Documentação de Projeto

para o sistema

DigiBank

Versão 1.0

Projeto de sistema elaborado pelo aluno Flávio de Souza Ferreira Júnior

como parte da disciplina Projeto de Software.

13 de Novembro de 2025

Tabela de Conteúdo

Nome	**Data**	**Razões para Mudança**	**Versão**
Flávio de S. F. Júnior	13/11/2025	Criação inicial do documento e capa	1.0
Flávio de S. F. Júnior	13/11/2025	Adicionada descrição dos atores e casos de uso	1.1
Flávio de S. F. Júnior	14/11/2025	Inseridos diagramas (casos de uso, sequência, classes, arquitetura)	1.2
Flávio de S. F. Júnior	14/11/2025	Inclusão dos contratos de operação (UC-01 a UC-05)	1.3

1 Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema DigiBank. A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o documento de especificação que descreve a visão de domínio do sistema.

2 Modelos de Usuário e Requisitos

2.1 Descrição de Atores

Cliente (Ator Primário)

Papel: Usuário final do DigiBank (pessoa física ou jurídica).
Responsabilidades: Registrar conta, autenticar (login), ver seu saldo, realizar transferências, consultar histórico de transações e gerenciar dados pessoais.
Relação com o sistema: Interage pela interface web (React). Envia requisições HTTP (via api.ts) às rotas do backend (/clientes, /transacoes).
Exemplos de ações: UC-01 Registrar, UC-02 Login, UC-03 Transferir, UC-04 Ver Saldo, UC-05 Ver Histórico.

Sistema (Backend/API) — Ator Secundário / Sistema Interno

Papel: Fornece serviços (endpoints Express) para autenticação, gerenciamento de clientes e transações.
Responsabilidades: Validar dados, autenticar tokens JWT, executar transações atômicas (BEGIN/COMMIT), persistir dados no PostgreSQL.
Interface: Rotas em `server.js`, controllers em `controllers/*`, models como `clienteModel.js` e módulo `db.js`.

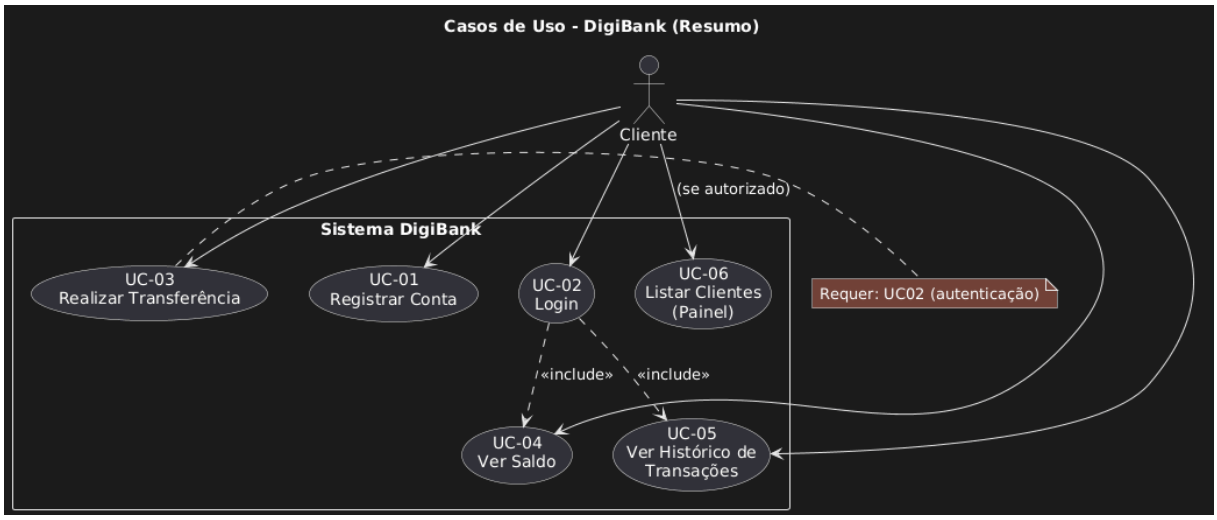
Banco de Dados (PostgreSQL)

Papel: Persistência de dados (tabelas `clientes` e `transacoes` conforme `init.sql`).
Responsabilidades: Armazenar clientes, senhas (hash), saldo, histórico de transações; garantir integridade referencial através de chaves estrangeiras.

