Documentação de Projeto

para o sistema

Gerenciamento de Inadimplentes

Versão 1.2

Projeto de sistema elaborado pelo aluno Matheus Pereira e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo do professor Cleiton Silva Tavares, Danilo de Quadros Mais Filho , Leonardo Vilela Cardoso e Raphael Ramos Dias Costas, orientação acadêmica do professor Cleiton e orientação de TCC II do professor (a ser definido no próximo semestre).

20/08/2025

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Matheus Pereira	28/09/25	Preenchimento das seções 1 até 2.1	1.0
Matheus Pereira	09/10/25	Preenchimento das seções 2.1,2,2,2.3,3.1, 3.2, 3.3	1.1
Matheus Pereira	02/11/25	Preenchimento das seções 3.4,3.5,3.6 e 5	1.2

1. Introdução

O presente documento contempla a elaboração e a revisão de modelos de domínio, bem como a definição de modelos de projeto referentes ao Sistema de Gerenciamento de Inadimplentes. Trata-se de uma aplicação voltada ao controle e acompanhamento de clientes inadimplentes, permitindo o registro de débitos, o monitoramento de pagamentos em atraso e a emissão de relatórios gerenciais que apoiam a tomada de decisão.Tais modelos têm como objetivo fornecer uma representação estruturada dos principais elementos do sistema, suas interações e responsabilidades, subsidiando, assim, o processo de desenvolvimento da aplicação de maneira consistente e alinhada aos requisitos previamente levantados.

2. Modelos de Usuário e Requisitos

Esta seção tem como objetivo descrever os usuários e atores do sistema, assim como os requisitos aos quais esse deve atender. Para isso, é apresentada uma breve descrição de cada ator, assim como um modelo de caso de uso e as histórias de usuários relacionadas, ambos que servem de referência para o desenvolvimento do sistema.Por último,são apresentados o diagrama de sequência do sistema.

2.1 Descrição de Atores

Nesta subseção, são apresentados os atores que interagem com o Sistema de Gerenciamento de Inadimplentes. Cada ator possui responsabilidades específicas, que estão diretamente relacionadas ao uso e às funcionalidades disponibilizadas pela aplicação.

Responsável Financeiro

O Responsável Financeiro constitui-se como o principal usuário do sistema. Compete a este ator o cadastramento de clientes inadimplentes, o registro de pagamentos realizados, a atualização das informações financeiras e a emissão de relatórios gerenciais. Além disso, é incumbido de efetuar a autenticação no sistema, garantindo a segurança do acesso às informações. Sua participação é essencial para assegurar a confiabilidade dos dados e a efetividade do controle da inadimplência, uma vez que concentra a maior parte das operações críticas do sistema.

Proprietária do Escritório de Contabilidade

A Proprietária do Escritório de Contabilidade interage com o sistema de forma complementar, assumindo um papel de caráter gerencial. Suas atividades concentram-se na consulta a clientes inadimplentes, na visualização de relatórios financeiros e na análise de indicadores disponibilizados em dashboards. A partir destas informações, a proprietária obtém uma visão consolidada do cenário financeiro do escritório, o que subsidia a definição de estratégias de cobrança e o planejamento administrativo.

2.2 Modelos de Usuários

Nesta seção são apresentadas as personas que representam os principais usuários do Sistema de Gerenciamento de Inadimplentes. As personas foram elaboradas a partir da análise do contexto do escritório de contabilidade, dos processos de negócio e das necessidades identificadas no levantamento de requisitos.

A Tabela 1 descreve a persona do usuário Cláudia Pereira e Silva, proprietária e gestora administrativa do escritório. Nela é possível identificar que Cláudia sente falta de relatórios claros e consolidados, o que dificulta a identificação rápida dos clientes que mais possuem dívidas. Também é perceptível que ela busca acompanhar indicadores de inadimplência e ter uma visão consolidada da situação financeira. Por fim, após análise é possível ver que Cláudia deseja obter relatórios periódicos, dashboards com indicadores financeiros e uma visão geral da inadimplência que apoie suas decisões estratégicas.

Persona 1: Claudia Pereira	
Idade:	55 anos
Cargo:	Proprietária e gestora administrativa
Objetivos:	Acompanhar indicadores de inadimplência, ter uma visão consolidada da situação financeira e apoiar decisões estratégicas do escritório.
Frustrações:	Falta de relatórios claros e consolidados; dificuldade de identificar rapidamente os clientes que mais devem.
Necessidades:	Obter relatórios periódicos, dashboards com indicadores financeiros e visão geral da inadimplência.

Tabela 1. Persona Claudia Pereira

A Tabela 2 descreve a persona do usuário José Carlos, responsável pelo setor financeiro. Nela é possível identificar que José, apesar de ser experiente na área financeira, ainda depende de anotações manuais em cadernos, o que aumenta o risco de falhas e dificulta a atualização em tempo real dos registros. Também é perceptível que ele busca reduzir erros manuais e gerar relatórios confiáveis para apoiar a gestão do escritório. Por fim, após análise é possível ver que José deseja um sistema simples, seguro e de fácil usabilidade, que permita cadastrar inadimplentes, registrar pagamentos e emitir relatórios precisos.

Persona 2: José Carlos	
Idade:	55 anos
Cargo:	Responsável pelo setor financeiro
Objetivos:	Manter controle atualizado sobre clientes inadimplentes; reduzir erros manuais; gerar relatórios para apoiar a gestão.
Frustrações:	Atualmente depende de anotações em cadernos, o que aumenta o risco de falhas, dificulta a atualização em tempo real e compromete a precisão dos registros.
Necessidades:	Sistema simples, seguro e de fácil usabilidade, que permita cadastrar inadimplentes, registrar pagamentos e emitir relatórios confiáveis.

Tabela 2. Persona José Carlos

2.3 Modelo de Casos de Uso e Histórias de Usuários

Esta subseção tem como objetivo descrever os casos de uso e as histórias de usuário previstos para o projeto. Para tanto, apresenta-se um diagrama de casos de uso em que todos são listados. De forma complementar, também são apresentadas as histórias de usuário relacionadas às funcionalidades previstas para o sistema a ser desenvolvido.

2.3.1 Diagrama de caso de uso

O diagrama de casos de uso (figura 1) representa as funcionalidades do sistema financeiro do escritório, com dois atores principais: o Responsável Financeiro e a Proprietária do Escritório. O Responsável Financeiro realiza operações como registro de pagamentos, consulta e cadastro de clientes inadimplentes, geração de relatórios, exportação de dados, autenticação e visualização de indicadores. A ação de registrar pagamento inclui automaticamente a atualização do saldo devedor. Já a Proprietária do Escritório possui acesso à visualização do dashboard e à geração de relatórios, com foco em análise e tomada de decisão.

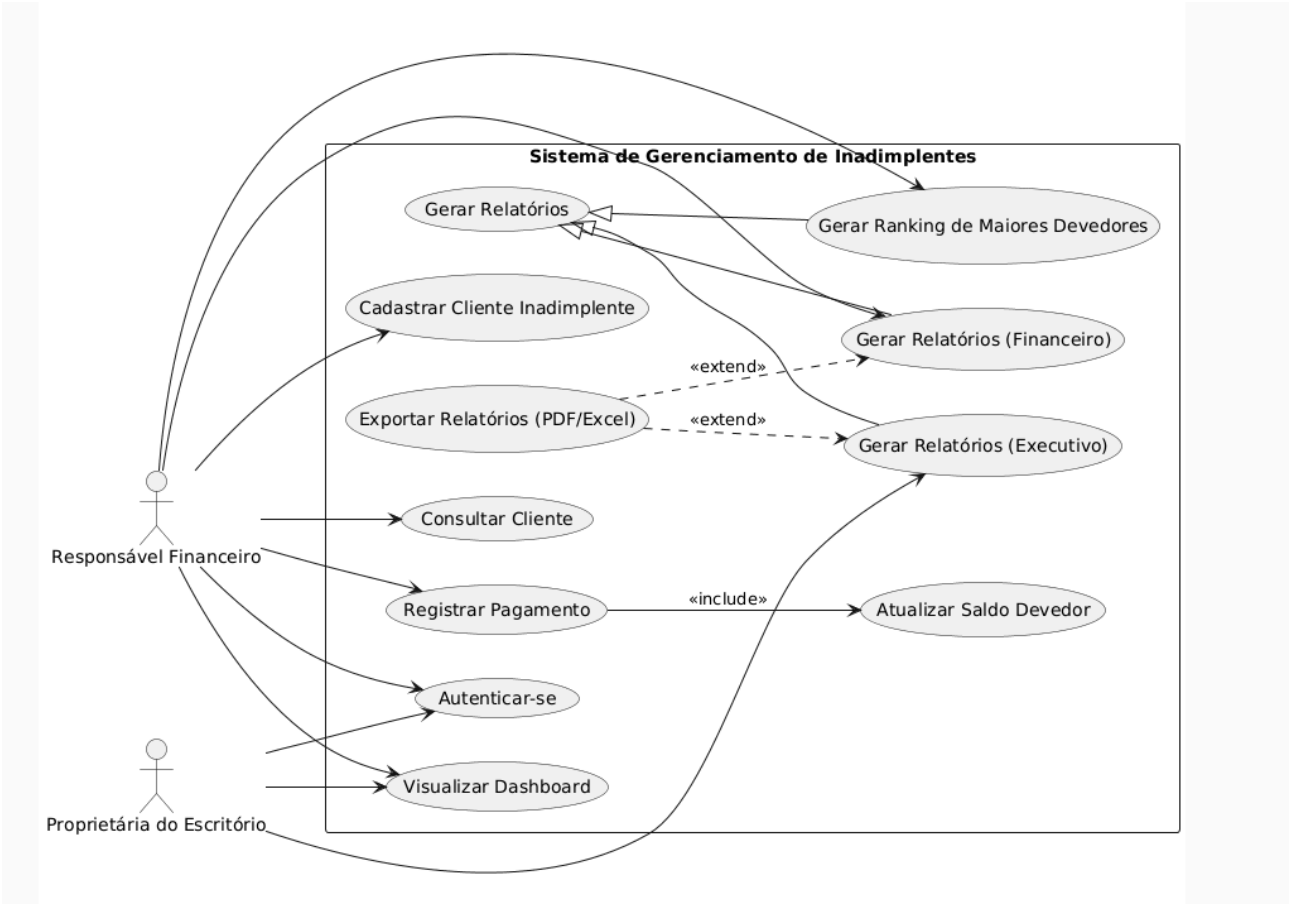


Figura 1. Diagrama de Caso de Uso

2.3.2 Historias do Usuario

Nesta seção são listadas as histórias de usuários elicitadas para o sistema financeiro proposto. Para fins de organização, utiliza-se o identificador no formato US#ID, em que US refere-se a *User Story*.

As histórias de usuários identificadas para o sistema são:

US1. Como responsável financeiro, eu gostaria de registrar pagamentos de clientes para manter o controle atualizado das pendências.

US2. Como responsável financeiro, eu gostaria de atualizar automaticamente o saldo devedor ao registrar um pagamento, para garantir consistência nas informações.

US3. Como responsável financeiro, eu gostaria de cadastrar clientes inadimplentes para controlar melhor os casos de atraso.

US4. Como responsável financeiro, eu gostaria de gerar relatórios financeiros para análise interna.

US5. Como responsável financeiro, eu gostaria de exportar relatórios em PDF ou Excel para facilitar o compartilhamento e arquivamento.

US6. Como responsável financeiro, eu gostaria de visualizar o *ranking* de maiores devedores para identificar os principais inadimplentes.

US7. Como responsável financeiro, eu gostaria de consultar dados dos clientes para facilitar o atendimento e a tomada de decisões.

US8. Como responsável financeiro, eu gostaria de gerenciar autenticação para garantir o acesso seguro ao sistema.

US9. Como proprietária do escritório, eu gostaria de visualizar um *dashboard* com indicadores financeiros para acompanhar a saúde do negócio.

US10. Como proprietária do escritório, eu gostaria de gerar relatórios sempre que necessário para entender o desempenho financeiro da empresa.

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Esta seção descreve os principais fluxos de interação do SGI – Sistema de Gerenciamento de Inadimplentes por meio de diagramas de sequência, detalhando a ordem das mensagens entre App (usuário/Responsável Financeiro), Backend (API) e serviços de domínio. Cada fluxo possui um Contrato de Operação com pré-condições, pós-condições e exceções. As legendas e referências seguem o padrão do documento.

O fluxo de autenticação inicia quando o usuário informa suas credenciais no app; o backend delega a verificação ao AuthService, que consulta o *UsuarioRepository*; com credenciais válidas, o serviço emite o tokenSessao e identifica o perfil; por fim, o backend retorna 200 OK ao app com o token e o perfil do usuário.

