UEPB
Universidade Estadual da Paraíba
Centro de Ciências e Tecnologia Prof. Thiago Soares Marques

Trabalho Final: Sistema de Gestão Financeira

Programação Web

---

1 Introdução e Motivação

O desenvolvimento de aplicações web modernas exige que o desenvolvedor saiba integrar diversas camadas: uma interface reativa, uma API segura e um banco de dados persistente. O objetivo deste projeto é construir um **Sistema de Gestão Financeira**, que simula um cenário real de mercado.

Motivação: Gerenciar finanças pessoais não é apenas subtrair despesas de rendas. Envolve lidar com a temporalidade (parcelamentos), diferentes fontes de custeio (cartões vs. dinheiro) e planejamento a longo prazo (reservas e relatórios anuais). Ao concluir este projeto, você terá em seu portfólio uma aplicação robusta que demonstra domínio sobre o fluxo completo de dados em uma arquitetura Next.js.

2 Formação dos Grupos

O trabalho deve ser realizado obrigatoriamente em **grupos de 3 (três) alunos**. Não serão aceitos trabalhos individuais ou com número diferente de componentes sem autorização prévia do professor.

3 Requisitos Funcionais (Interface e Regras de Negócio)

O sistema deve implementar obrigatoriamente as seguintes interfaces e funcionalidades:

3.1 Módulo 1: Acesso e Administração

- **Tela de Login:** Interface para entrada de Usuário/E-mail e Senha, com autenticação processada no backend e proteção via *JSON Web Token* (JWT).
- **Painel de Administração:** Área restrita para gestão do sistema, permitindo a adição de novos membros e a listagem de usuários já cadastrados no banco de dados.

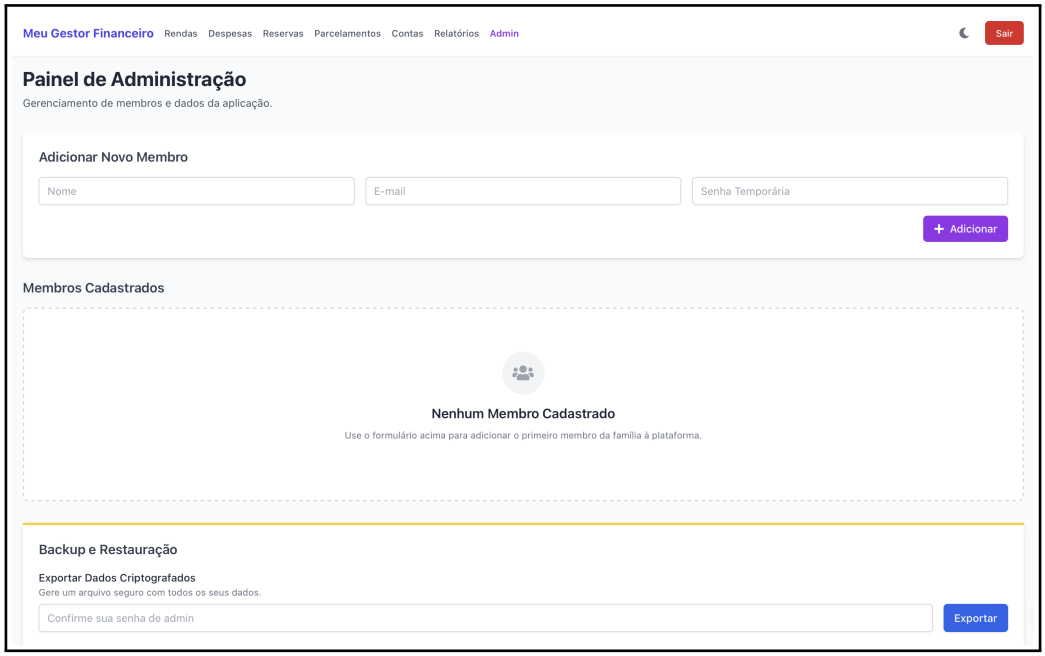

Figura 1: Painel Administrativo

3.2 Módulo 2: Dashboard Principal (Visão Mensal)

O ponto central da aplicação, com lógica voltada para o mês de referência selecionado.