2.2 Modelo de Casos de Uso

Casos de uso identificados (IDs para referência):

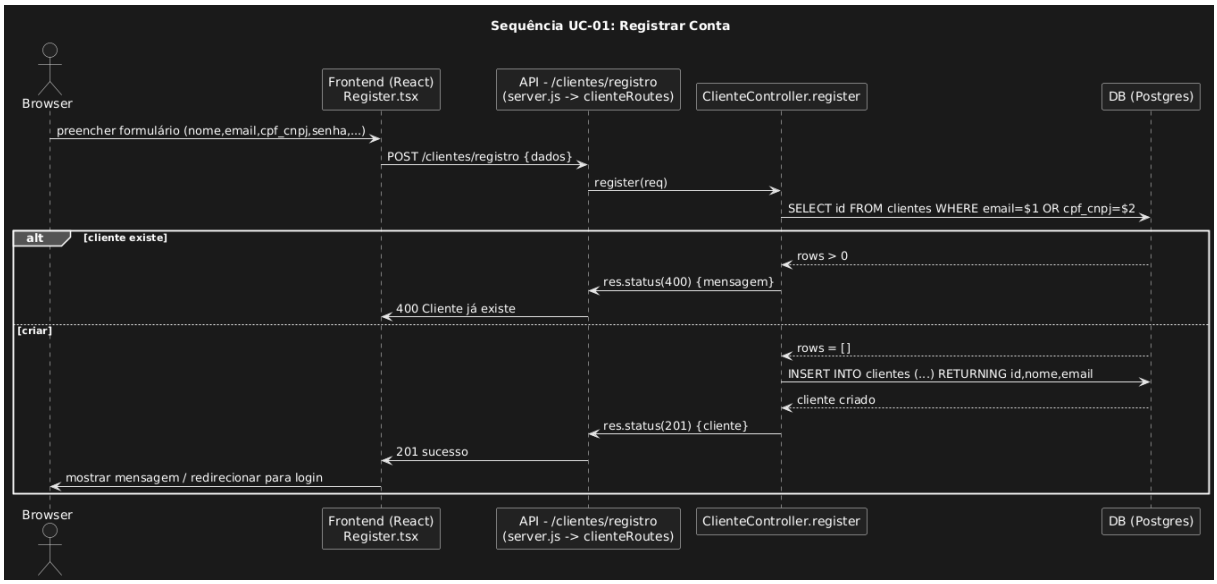
UC-01: Registrar Conta
UC-02: Login / Autenticação
UC-03: Realizar Transferência
UC-04: Ver Saldo
UC-05: Ver Histórico de Transações



2.3 Diagrama de Sequência do Sistema

UC-01 — Registrar Conta (Register)

Rota: POST /clientes/registro
Ator: Cliente (Browser)
Pós-condição: cliente criado na tabela `clientes`, senha armazenada como hash, saldo inicial definido.

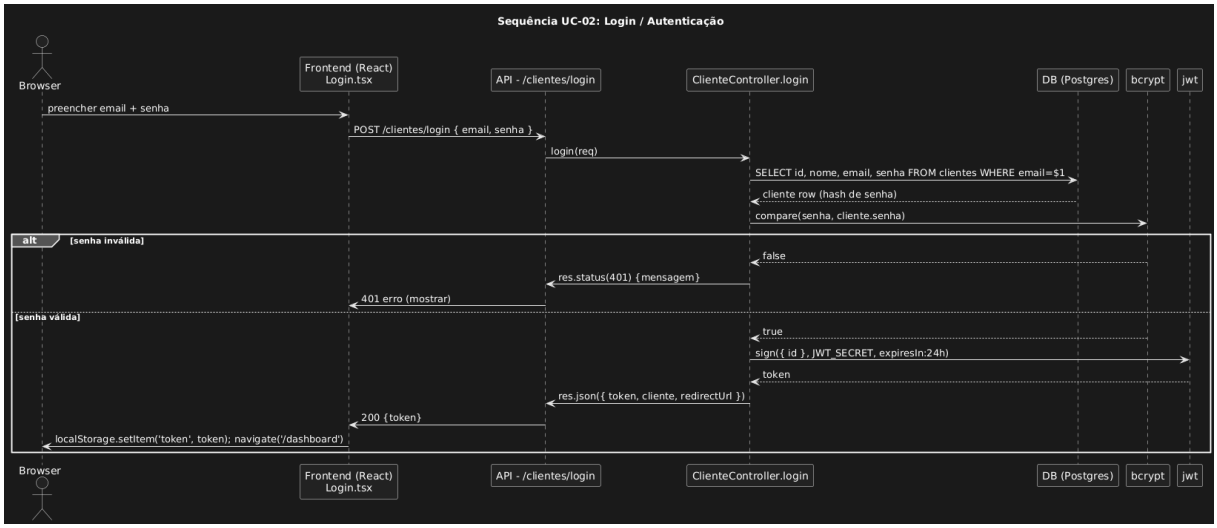


UC-02 — Login / Autenticação

Rota: POST /clientes/login

Ator: Cliente

Pós-condição: token JWT emitido (24h), cliente recebe token e navega ao dashboard.