Contrato	Autenticar Usuário
Operação	autenticar(telefone: String, senhaOuOTP: String) -> token: String, perfil: Perfil
Referências cruzadas	US8 – Gerenciar autenticação (acesso seguro ao sistema)
Pré-condições	App online; credenciais informadas; usuário existente/ativo.
Pós-condições	Sessão ativa com tokenSessao válido; perfil associado; auditoria “login_sucesso”.

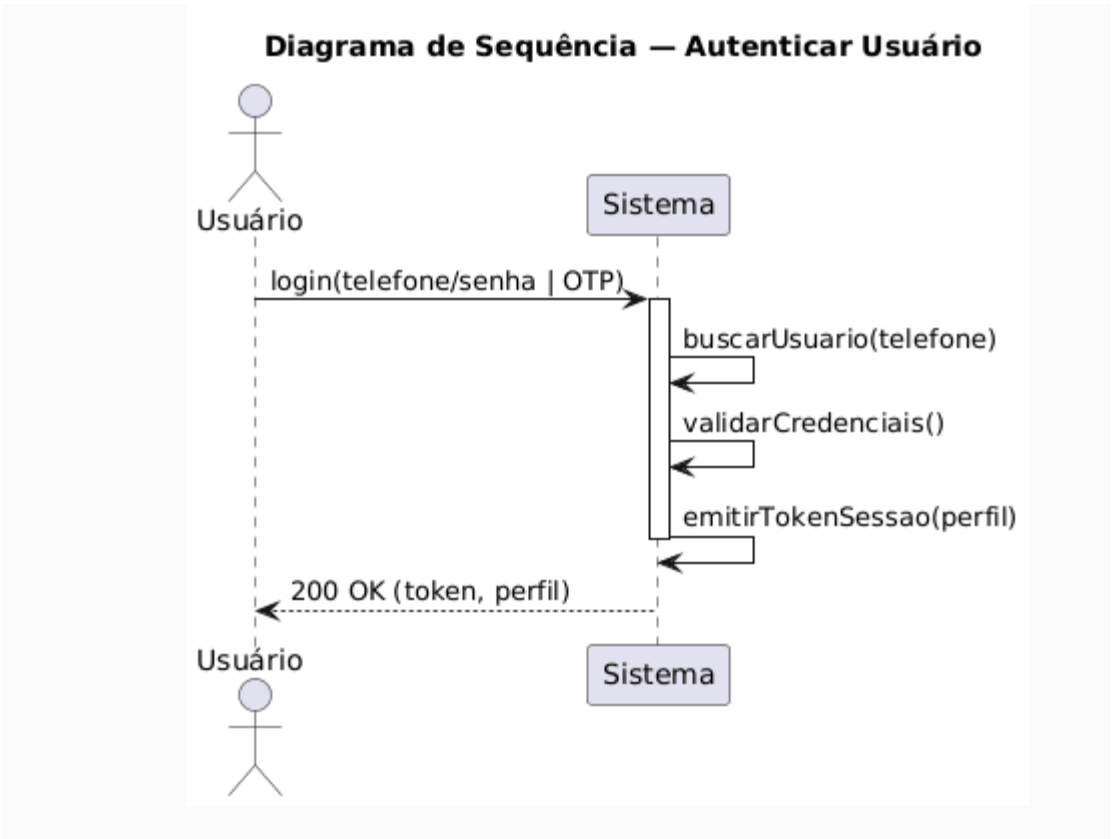


Figura 2 - Autenticar Usuario

Conforme será mostrado na Figura 3. O Responsável Financeiro seleciona o cliente e envia os dados do pagamento; no backend, o PagamentoService é acionado para recuperar o cliente, persistir o pagamento e, no mesmo fluxo transacional, recalcular o saldo devedor no repositório; ao final, a API retorna ao app o recibo e o novo saldo confirmado.

Contrato	Registrar Pagamento
Operação	registrarPagamento(clienteId: UUID, valor: Money, data: Date, comprovante: Blob) -> recibo: PDF, novoSaldo: Money
Referências cruzadas	US1 e US2
Pré-condições	Usuário autenticado; clienteId existente/ativo; valor > 0; data ≤ hoje.
Pós-condições	Pagamento persistido; saldo recalculado; recibo emitido; auditoria "pagamento_registrado".

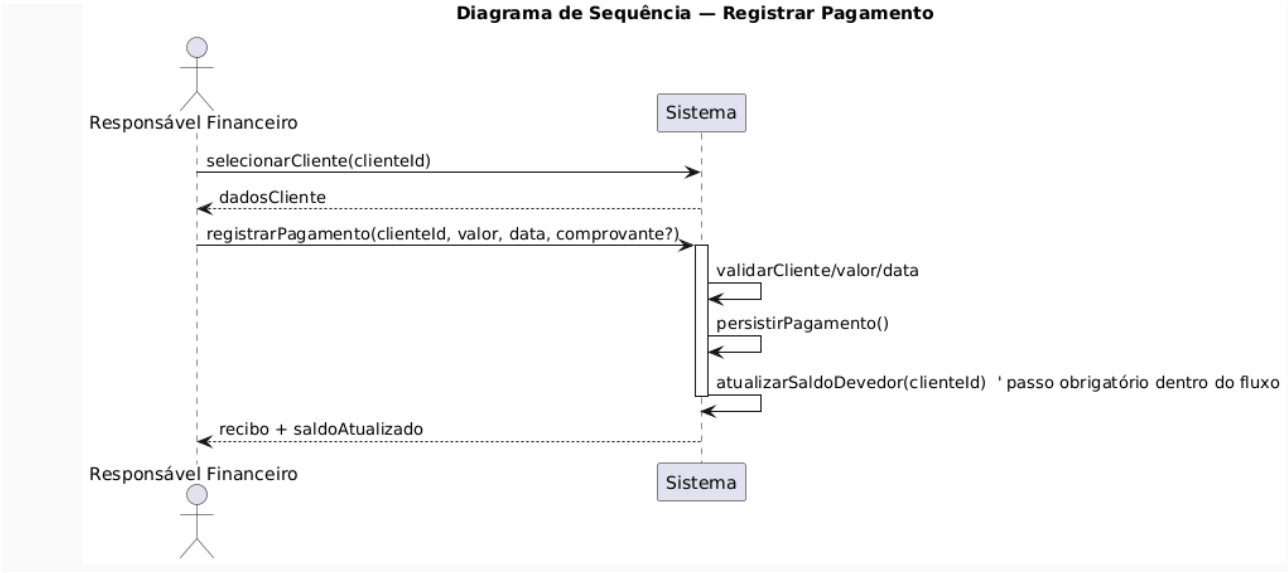


Figura 3 - Registrar Pagamento

Conforme será mostrado na Figura 4, para um cliente previamente validado o responsável financeiro registra uma nova inadimplência informando valor, vencimento e descrição; o InadimplenciaService aplica as regras de negócio, valida os dados e persiste o lançamento com status inicial “Em Aberto”, gerando o respectivo protocolo; em seguida, o backend confirma ao aplicativo a criação do registro (incluindo o identificador/protocolo) e registra a trilha de auditoria do evento, com data-hora, usuário e origem da operação.

Contrato	Registrar Inadimplência
Operação	registrarInadimplencia(clienteId: UUID, valor: Money, vencimento: Date, descricao?: String) -> protocolo: String
Referências cruzadas	US3
Pré-condições	clienteId existente/ativo; valor > 0; vencimento ≥ hoje; usuário autenticado.
Pós-condições	Inadimplência criada (status = EmAberto); auditoria “inadimplencia_registrada”.

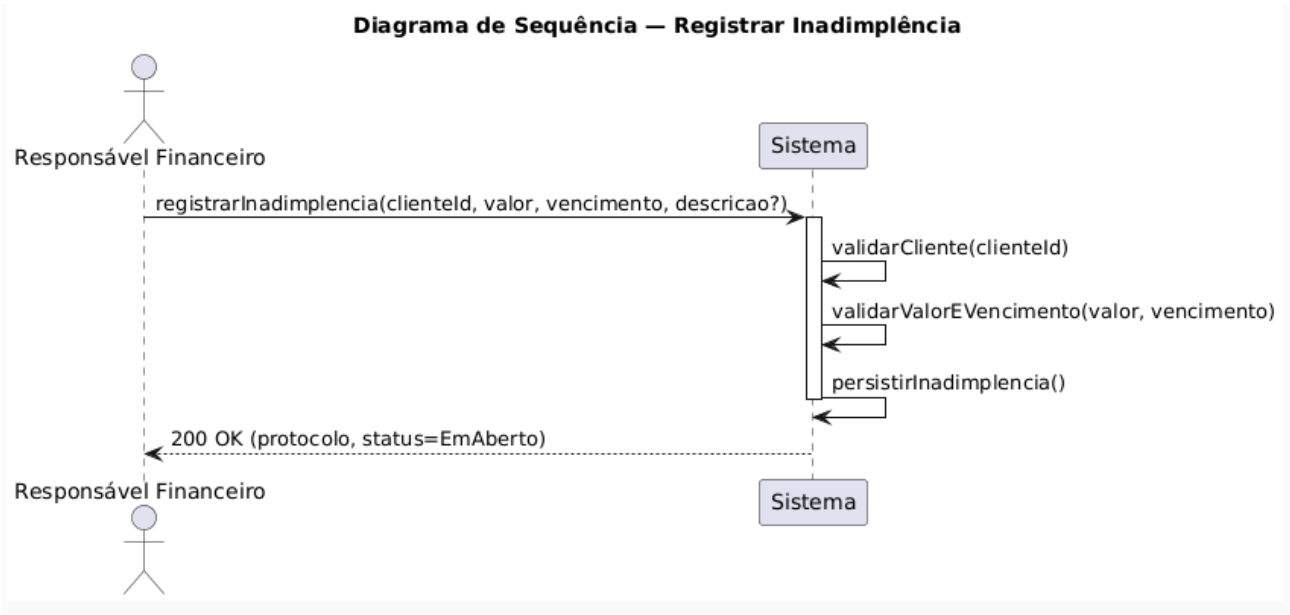


Figura 4 — Registrar Inadimplência.

Conforme será mostrado na Figura 5, o usuário solicita a geração de um relatório (por exemplo, ranking de maiores devedores ou resumo por período), parametrizando os filtros desejados; o RelatorioService consulta o repositório, consolida os dados e retorna o dataset/visão para exibição no aplicativo; opcionalmente, o usuário pode acionar a exportação em PDF ou Excel, gerando o arquivo correspondente para download e arquivamento.

Contrato	Gerar Relatório
Operação	gerarRelatorio(tipo: Enum, periodo: {ini, fim}, filtros?: Map) -> view/dataset
Referências cruzadas	US4,US5,US6
Pré-condições	Usuario autenticado; filtros válidos.
Pós-condições	Resultado exibido; se exportado, arquivo gerado (PDF/Excel) e link disponibilizado

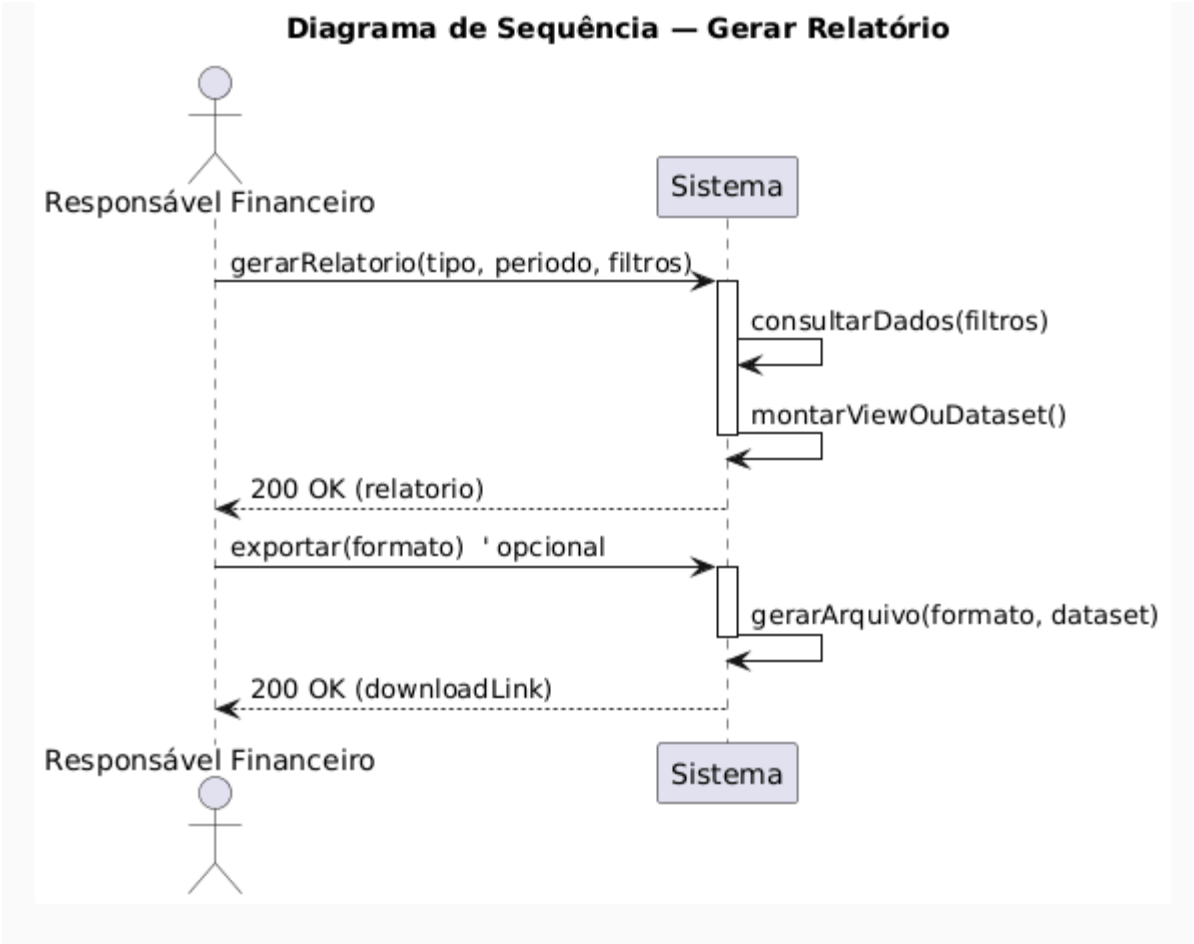


Figura 5 — Gerar Relatório.