- **Navegação Temporal:** Controles (anterior, atual, próximo) para alternar o escopo de visualização dos dados.
- **Cards de Resumo:** Exibição em destaque do Total de Renda (Entradas), Total de Despesas (Saídas) e o Saldo Líquido resultante.
- **Visualização de Dados:**
  - Gráfico de Pizza: Distribuição de despesas por Categoria.
  - Gráfico de Barras: Comparativo direto entre Rendas vs. Despesas.
  - Ranking: Listagem automática das 5 maiores despesas registradas no mês.

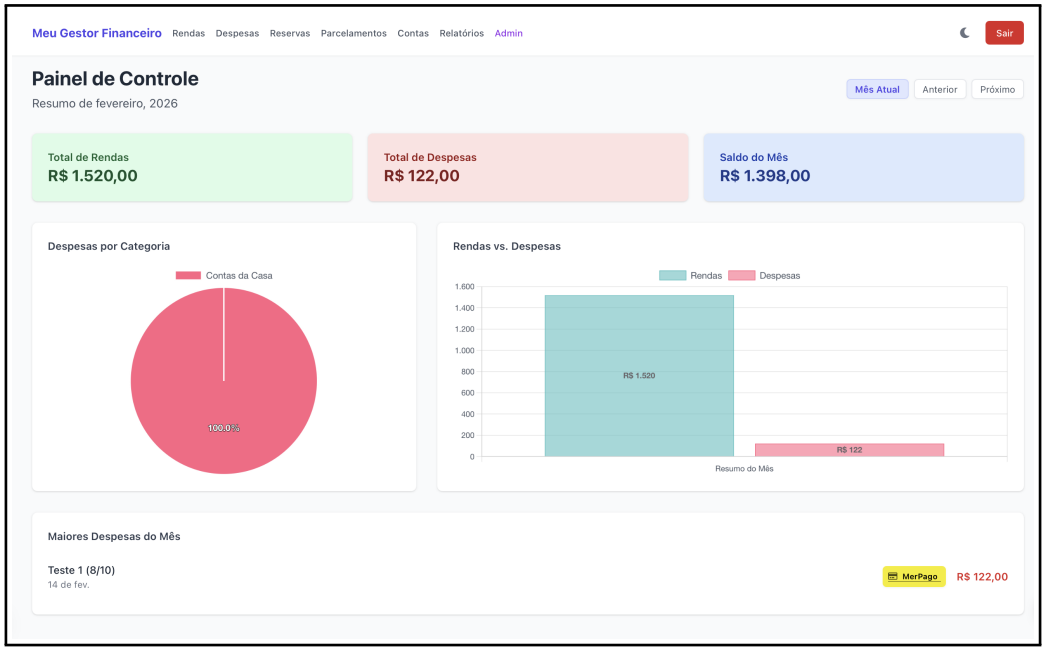

Figura 2: Tela Principal

3.3 Módulo 3: Gestão de Fluxo de Caixa (Rendas e Despesas)

- **Gerenciar Rendas:** CRUD completo com campos de descrição, valor, tipo e data. A listagem deve ser filtrada dinamicamente pelo mês selecionado.
- **Gerenciar Despesas:**
  - **Campos:** Descrição, valor, data, categoria, número de parcelas e vínculo (Conta corrente ou Cartão de Crédito).
  - **Regra de Parcelamento:** Para compras parceladas ($n > 1$), o sistema deve realizar a replicação automática da despesa para os meses subsequentes no banco de dados.
  - **Exportação:** Funcionalidade de download da lista mensal em formato CSV.