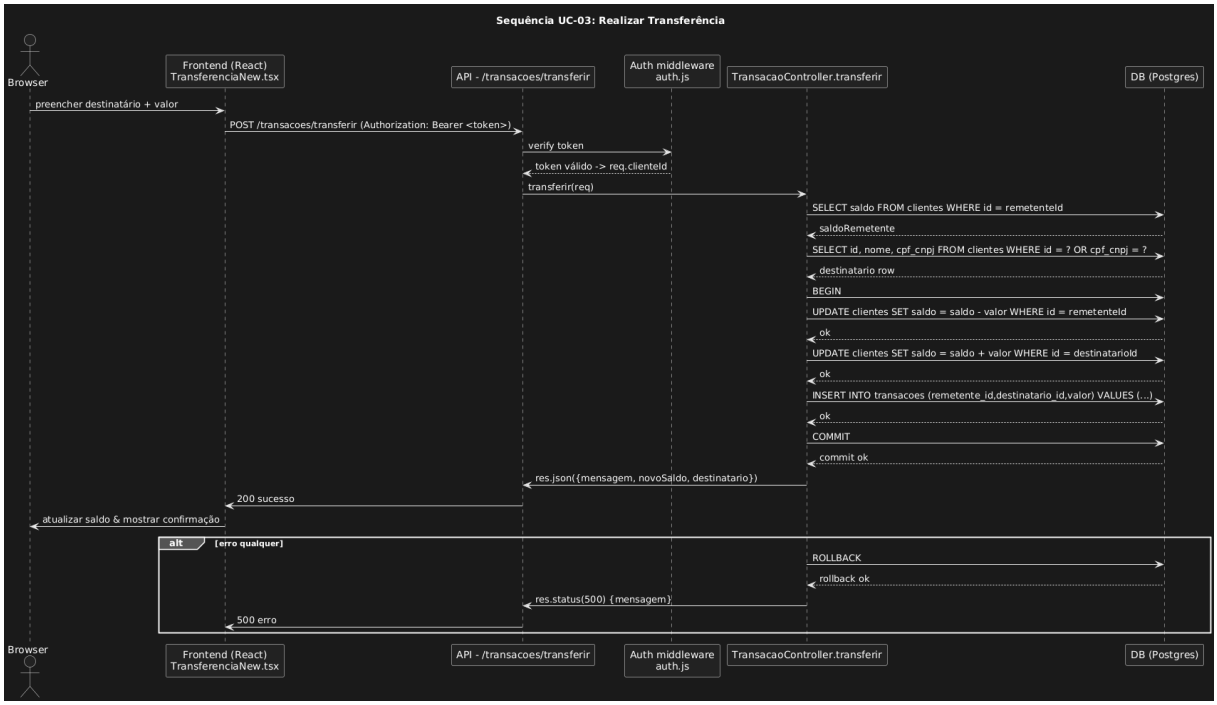


UC-03 — Realizar Transferência

Rota: POST /transacoes/transferir

Ator: Cliente (autenticado)

Observações: Operação envolve débito do remetente, crédito do destinatário e inserção em `transacoes`. É executada em transação DB (`BEGIN`/`COMMIT`) com `ROLLBACK` em caso de erro.



Formato para cada contrato de operação

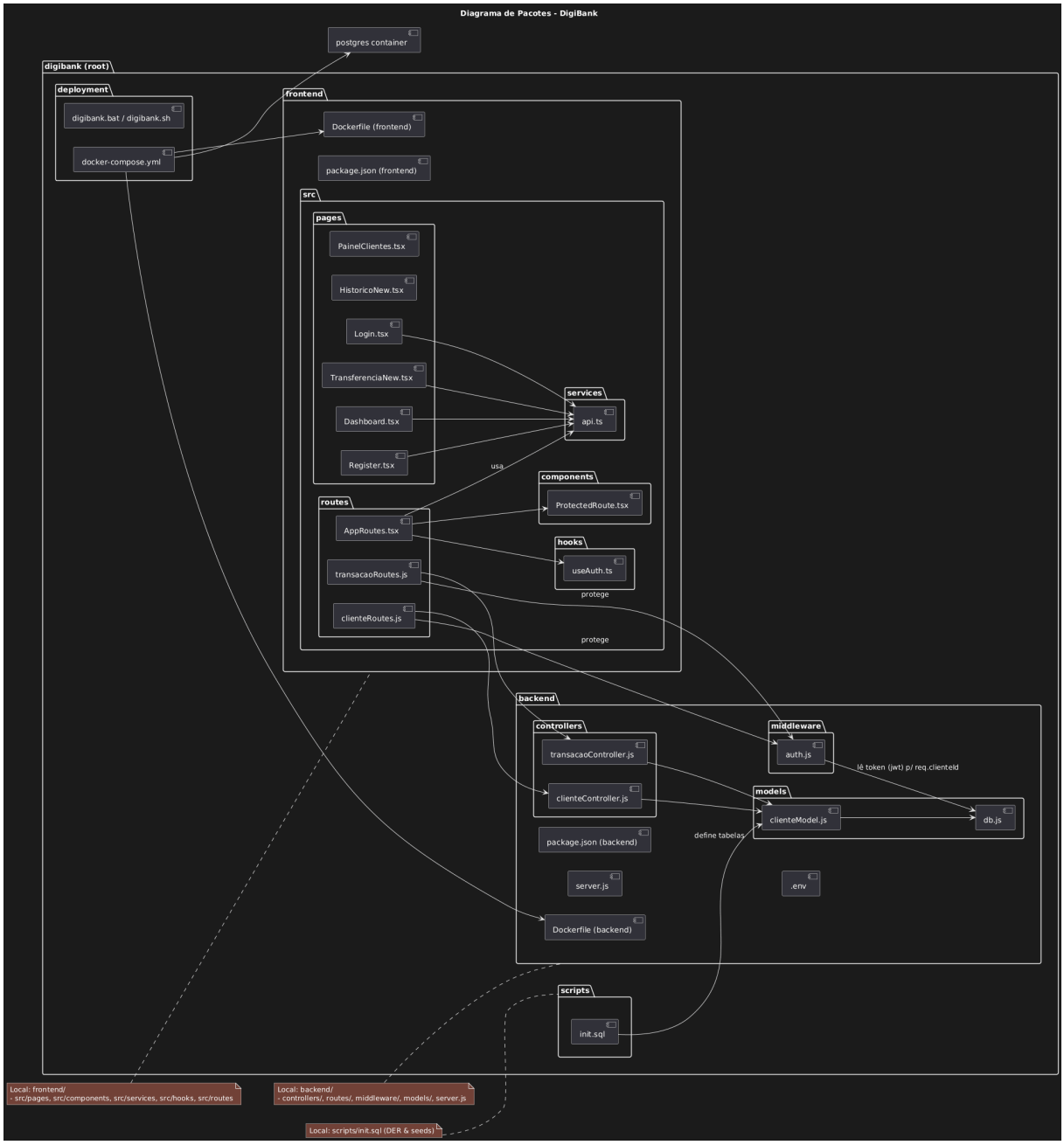
Contrato	Contrato de Registro de Cliente
Operação	registrarCliente(nome, email, senha)
Referências cruzadas	UC-01 Registrar Conta; RF-01; Diagrama de Sequência UC-01
Pré-condições	• Requisição enviada pelo cliente via interface web. • Dados obrigatórios fornecidos: nome, email e senha. • Email ainda não cadastrado no sistema.
Pós-condições	• Registro criado na tabela `clientes`. • Senha armazenada como hash. • Saldo inicial definido como 0. • Cliente apto a realizar login.