Conforme será mostrado na Figura 6, o fluxo de recálculo de saldo consolida os débitos em aberto e os pagamentos efetivados de um cliente, podendo ser acionado manualmente pela interface do usuário ou automaticamente como processo interno subsequente à confirmação de pagamento; ao término da execução, o novo saldo é persistido no repositório de dados e retornado ao aplicativo, assegurando a consistência entre os lançamentos registrados e a visão financeira apresentada ao usuário.

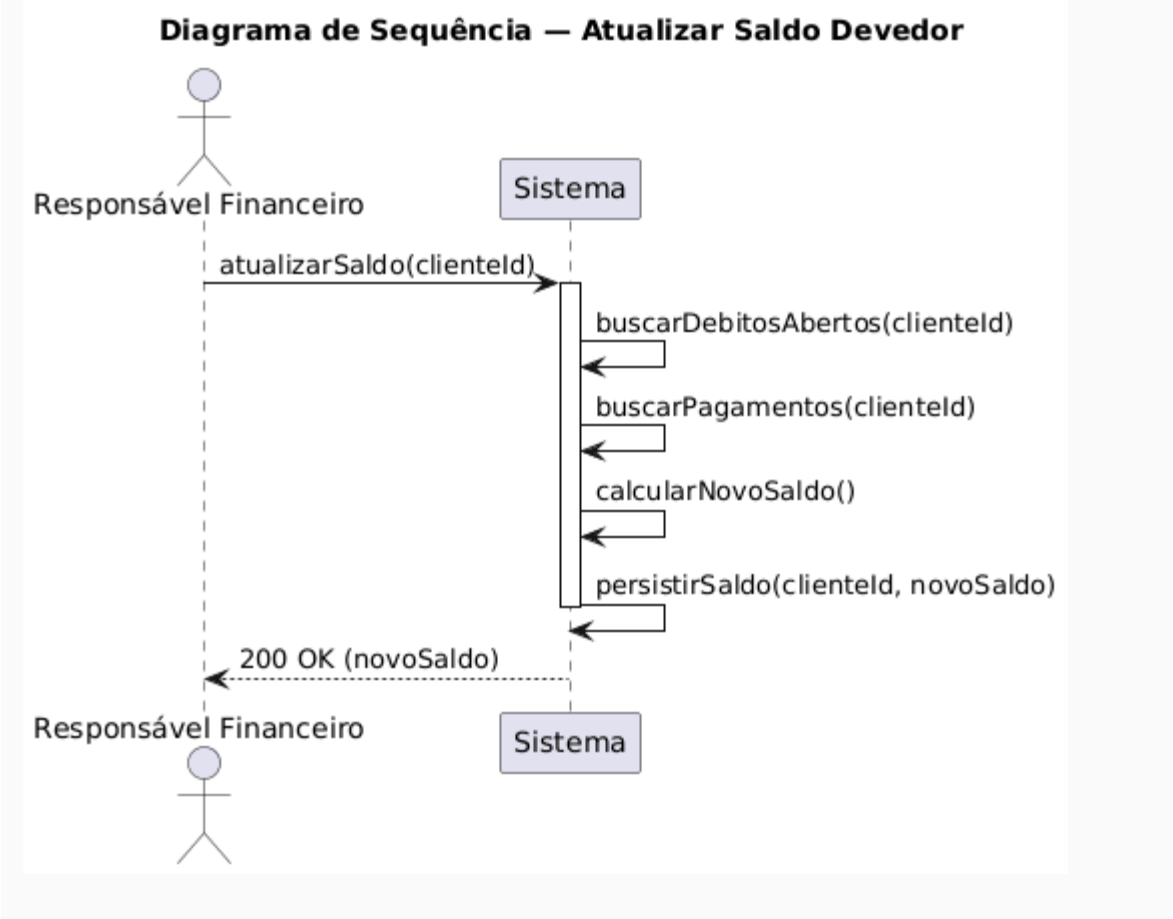


Figura 6 — Atualizar saldo devedor

Contrato	Atualizar Saldo Devedor
Operação	atualizarSaldoDevedor(clienteId)
Referências cruzadas	US2, atualizar saldo devedor.
Pré-condições	Usuário autenticado (na execução manual); *clienteId* válido.
Pós-condições	novoSaldo calculado e persistido; auditoria “saldo_recalculado”.

3. Modelos de Projeto

Esta seção apresenta os modelos de projeto do sistema proposto, organizados nas seguintes subseções: Visão Geral dos Modelos (Seção 3.1), Diagrama de Classes (Seção 3.2), Diagramas de Sequência (Seção 3.3), Diagrama de Comunicação (Seção 3.4), Arquitetura (Seção 3.5), Diagramas de Estados (Seção 3.6) e Diagramas de Componentes e Implantação (Seção 3.7). Em conjunto, esses artefatos oferecem uma visão integrada das estruturas, dos comportamentos e da distribuição do sistema.

Modelos de Projeto

3.1 Diagrama de Classes

A Figura 6 apresenta o diagrama de classes do núcleo de domínio do sistema de gerenciamento de inadimplentes, destacando as entidades centrais, seus principais atributos e relacionamentos. A classe Cliente representa pessoas físicas ou jurídicas cadastradas, com dados básicos e um atributo status que indica se está adimplente ou inadimplente, conforme o enum StatusCliente. A classe Divida representa os títulos em aberto de cada cliente, contendo informações como valor, vencimento e status definido pelo enum StatusDivida (EM_ABERTO, PARCIAL, QUITADA ou VENCIDA), além de métodos para registrar pagamentos e atualizar saldos. Já a classe Pagamento registra valores, datas e métodos de pagamento, permitindo múltiplos pagamentos por dívida, contemplando liquidações parciais ou totais.

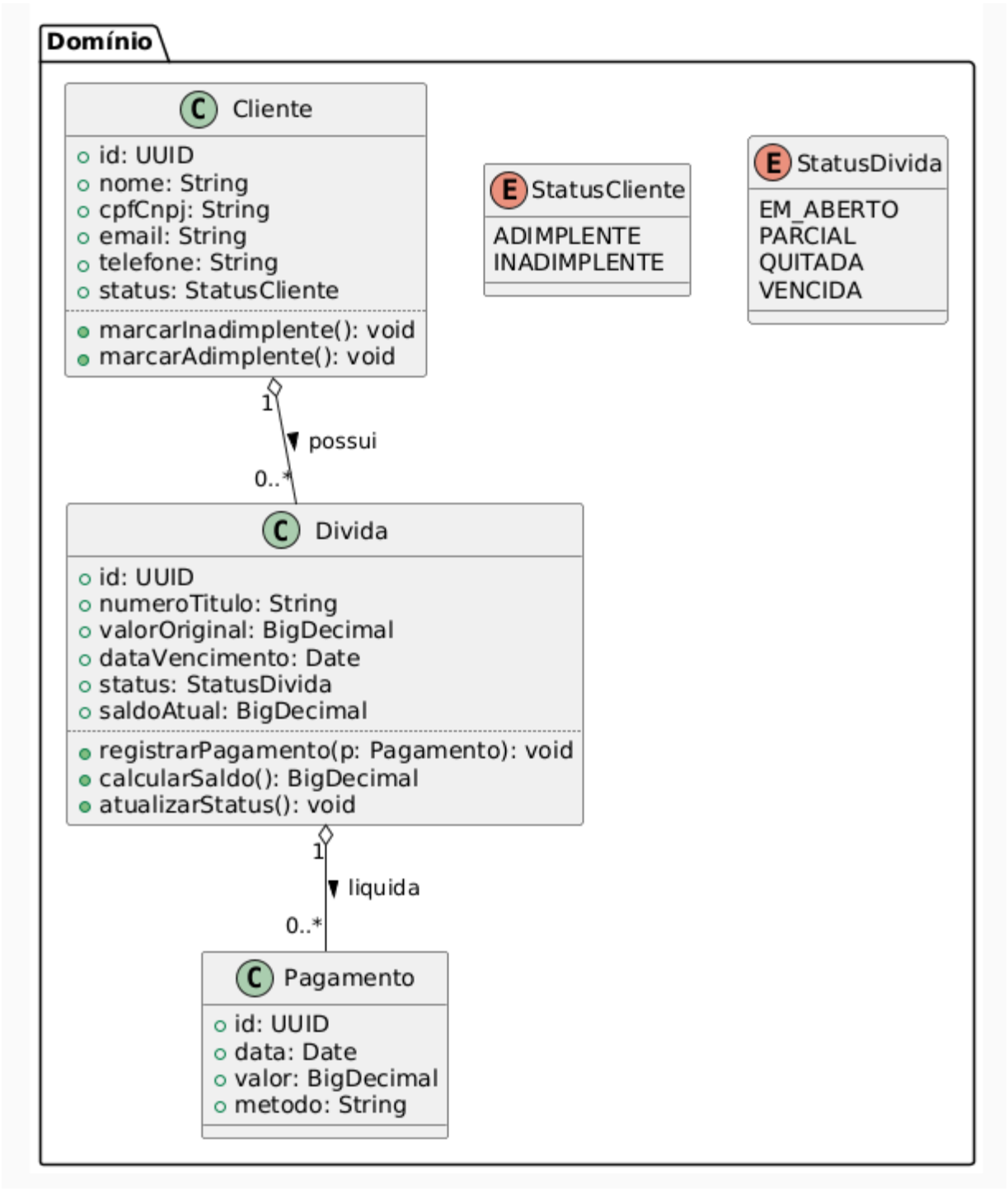


Figura 7 — Diagrama de Classes

3.2 Diagramas de Sequência

A Figura 7 apresenta o diagrama de sequência referente ao fluxo principal de registro de pagamento no sistema de gerenciamento de inadimplentes. Ele descreve a interação entre o Responsável Financeiro, a Aplicação Web, o Serviço de Cobrança e os repositórios de dados. O processo inicia com a solicitação de registro de pagamento, seguida da validação dos dados e da verificação do status da dívida. Caso os dados sejam inválidos ou a dívida esteja quitada, uma mensagem de erro é exibida ao usuário. Se estiver tudo correto, o sistema registra o pagamento, atualiza o saldo e o status da dívida, e ajusta o status do cliente conforme a situação financeira. Ao final, uma confirmação com o novo saldo é retornada ao responsável, garantindo consistência e rastreabilidade da operação.