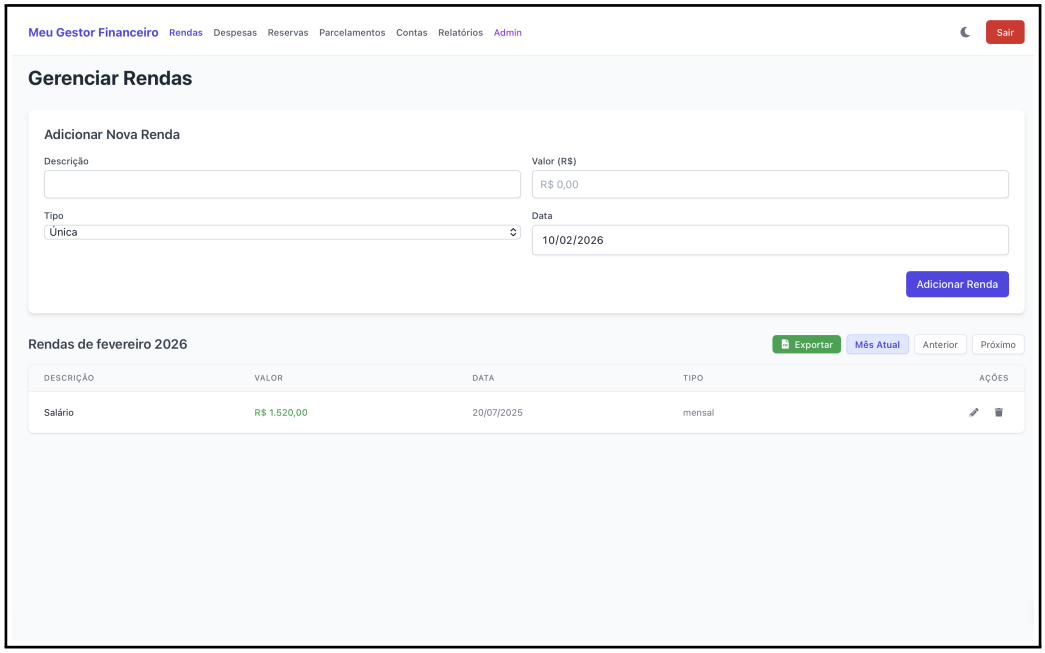

Figura 3: Rendas

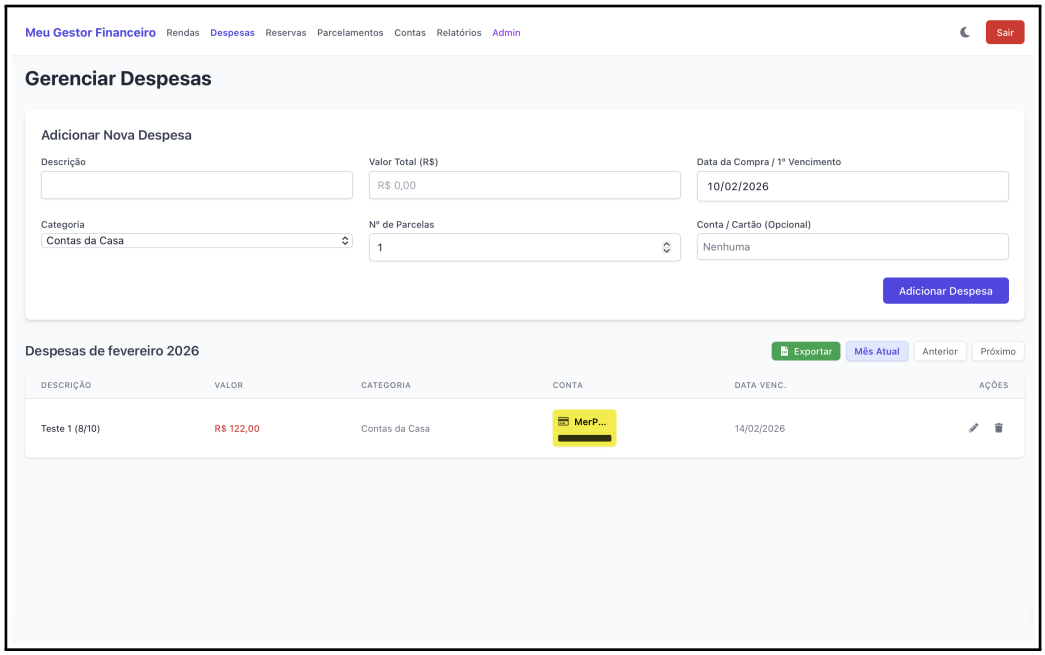

Figura 4: Despesas

3.4 Módulo 4: Reservas e Investimentos

- **Minhas Reservas:** Módulo focado na evolução do patrimônio guardado.