Contrato	Contrato de Autenticação de Cliente
Operação	autenticar(email, senha)
Referências cruzadas	UC-02 Login; RF-02; Diagrama de Sequência UC-02
Pré-condições	• Cliente possui cadastro ativo. • Email informado existe na base. • Senha fornecida corresponde ao hash armazenado.
Pós-condições	• Token JWT emitido (válido por 24h). • Cliente autenticado e autorizado a acessar rotas protegidas. • Sistema registra tentativa bem-sucedida de login.

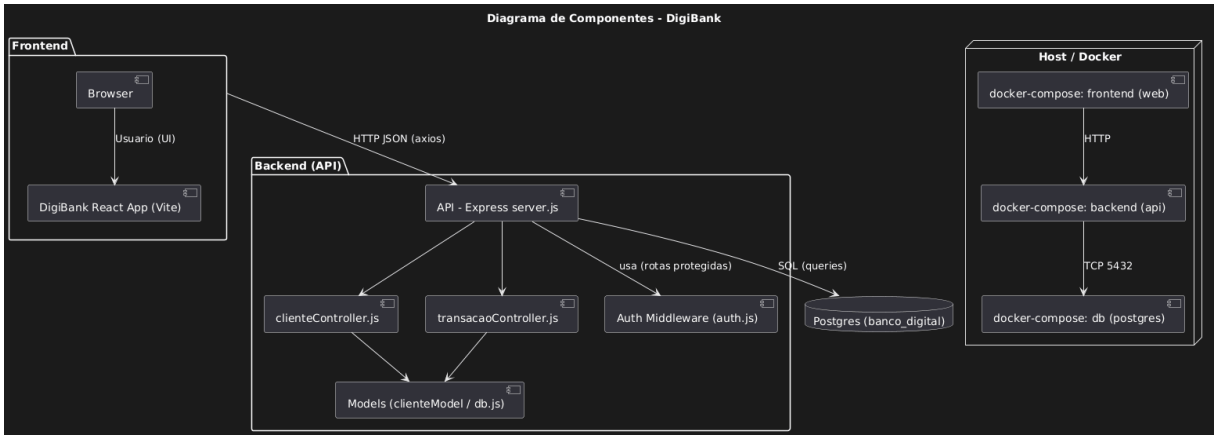
Contrato	Contrato de Transferência de Valores
Operação	transferirValor(idRemetente, idDestinatario, valor)
Referências cruzadas	UC-03 Realizar Transferência; RF-03; Diagrama de Sequência UC-03
Pré-condições	• Remetente autenticado com token JWT válido. • Contas do remetente e destinatário existem. • Saldo do remetente é suficiente para a operação. • Valor da transferência é maior que zero. • Operação ocorre dentro de uma transação do banco (BEGIN).
Pós-condições	• Saldo do remetente decrementado (saldo - valor). • Saldo do destinatário incrementado (saldo + valor). • Registro inserido na tabela `transacoes`. • Transação finalizada com `COMMIT`. • Em caso de erro: `ROLLBACK` restaurando estado anterior.