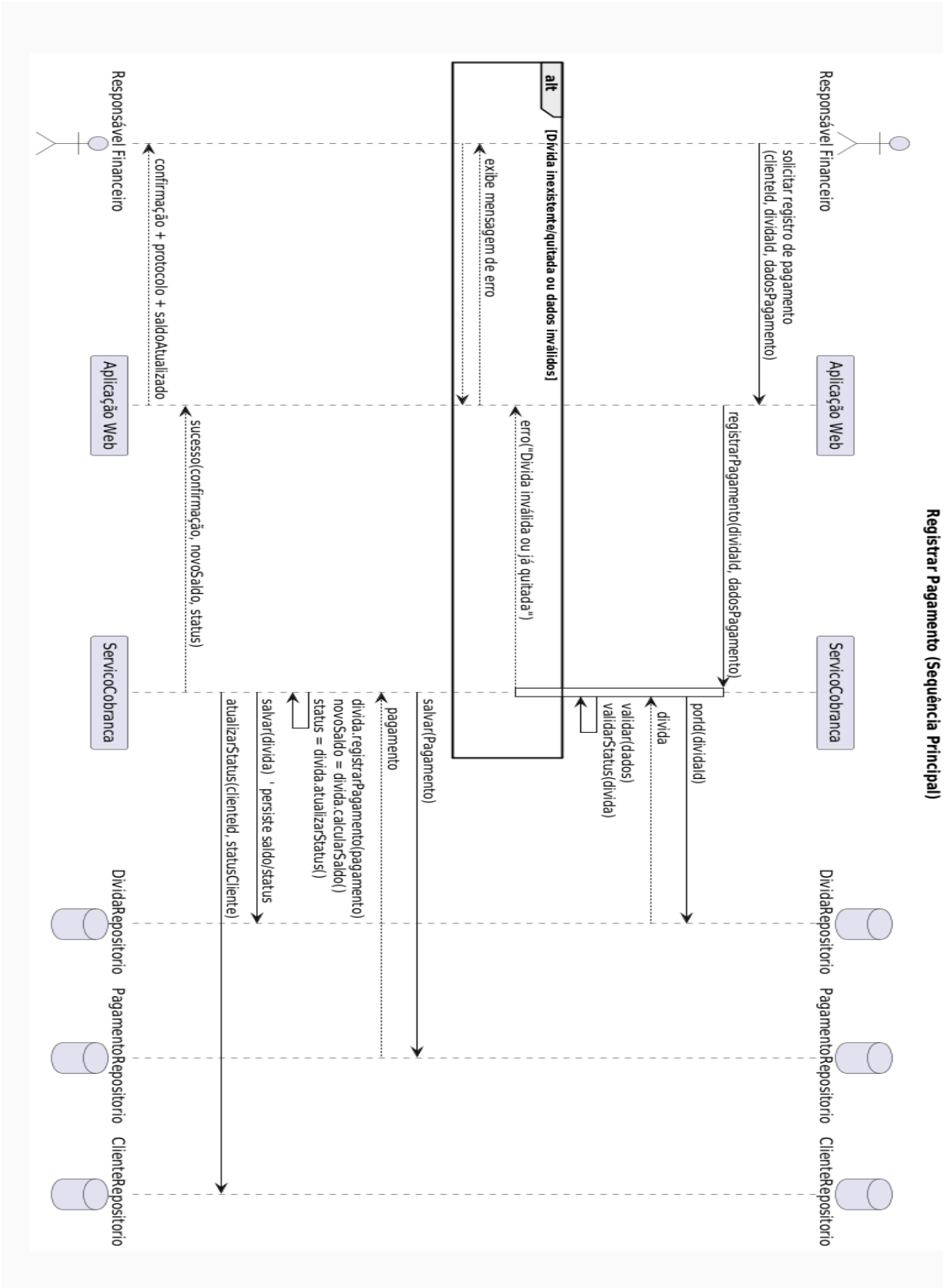


Figura 8— Diagrama de Sequencia

A Figura 8 apresenta o diagrama de sequência do cadastro de dívidas no sistema de gerenciamento de inadimplentes. O processo inicia quando o Responsável Financeiro solicita o registro de uma nova dívida pela aplicação web. O sistema então valida o cliente e os dados informados. Caso haja inconsistências, é exibida uma mensagem de erro. Se estiver tudo correto, a dívida é persistida no repositório e uma confirmação de sucesso com o protocolo de registro é retornada, garantindo a integridade das informações financeiras.

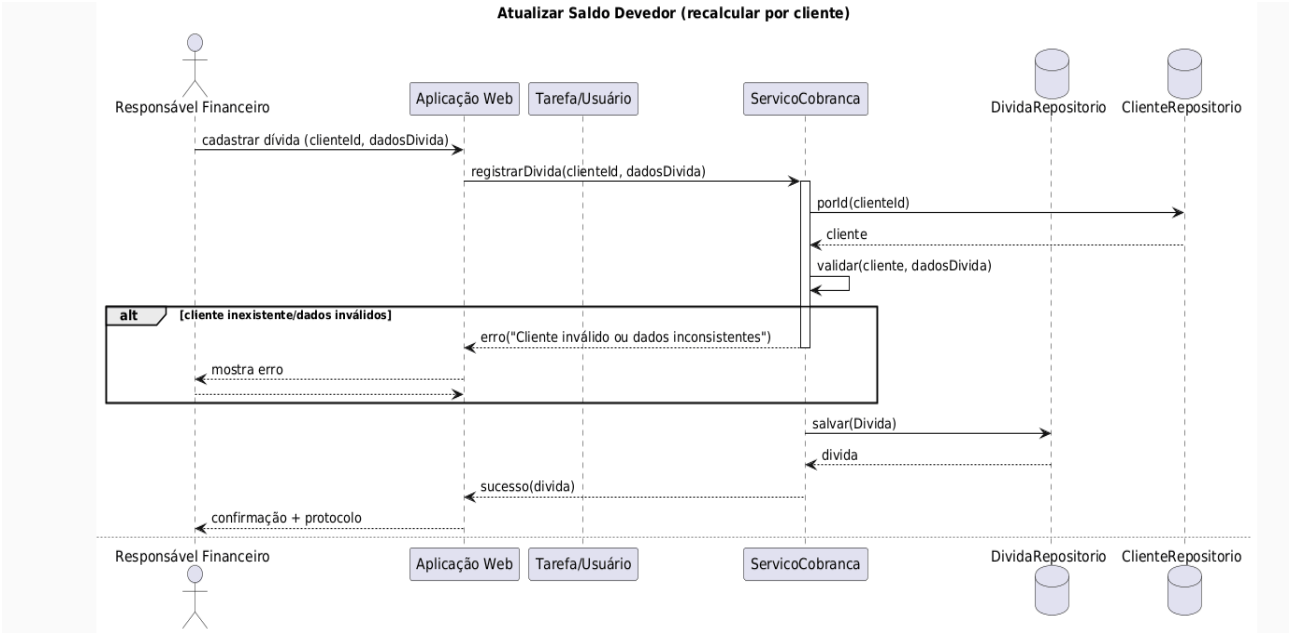


Figura 9 — Diagrama de Sequência

3.3 Diagramas de Comunicação

Nesta seção, são apresentados os diagramas de comunicação que modelam as trocas de mensagens entre atores, camadas e serviços do SGI – Sistema de Gerenciamento de Inadimplentes. O objetivo é documentar quem se comunica com quem e em que ordem, cobrindo os principais fluxos: autenticação, registro de pagamento com atualização de saldo, consulta de clientes/inadimplências e geração/expedição de relatórios. Esses diagramas complementam os diagramas de sequência e servem como referência dos contratos de mensagem da aplicação.

A Figura 1 apresenta as mensagens envolvidas no registro de pagamento de um cliente inadimplente. O Responsável Financeiro inicia o processo pela UI-Web, que valida a sessão no AuthService e encaminha a solicitação ao PagamentoService. Esse consulta o ClienteService para obter o cliente, persiste o pagamento no Repositório e, obrigatoriamente, dispara a atualização do saldo devedor (comportamento interno reutilizado). Ao final, o serviço retorna a confirmação e recibo à interface, que exibe o resultado ao usuário.

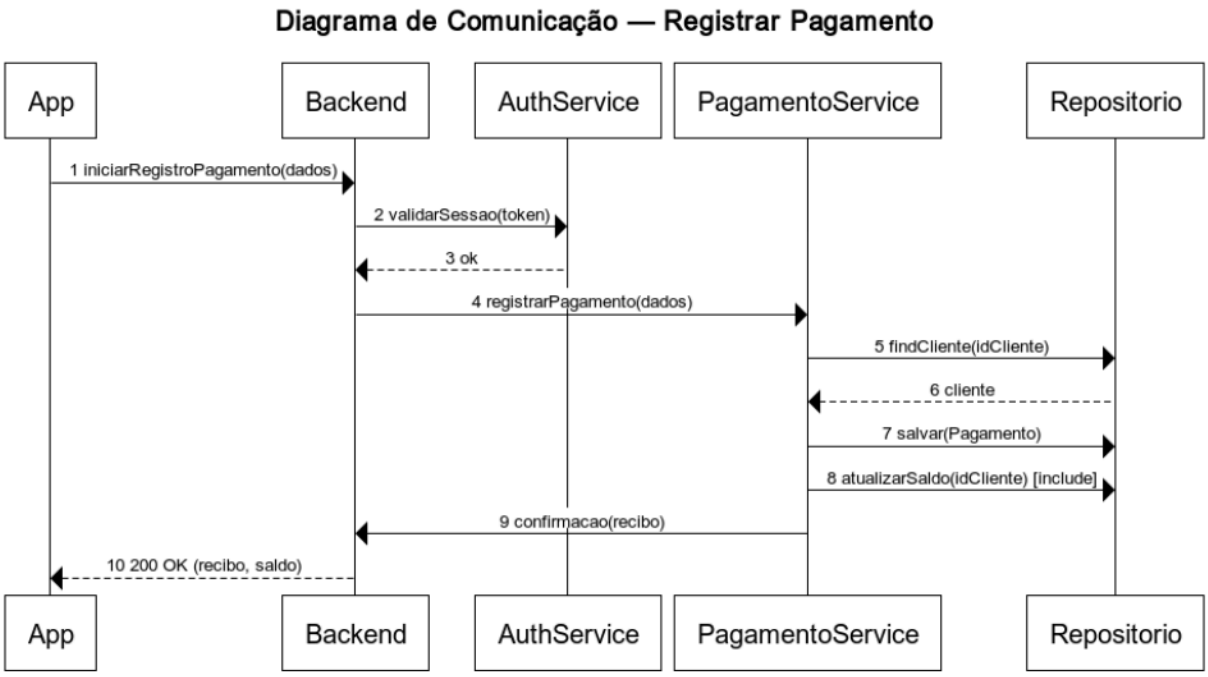


Figura 10 — Diagrama de Comunicação

Figura 2 – Diagrama de Comunicação: Gerar Relatórios (Financeiro) e Exportar (PDF/Excel)

A Figura 2 descreve o fluxo de geração de relatórios financeiros. O Responsável Financeiro solicita o relatório à UI-Web, que chama o RelatorioService. O serviço consulta dados no Repositório e devolve o conjunto resultante para exibição. Opcionalmente, quando o usuário seleciona Exportar, a interface aciona o *ExportService* para produzir o arquivo em PDF ou Excel, disponibilizando o download ao término.

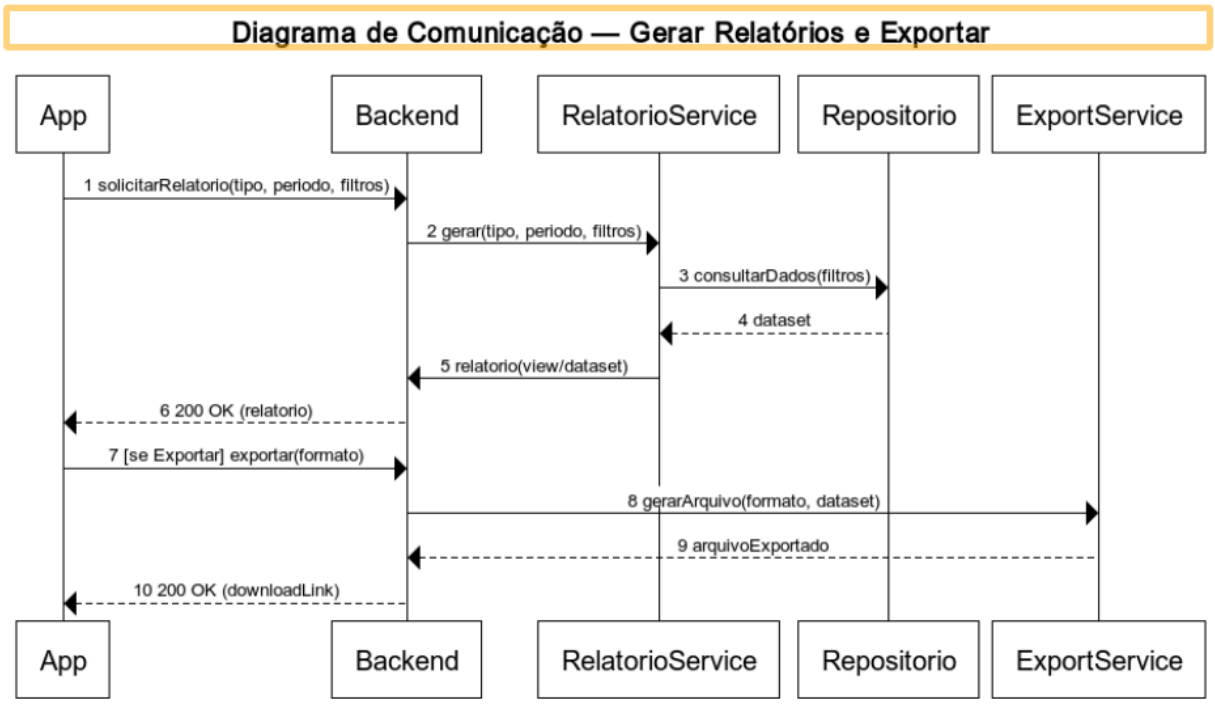


Figura 11 — Diagrama de Comunicação

Figura 3 – Diagrama de Comunicação: Consultar Cliente/Inadimplência (com cache e iteração)

A Figura 3 mostra as mensagens trocadas na consulta de clientes e suas inadimplências. O Responsável Financeiro solicita a busca à UI-Web, que encaminha ao ClienteService. Quando habilitado, o serviço consulta primeiro o Cache; em caso de cache miss, realiza query no Repositório. Em seguida, para cada cliente retornado, o serviço itera a obtenção dos detalhes de inadimplência, consolidando o resultado para exibição.