- **Visualização:** Gráfico de linha demonstrando o crescimento ou oscilação das reservas ao longo do tempo.
- **CRUD de Aportes:** Registro de novos valores, datas e a origem/fonte do recurso.

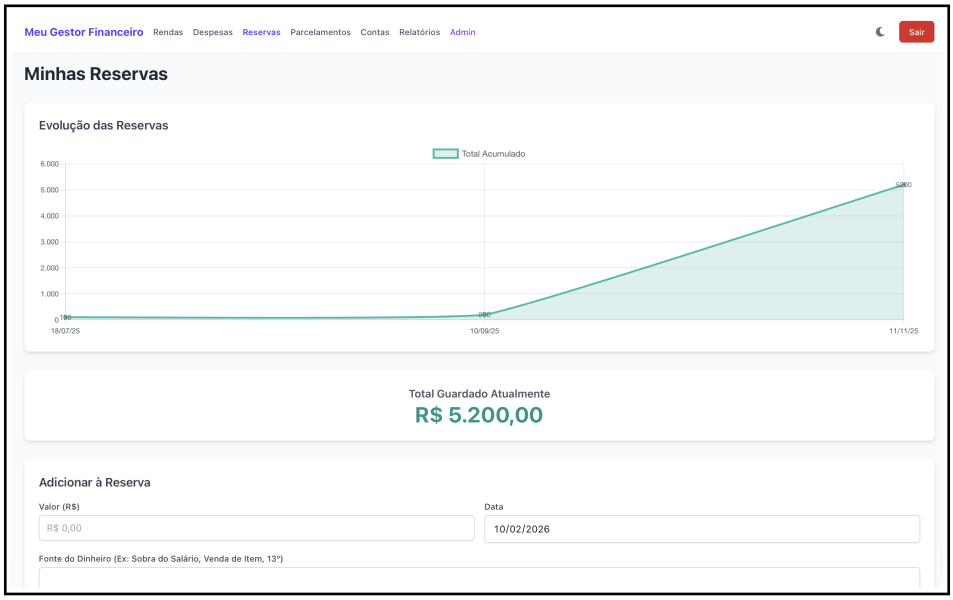

Figura 5: Tela de Reservas (Parte 1)