3 Modelos de Projeto

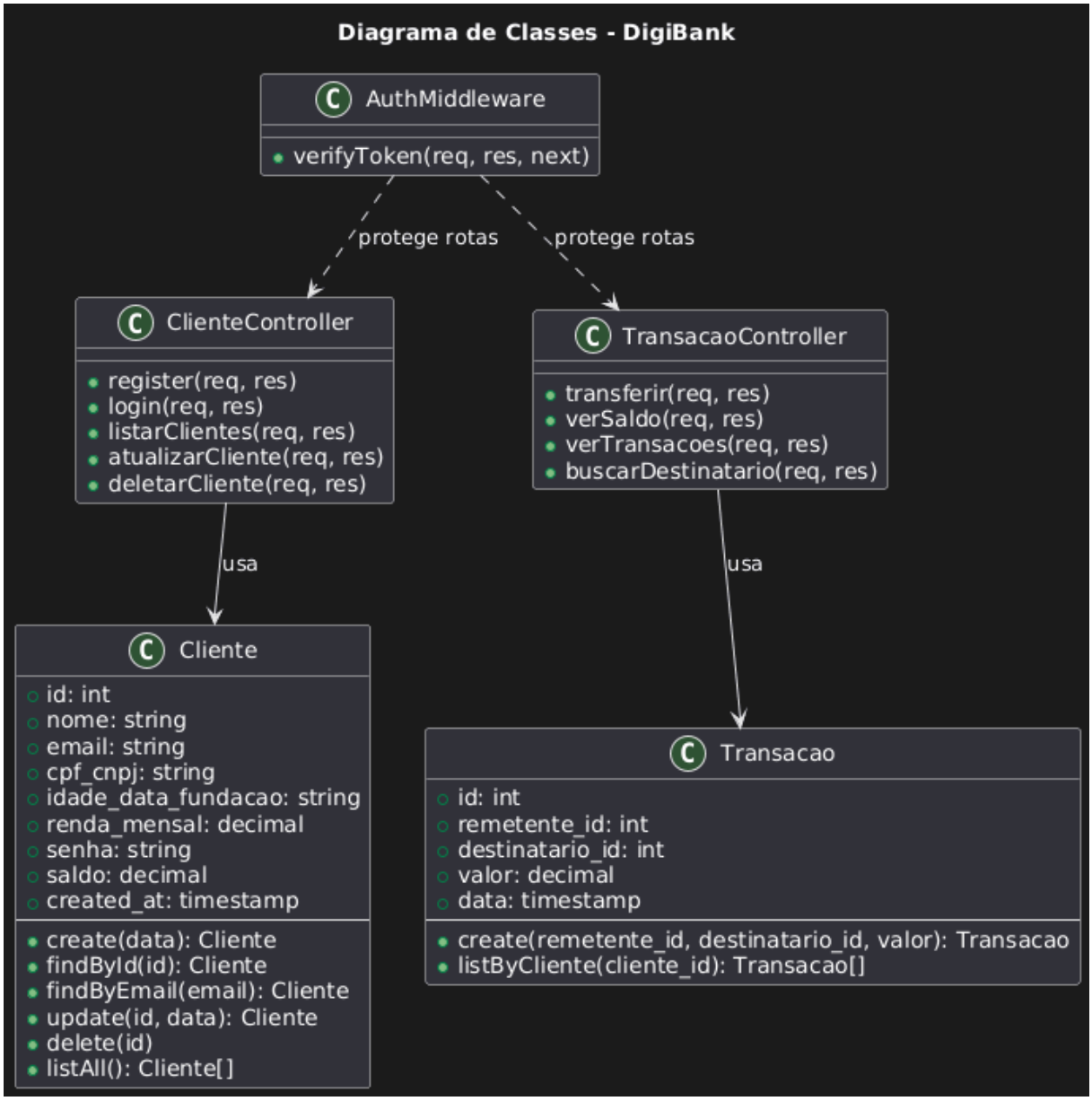
3.1 Arquitetura