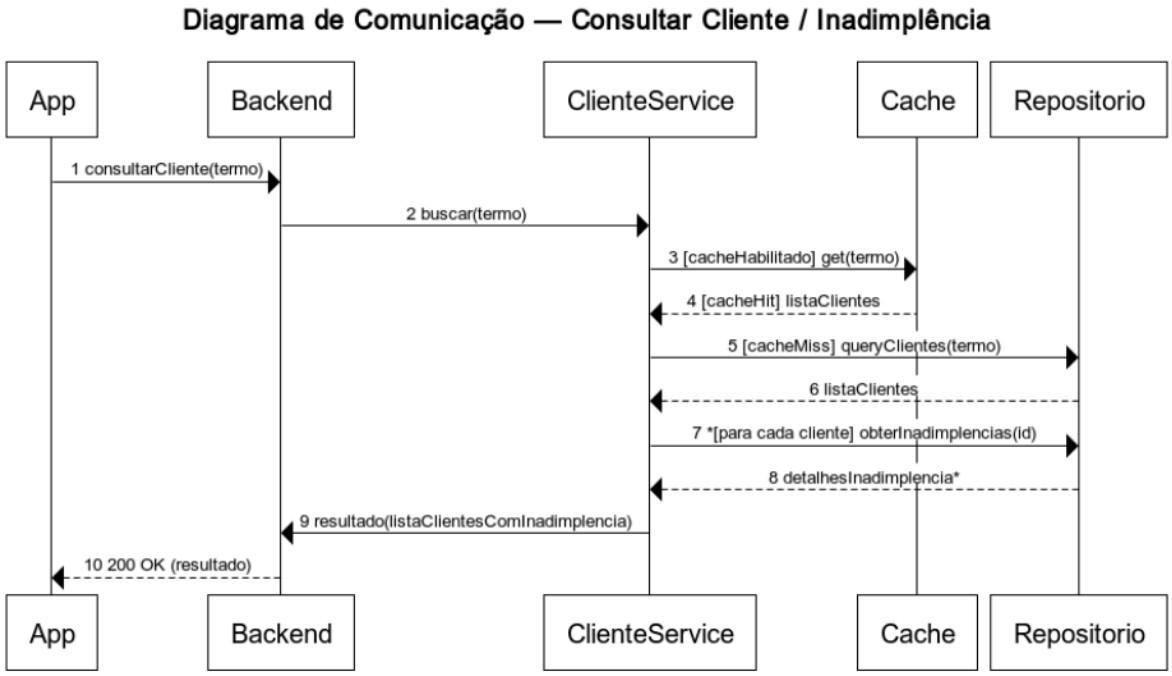


Figura 12 — Diagrama de Comunicação

3.4 Arquitetura

A arquitetura do sistema está organizada em três camadas principais — UI, API e Data — com separação explícita de responsabilidades. Na camada de UI, o repositório contempla duas frentes clientes: um Frontend Web (React + Vite) e um Aplicativo Desktop (Electron + React). Ambas as interfaces consomem a API por HTTP/JSON (REST), exibindo informações ao usuário e capturando ações como cadastro, consulta e edição de dados.

A camada de API é implementada em Spring Boot (Java 21) e concentra a orquestração de regras de negócio e a exposição dos endpoints REST. Nela, os Controllers atuam como pontos de entrada da aplicação, recebendo as requisições dos clientes, acionando a lógica de domínio e retornando respostas padronizadas. O acesso a dados é encapsulado por Repositories (Spring Data JPA), que abstraem consultas e operações CRUD. A organização em pacotes (por exemplo, um pacote de domínio como cliente contendo entidade, repository e controller) favorece coesão e facilita a evolução incremental de novas funcionalidades. Há também um pacote de configuração (ex.: CorsConfig) para políticas de CORS e demais ajustes transversais, além de integrações de suporte

como validação (Bean Validation) e documentação (springdoc/OpenAPI), que facilitam testes e inspeção dos endpoints durante o desenvolvimento.

Na camada de Data, o sistema utiliza SQLite como banco de dados local, acessado via JPA/Hibernate. As entidades modelam o domínio (por exemplo, Cliente), enquanto os Repositories executam persistência e consultas tipadas. Essa abordagem simplifica a configuração, reduz dependências externas e é adequada para cenários de implantação leve; caso a aplicação cresça em concorrência e volume, a troca por um SGDB servidor (PostgreSQL, MySQL, etc.) é direta, mantendo o mesmo contrato JPA.

Esse arranjo em UI → API → Data promove alta coesão dentro de cada módulo e baixo acoplamento entre camadas, o que melhora testabilidade, manutenibilidade e escalabilidade. Os Controllers permanecem finos, delegando persistência aos Repositories; e, à medida que as regras de negócio se tornarem mais ricas, recomenda-se introduzir uma camada de Services (ex.: ClienteService) para centralizar validações e transações, mantendo a API estável e o acesso a dados isolado. Assim, o projeto segue um padrão de arquitetura limpo e extensível, pronto para receber novos domínios (novos pacotes com Entity/Repository/Controller) sem comprometer o restante do sistema.

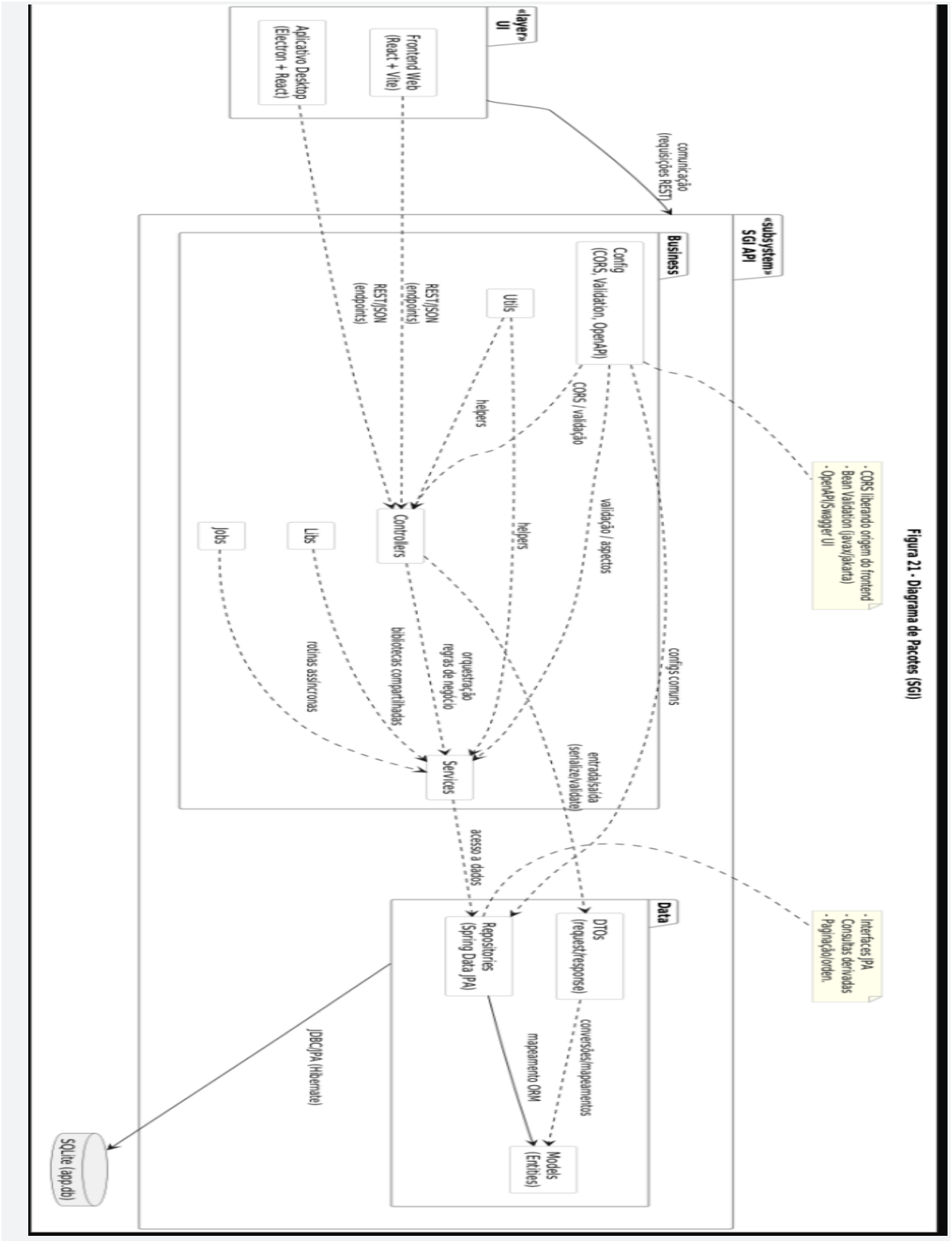


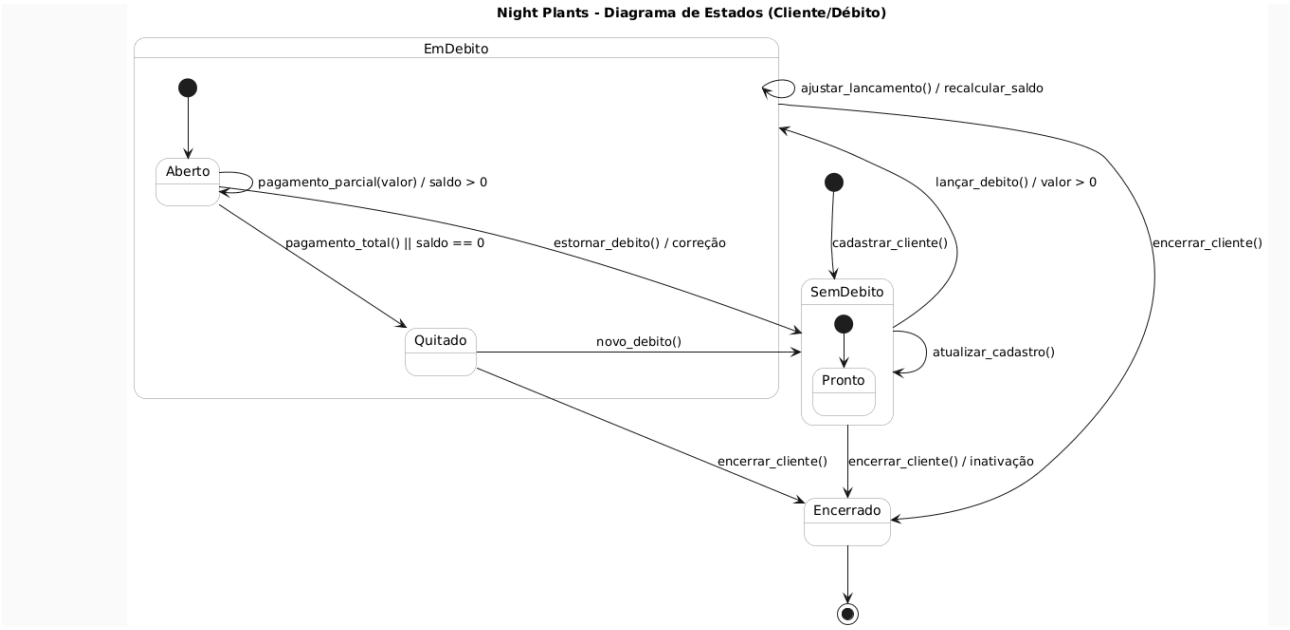
Figura 13 — Diagrama de pacote

3.5 Diagramas de Estados

O diagrama de estados modela o ciclo de vida do status de débito de um cliente no sistema, cujo escopo é apenas registrar e manter informações de inadimplência — sem ações de cobrança externa. O fluxo inicia em SemDébito, que representa o cliente cadastrado sem pendências financeiras. A partir da operação lançar_debito(), o estado transita para EmDébito, indicando a existência de saldo devedor. Dentro desse estado, o subestado Aberto captura a situação padrão do débito em acompanhamento, permitindo pagamento_parcial(valor) enquanto persistir saldo (saldo > 0) e evoluindo para Quitado quando ocorre pagamento_total() (ou quando saldo == 0 após ajustes). Em casos de correção operacional, estornar_debito() retorna o cliente a SemDébito. O ciclo pode ser encerrado a qualquer momento por encerrar_cliente(), levando ao estado Encerrado (inativação no sistema).

