integração com o front-end. Também será considerada a implementação de mecanismos de segurança, como autenticação via tokens e controle de acesso a recursos protegidos, assegurando que apenas usuários devidamente autenticados possam executar ações restritas.

- Aplicação Front-end **[15 pontos]**
  - Será avaliada a implementação das funcionalidades de cadastro e autenticação de clientes, garantindo o controle de acesso ao sistema. A criação de contas deve assegurar a geração automática e única do número da conta. As operações de saque e depósito serão verificadas quanto ao cumprimento das regras de negócio, incluindo a validação de saldo disponível. Além disso, será avaliada a funcionalidade de consulta de extrato e a responsividade da interface, assegurando usabilidade adequada em diferentes dispositivos.

Parte II

- Os requisitos e os critérios de avaliação referentes à Parte II serão divulgados no início da avaliação prática que será realizada no dia 06/06/2025, na sala 47 do Bloco B (Laboratório de Informática I), com duração máxima de 4 horas, entre 13h30 e 17h30.