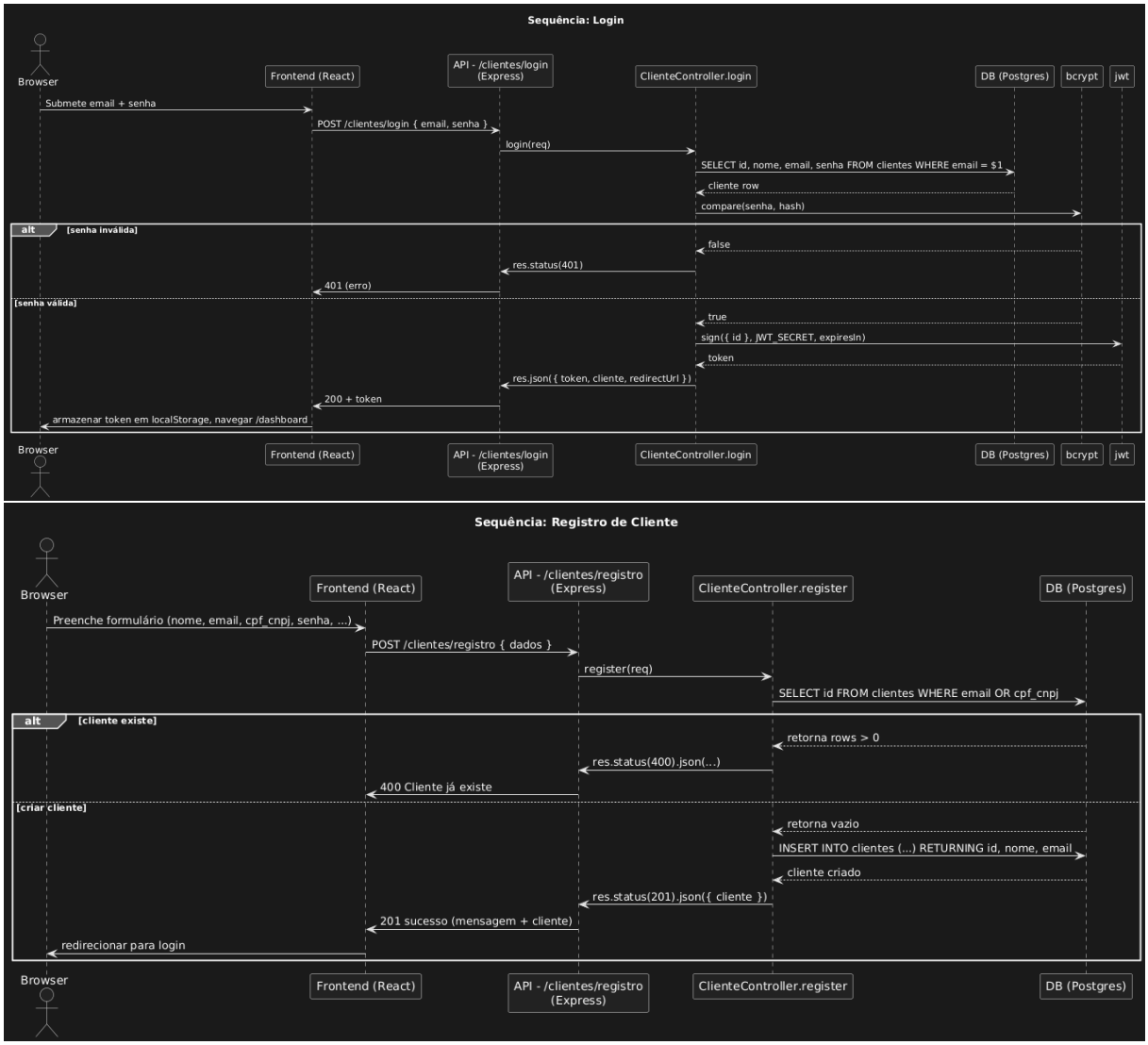
3.2 Diagrama de Componentes e Implantação