As transições foram definidas para refletir operações rotineiras e controladas: cadastrar_cliente() cria o registro inicial; lançar_debito() cria/atualiza lançamentos; pagamento_parcial() e pagamento_total() reduzem o saldo até a quitação; ajustar_lancamento() permite recalcular o saldo sem alterar o estado macro (EmDébito), preservando o histórico; estornar_debito() corrige lançamentos indevidos; encerrar_cliente() finaliza a relação ativa no sistema; e novo_debito() reabre o ciclo a partir de Quitado, retornando a SemDébito → EmDébito conforme necessário. Não há transições para eventos de protesto, negativação ou cobrança judicial, pois não fazem parte do escopo do Night Plants.

Esse modelo oferece clareza de regras, previsibilidade de transições e simplicidade de manutenção: cada estado representa uma situação operacional inequívoca (sem débito, em débito, quitado, encerrado), e cada evento corresponde a ações do usuário ou rotinas do sistema com efeitos bem definidos sobre o saldo. Além de facilitar testes e auditoria (por evidenciar quando e por que um status mudou), a máquina de estados serve como base para validações de negócio (por exemplo, impedir lançamentos em Encerrado) e para traçar métricas como tempo médio em EmDébito ou taxa de reversão por estorno, apoiando decisões de melhoria contínua.



.

Figura 14 — Diagrama de Estado

3.6 Diagrama de Componentes e Implantação.

O diagrama de componentes apresenta a decomposição lógica do sistema em elementos independentes e suas interfaces. A solução é organizada em três blocos: UI, API e Persistência.
Na camada de UI, há dois clientes: (i) o Frontend Web (React + Vite), executado no navegador, e (ii) o Aplicativo Desktop (Electron + React). Ambos se comunicam exclusivamente com a API por HTTP/JSON (REST), realizando operações de cadastro, consulta, atualização e encerramento de registros de devedores.

A camada de API é implementada com Spring Boot (Java 21) e expõe endpoints REST por meio de Controllers (por exemplo, DevedorController). Os Controllers delegam as regras de negócio ao Service (DevedorService), que centraliza validações e fluxos (lançar débito, ajustar lançamento, quitar, encerrar). O acesso a dados é encapsulado por Repositories (DevedorRepository, Spring Data JPA), enquanto as Entities/Models (como Devedor) representam o domínio persistido. Componentes transversais (Config) tratam de CORS, validação e documentação (OpenAPI). DTOs/Mapper padronizam entrada/saída e evitam acoplamento direto da API às entidades.

Na camada de Persistência, o sistema utiliza SQLite em arquivo único (app.db), acessado via JPA/Hibernate. Essa escolha reduz dependências operacionais e atende ao escopo do Night Plants — que somente registra devedores, sem cobrança externa, protesto ou integrações financeiras. O desenho favorece alta coesão (cada componente tem uma responsabilidade clara) e baixo acoplamento (UI ↔ API ↔ Dados), simplificando manutenção, testes e evolução incremental.

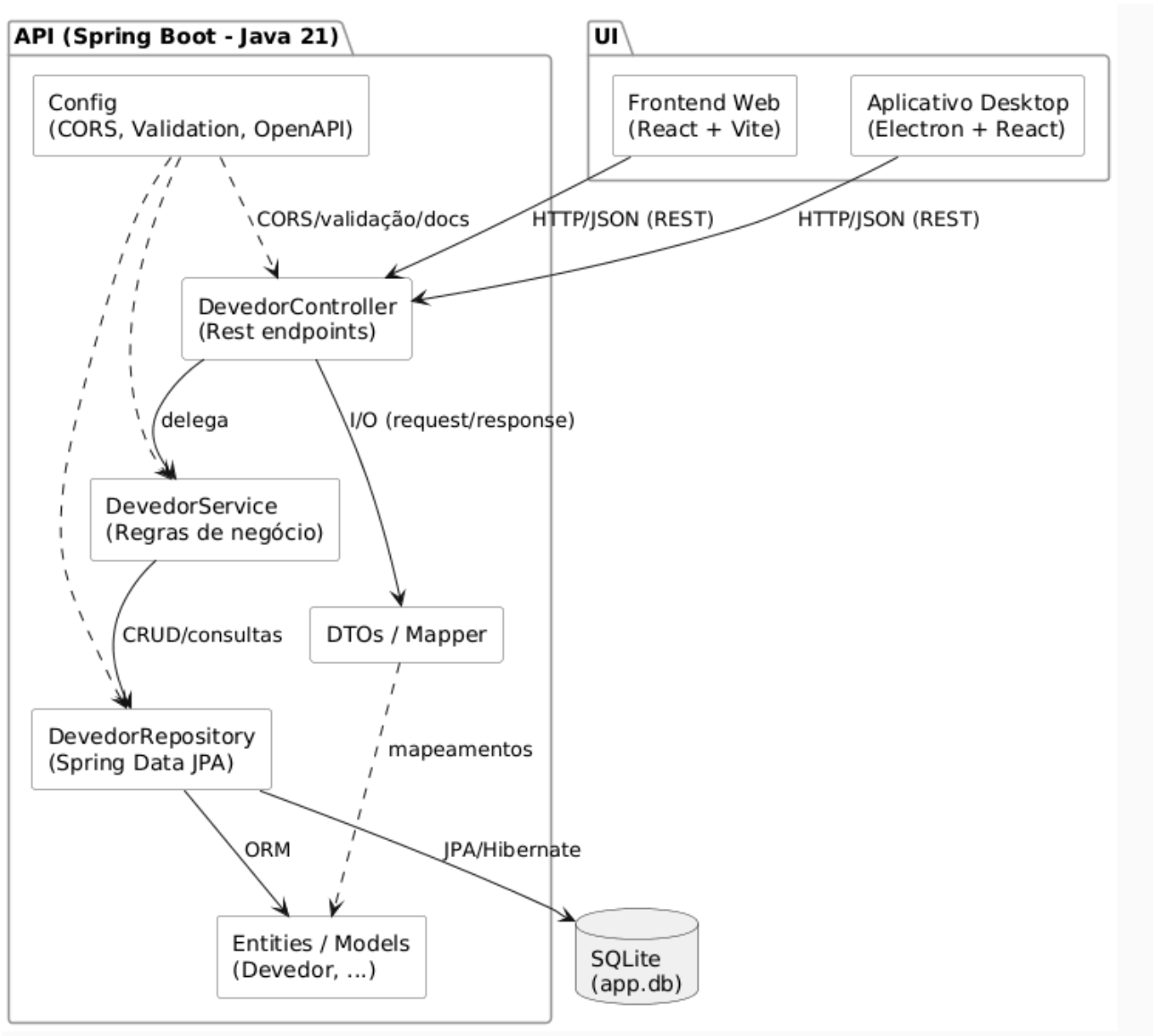


Figura 15 — Diagrama de Componentes

O diagrama de implantação descreve a distribuição física (ou virtual) dos artefatos. No cliente, o Navegador acessa o Frontend Web estático (que pode estar hospedado em um serviço de *static hosting*), enquanto o Aplicativo Desktop (Electron) roda localmente no Computador do Usuário. Ambos se conectam à API Night Plants por HTTPS.

No lado do servidor, um Servidor de Aplicação executa o artefato nightplants.jar (Spring Boot/Java 21). O serviço expõe a porta 8080/8443 e lê/escreve dados em um volume persistente onde reside o arquivo SQLite (app.db). Logs de aplicação são gravados no disco do servidor. Em produção, recomenda-se adicionar um proxy reverso (por exemplo, Nginx ou IIS) para terminação TLS, *rate limiting* e *logging* centralizado; contudo, a topologia mínima funciona com uma única instância da API e armazenamento local em SQLite, atendendo ao escopo do sistema.

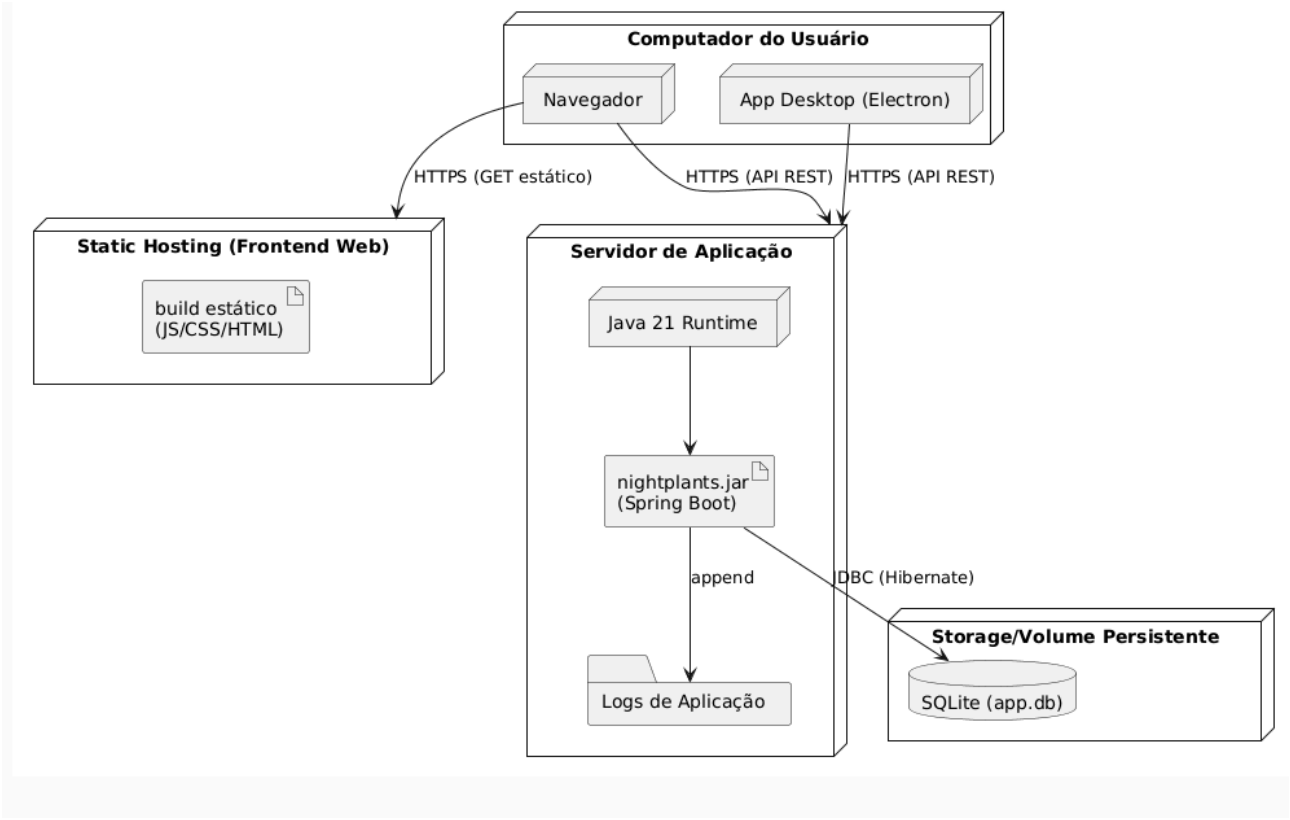


Figura 16 — Diagrama de Implantação

4. Projeto de Interface com Usuário

Esta seção tem como objetivo mostrar e descrever as interfaces de interação com o usuário das quais a aplicação é composta. Para isso foi realizado um *mockup* de baixa fidelidade usando a ferramenta paint.

4.1 Esboço das Interfaces Comuns a Todos os Atores

A tela Dashboard **a**presenta um resumo geral do sistema, exibindo informações como o total de clientes cadastrados, o número de inadimplentes e o valor total em aberto. Além disso, disponibiliza atalhos para o Cadastro de Cliente e para a área de Relatórios. Tanto a Cláudia quanto o José Carlos terão acesso a esta interface, permitindo que acompanhem rapidamente a situação da inadimplência e naveguem para as demais funcionalidades do sistema.

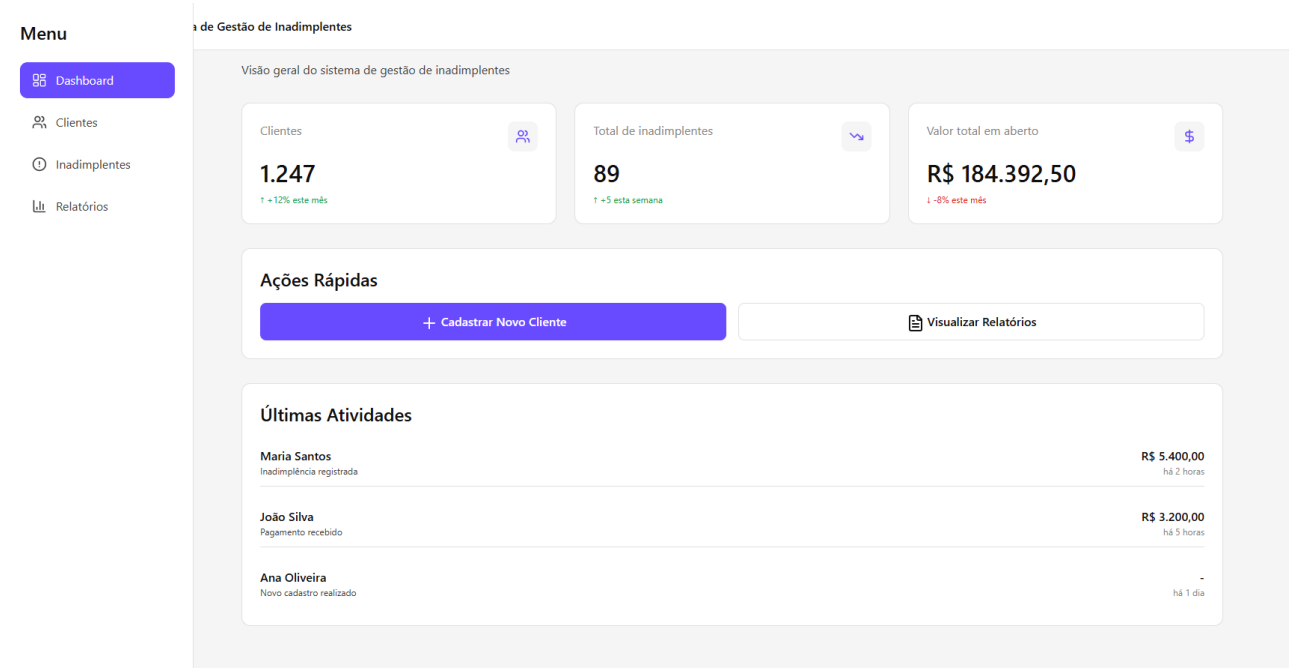


Figura 17. Tela de Dashboard

4.2 Esboço das Interfaces Usada por Ambos

A tela Cadastro de Cliente permite o registro de novos clientes no sistema, contendo campos para inserir informações essenciais como nome, CPF/CNPJ, endereço, telefone e e-mail. Após o preenchimento, é possível salvar os dados ou cancelar a operação. Tanto a Cláudia quanto o José Carlos terão acesso a esta interface, garantindo que novos clientes sejam adicionados de forma organizada e padronizada.

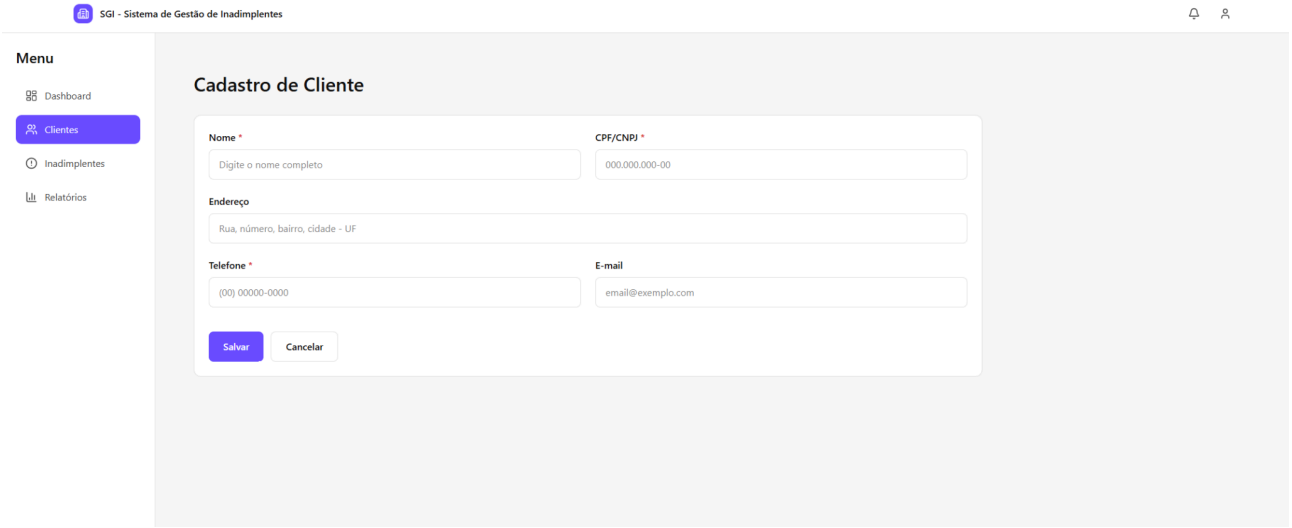


Figura 18. Tela de Cadastro de Cliente

4.3 Esboço das Interfaces Usadas por ambos

A tela Listagem de Clientes apresenta uma visão organizada de todos os clientes cadastrados no sistema, exibindo informações como nome, CPF/CNPJ e situação (adimplente ou inadimplente), e o telefone pra contato.

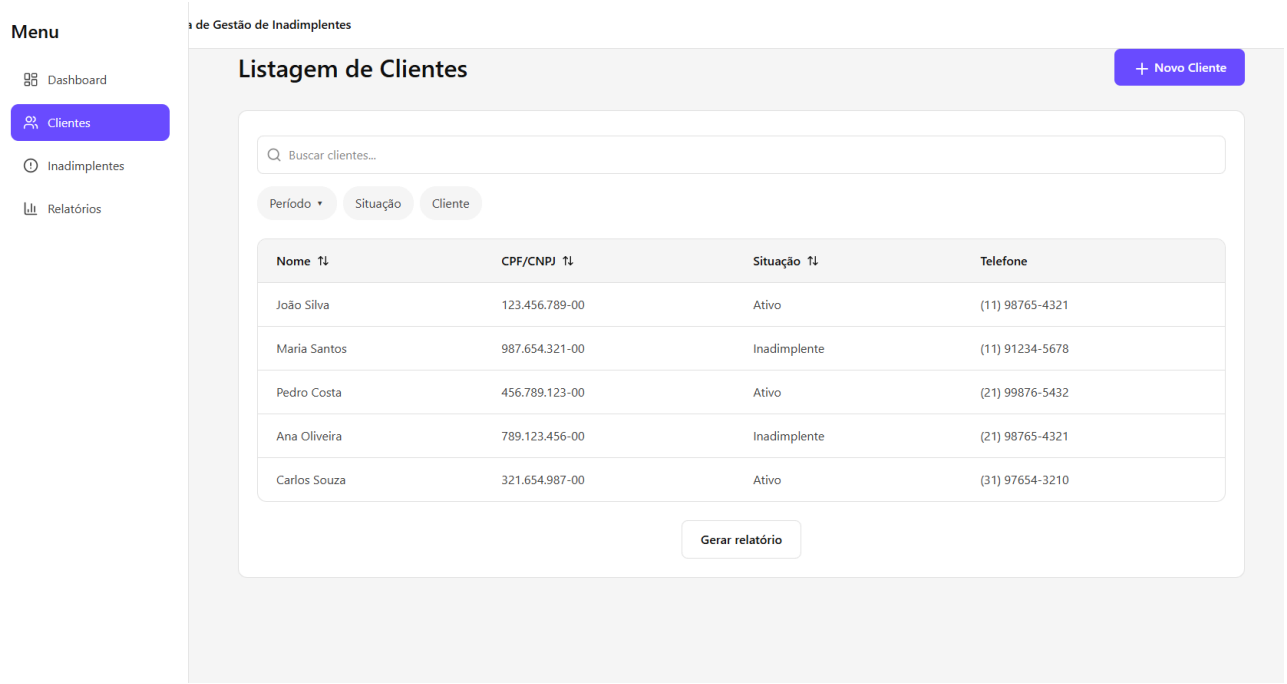


Figura 19. Tela de Listagem de Clientes

4.4 Esboço das Interfaces Usadas por ambos

A tela Lista de Inadimplentes exibe os clientes que possuem pendências financeiras no sistema. São apresentadas informações como o nome do cliente, o valor da dívida e a data de vencimento, permitindo o acompanhamento da situação de cada inadimplência. Tanto a Cláudia quanto o José Carlos terão acesso a esta interface, de forma a facilitar o controle e a tomada de medidas para regularização dos débitos.

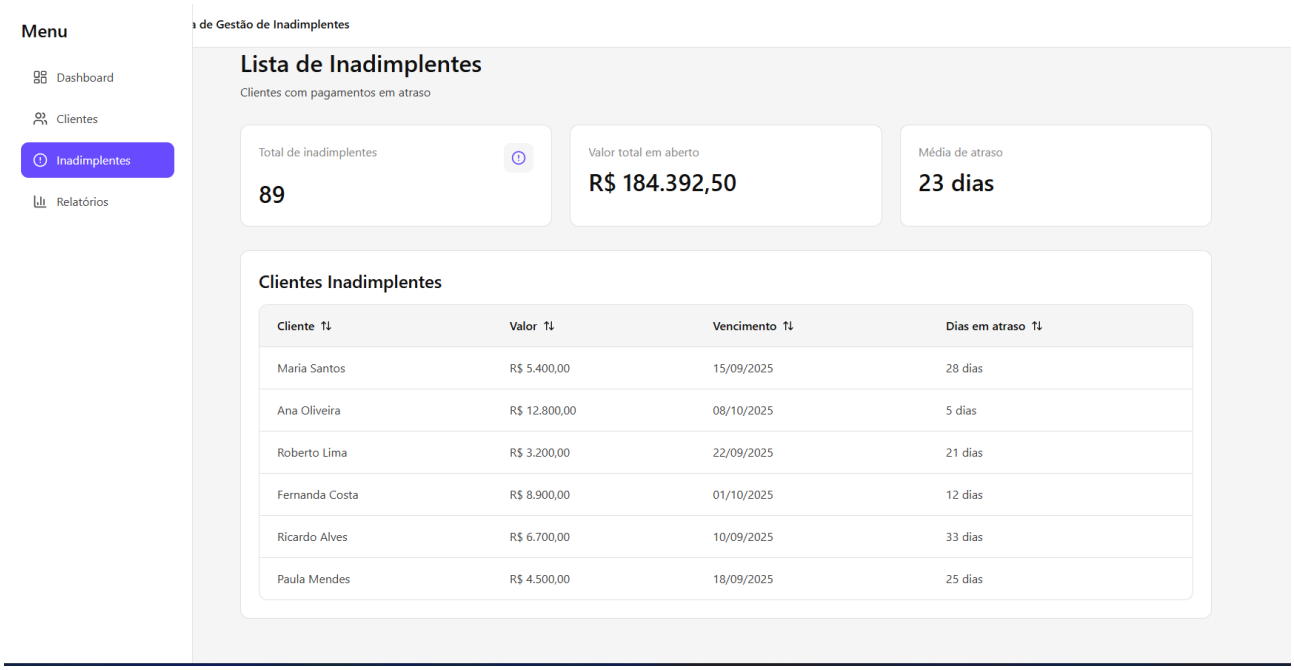


Figura 20. Tela de Lista de Inadimplentes

4.5 Esboço das Interfaces Usadas por ambos

A tela Relatórios possibilita a geração de relatórios personalizados a partir de filtros como período, cliente e status das dívidas. Além da listagem, a interface apresenta informações em formato gráfico, facilitando a análise visual dos dados. Tanto a Cláudia quanto o José Carlos terão acesso a essa funcionalidade, permitindo uma visão estratégica sobre a inadimplência e apoiando o processo de tomada de decisão.