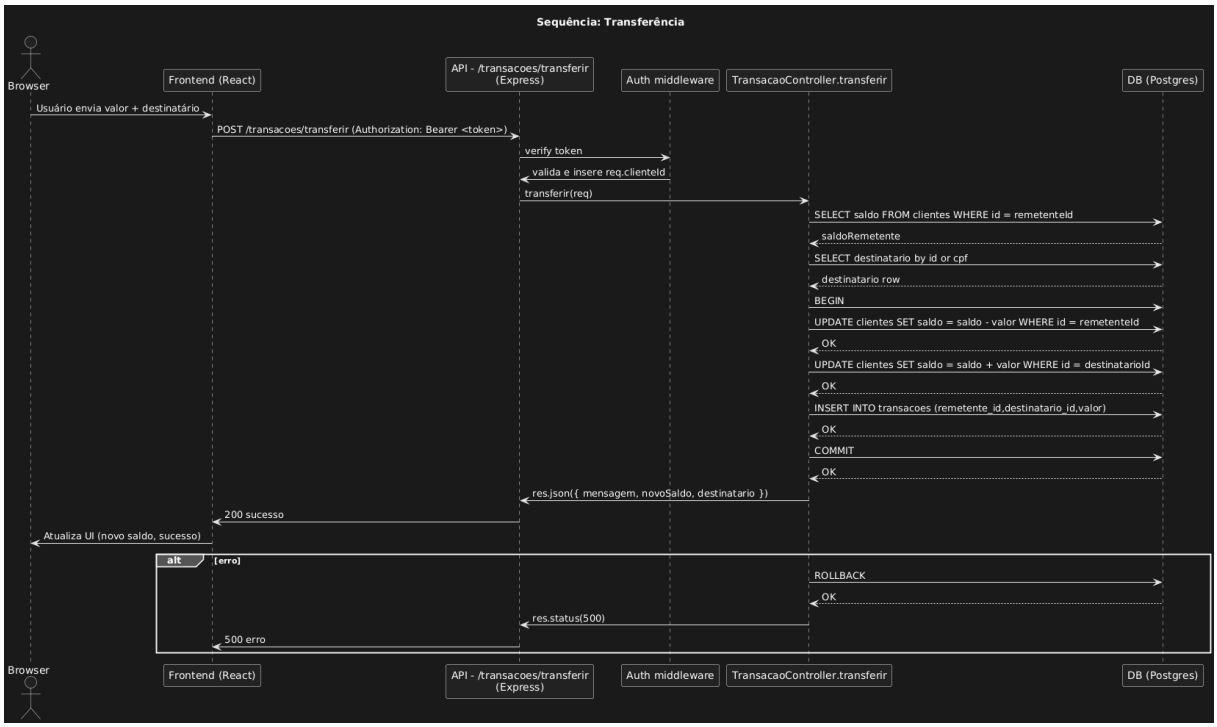


3.3 Diagrama de Classes

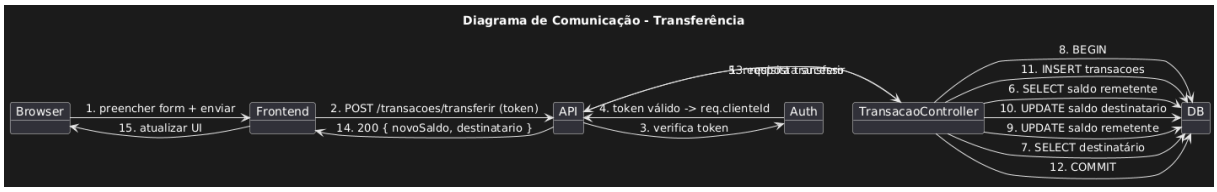


3.4 Diagramas de Sequência

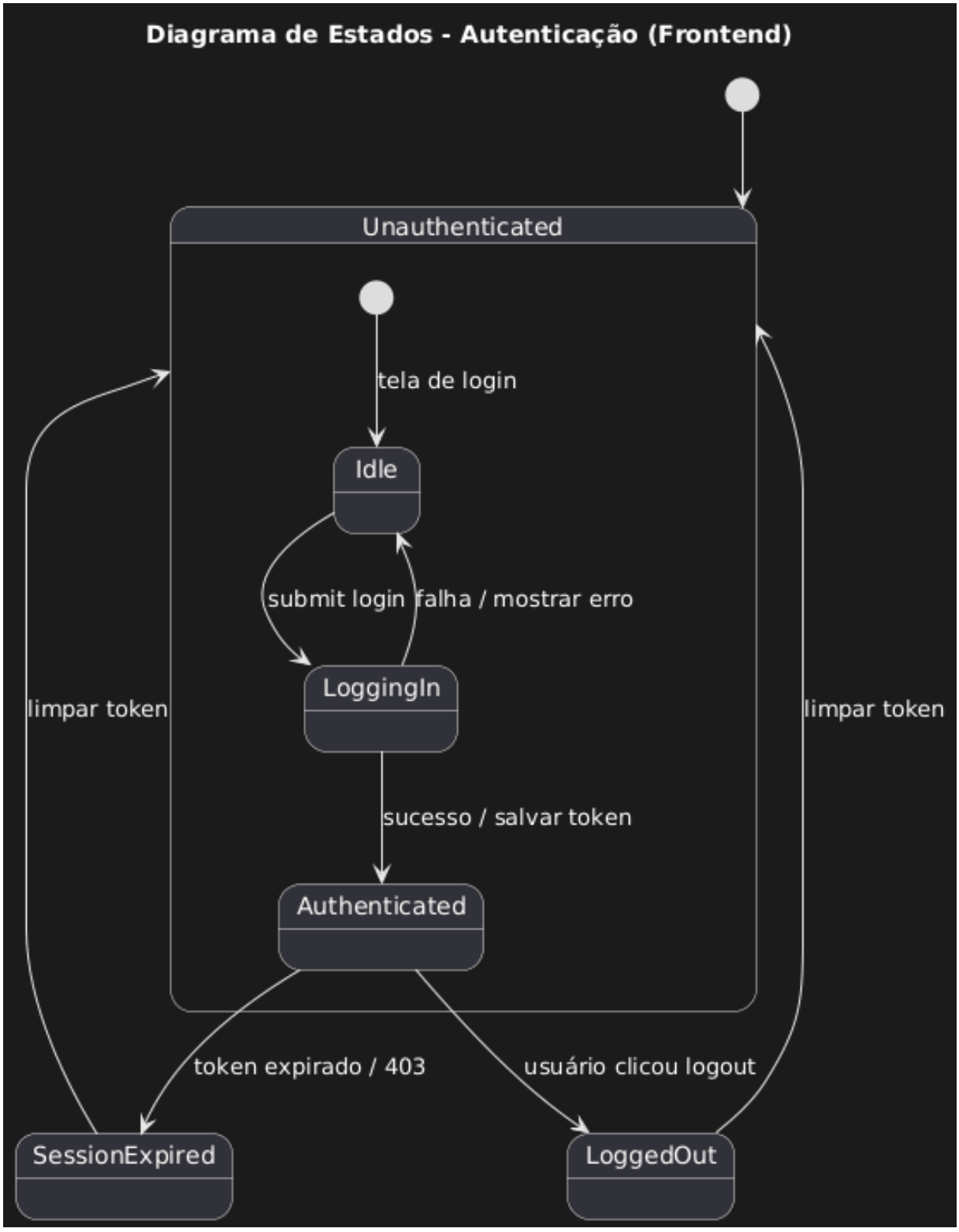




3.5 Diagramas de Comunicação



3.6 Diagramas de Estados



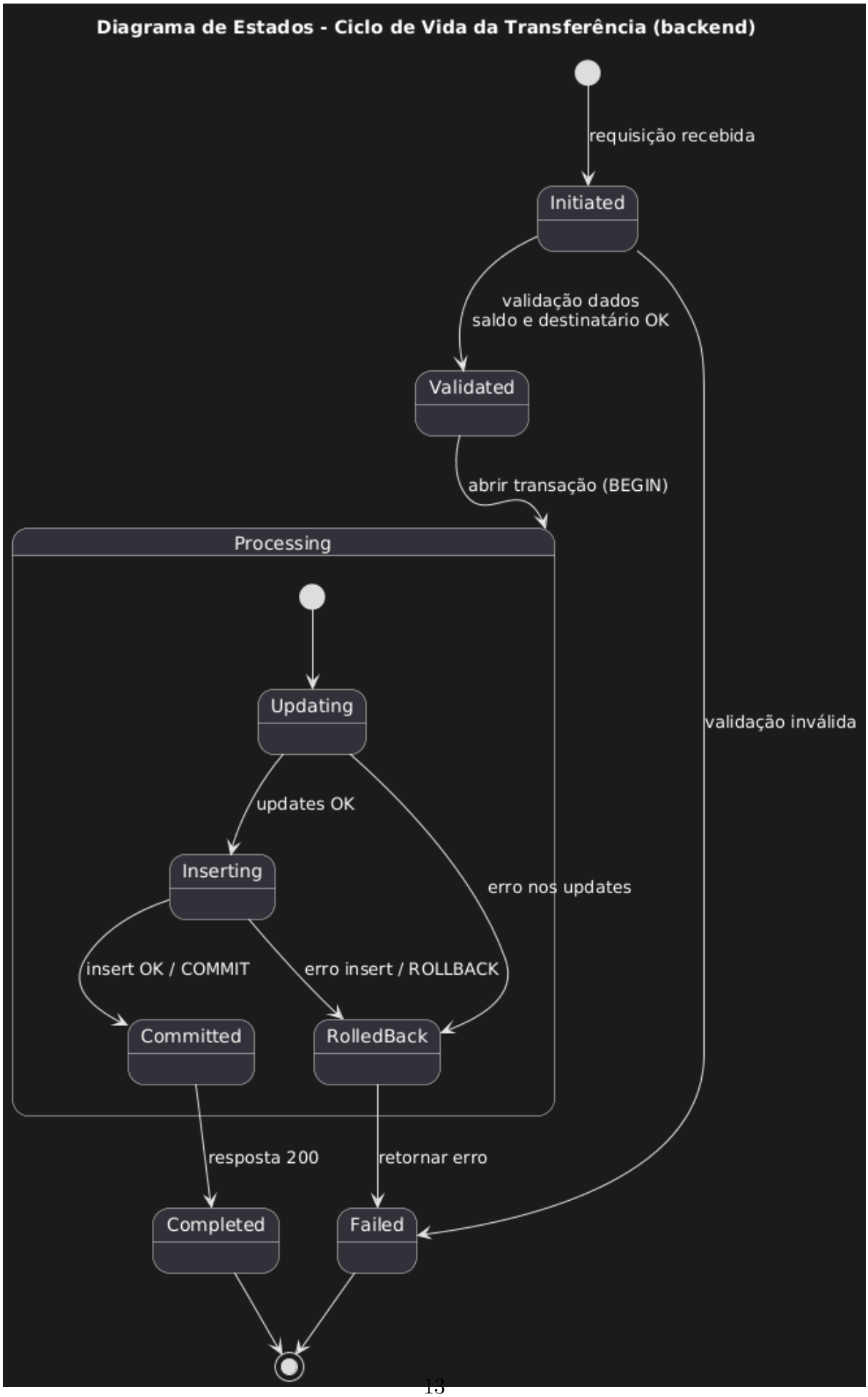


Diagrama de Estados - Ciclo de Vida da Transferência (backend)
requisição recebida
Initiated
validação dados
saldo e destinatário OK
Validated
abrir transação (BEGIN)
Processing
Updating
updates OK
Inserting
erro nos updates
insert OK / COMMIT
erro insert / ROLLBACK
Committed
RolledBack
resposta 200
retornar erro
validação inválida
Completed
Failed

4 Modelos de Dados