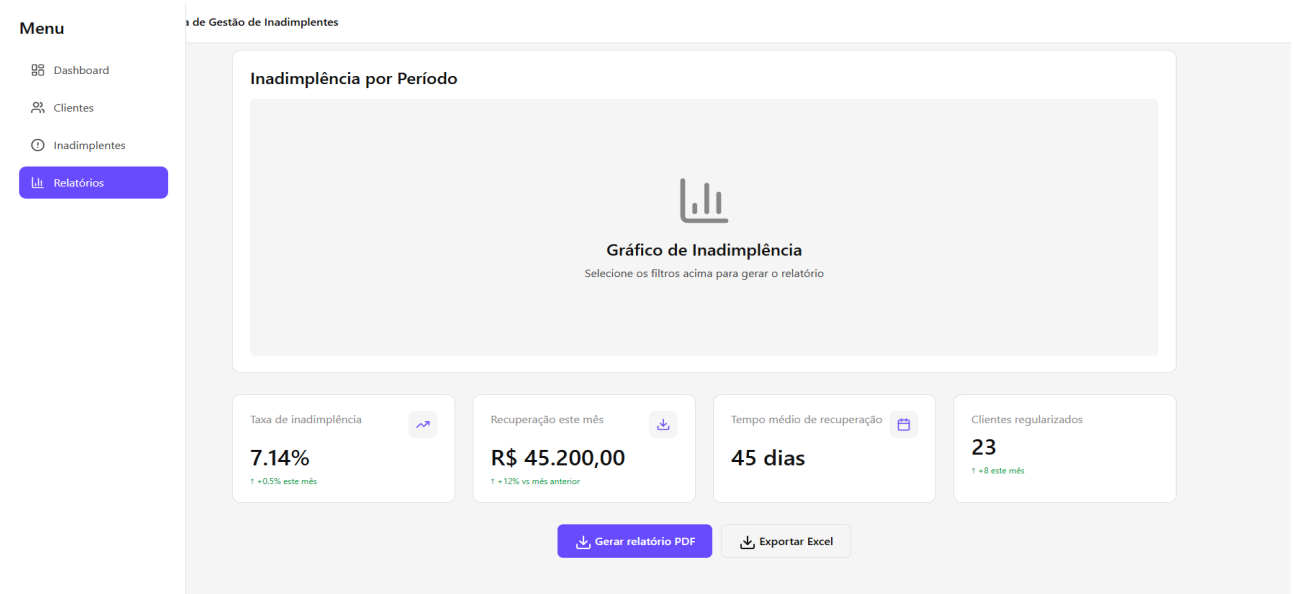


Figura 20 Tela de Relatórios

4.6 Esboço das Interfaces Usadas por ambos

A tela Registro de Inadimplência permite associar um cliente a uma dívida específica, registrando informações como o valor devido, a data de vencimento e a descrição da pendência. Essa funcionalidade garante que os débitos sejam controlados de forma detalhada e organizada. Tanto a Cláudia quanto o José Carlos terão acesso a esta interface, assegurando o correto acompanhamento das inadimplências no sistema.

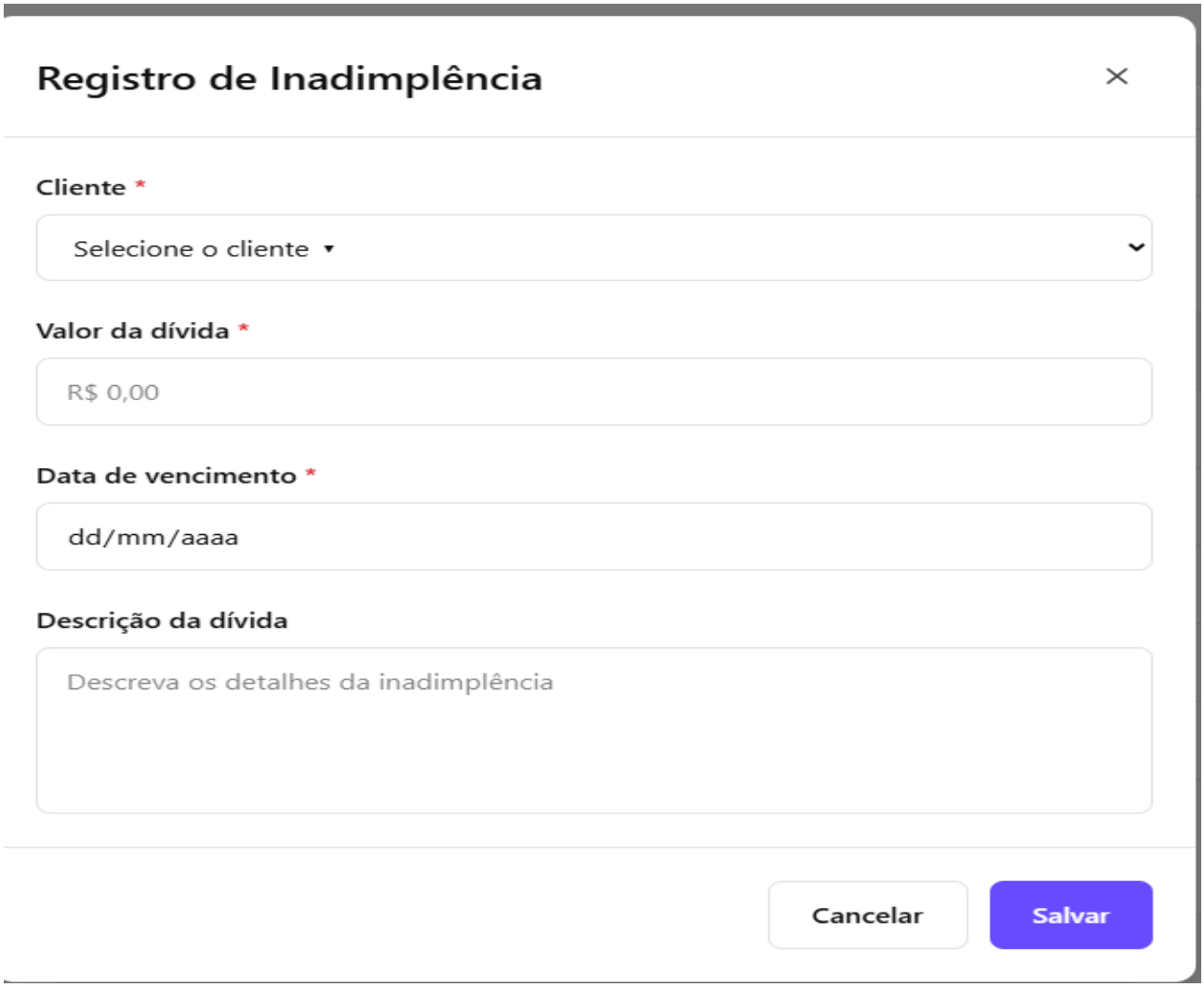


Figura 21. Tela de Registro de Inadimplentes..

5. Glossário e Modelos de Dados

Esta seção descreve o glossário do sistema e o modelo de dados do SGI – Sistema de Gerenciamento de Inadimplentes. Primeiro, apresentam-se tabelas com os atributos de entrada e

saída usados nas telas e relatórios, padronizando nome, formato e significado. Em seguida, apresenta-se o modelo relacional (DER e DDL essencial) e as estratégias de mapeamento entre objetos da aplicação (camada de domínio) e tabelas do banco de dados.

ACESSO

Atributo	Formato	Descriçao
Login	String	Telefone usado no login.
senha	String	Hash/salt da senha (se aplicável).
perfil	enum	Perfil de acesso
tokenacesso	String	Token
statusUsuario	Enum	Situaçao

Tabela 1 acesso.

TABELA CLIENTE

Atributo	Formato	Descriçao
clienteId	UUID	Identificador Cliente
nome	String	Nome completo.
cpfcnpj	String	Documento do cliente
email	String	email
telefone	String	Telefone para contato
endereço	String	Endereço completo
statuscliente	Enum {Ativo, Inativo}	Situação do cadastro
saldo	INTEGER	Saldo
atualizadoem	Timestamp	Ultima atualizaçao

Tabela 2 Tabela Cliente..

TABELA RELATÓRIO E INDICADORES

Atributo	Formato	Descriçao

rankingMaioresDevedores	Lista {clienteId, nome, saldoCentavos}	Ranking ordenado por saldo.
resumoPeriodo	{ini,fim, totalPago, totalAberto}	Indicadores por intervalo.
exportArquivo	PDF/Excel	Resultado exportado sob demanda.

Tabela 3 relatorio e indicadores.

TABELA PAGAMENTO

Atributo	Formato	Descriçao
pagamentoId	UUID/API • INTEGER/BD	Identificador do pagamento.
clienteId	FK → Cliente	Cliente vinculado.
valorPago	INTEGER	Valor pago
dataPagamento	Date	Data do pagamento.
comprovante	Blob/URL	Comprovante (opcional).
criadoEm	Timestamp	Registro do lançamento.

Tabela 4 Pagamentos

TABELA INADIMPLÊNCIA

Atributo	Formato	Descriçao
inadimplenciaId	UUID/API • INTEGER/BD	Identificador da dívida.
clienteId	FK → Cliente	Cliente associado.
valorDevidoCentavos	INTEGER	Valor da dívida (centavos).
vencimento	Date	Data de vencimento.
descricao	String	Observações (opcional).

statusInadimplencia	Enum	Estado da dívida.
criadoEm	Timestamp	Momento do registro.

Tabela 5 Tabela Inadimplencia..

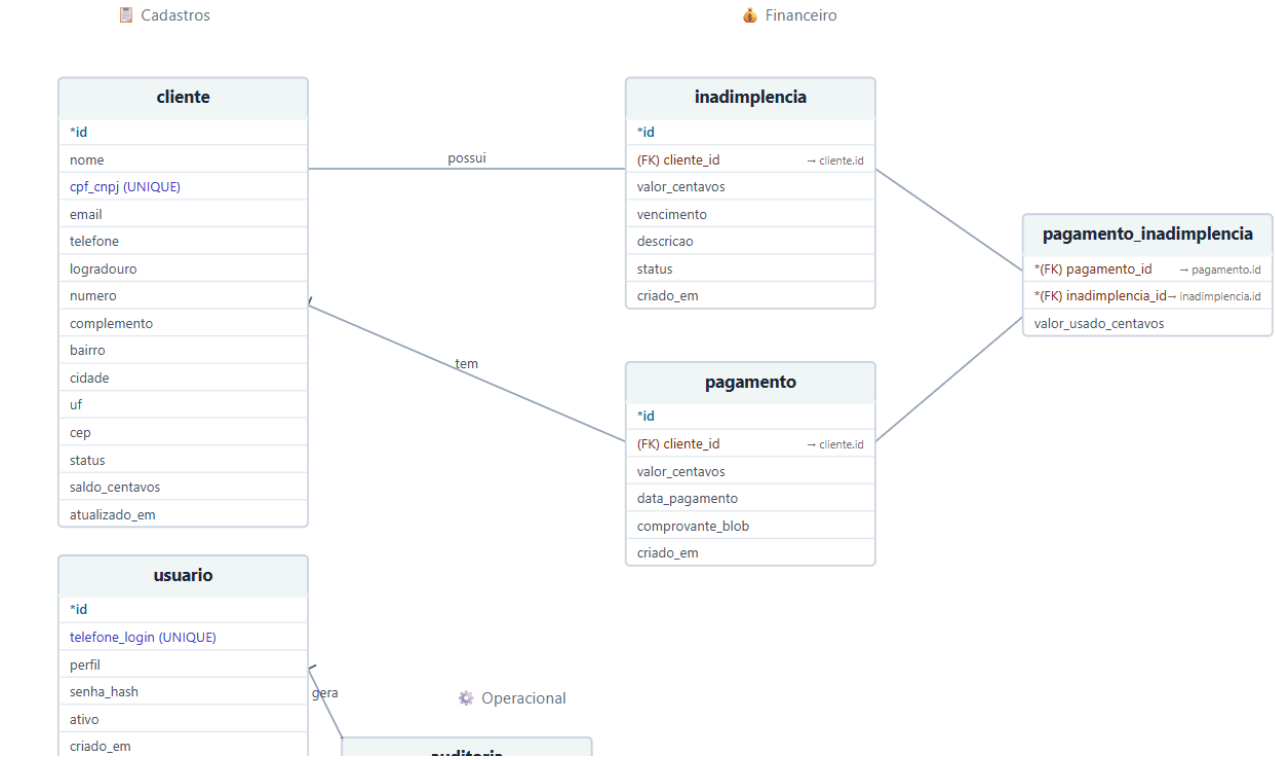


Figura 21. Diagrama de entidade e relacionamnto

6. Casos de Teste

Uma descrição de casos de teste para validação do sistema.

7. Cronograma e Processo de Implementação

Uma descrição do cronograma para implementação do sistema e do processo que será seguido durante a implementação.