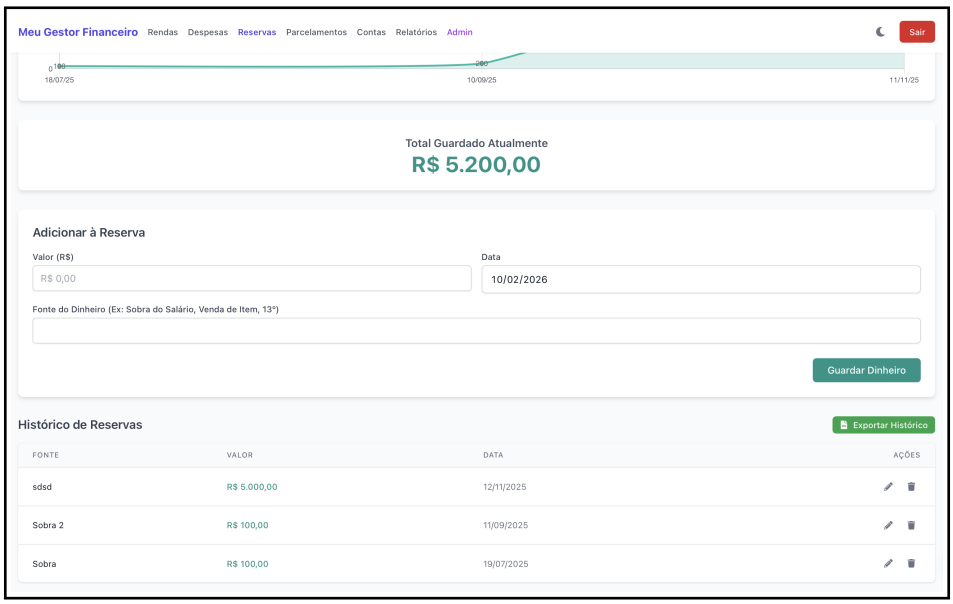

Figura 6: Tela de Reservas (Parte 2)

3.5 Módulo 5: Visão de Futuro e Cartões

- **Parcelamentos e Dívidas:**
  - Card consolidado com o somatório de todas as dívidas futuras.
  - Cronograma de pagamentos detalhando o valor total devido em cada mês futuro.
  - **Visão de Fatura:** Detalhamento técnico por cartão, apresentando o total da fatura e a lista de compras individuais vinculadas àquele período.
- **Gerenciar Cartões:** CRUD para cadastro de cartões, coletando Nome, Tipo, Titular e os últimos 4 dígitos para identificação.

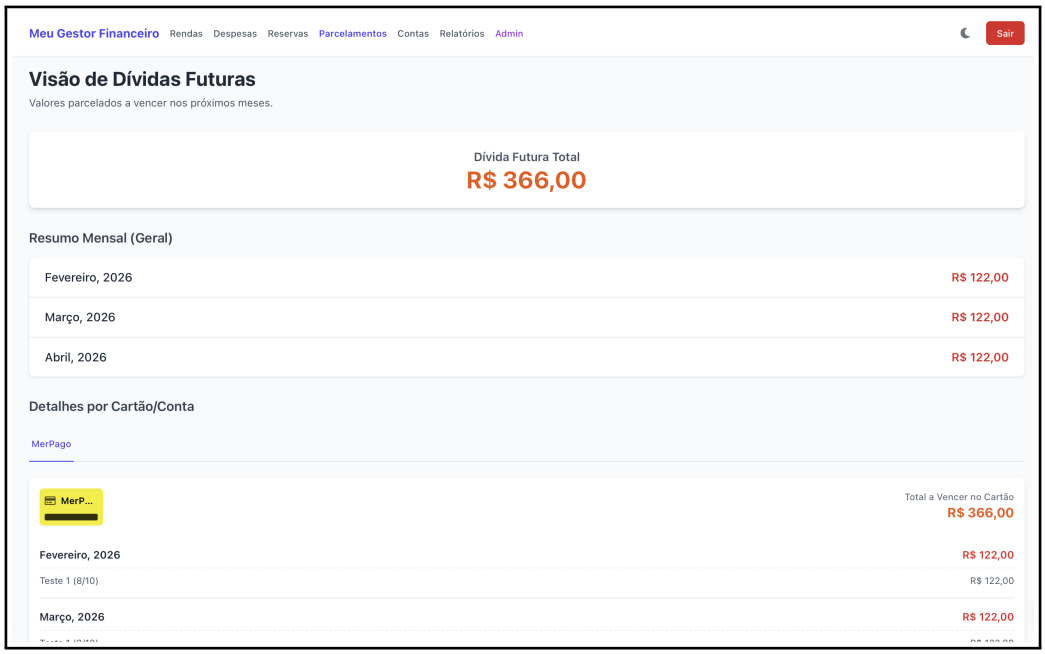

Figura 7: Tela de Parcelamentos

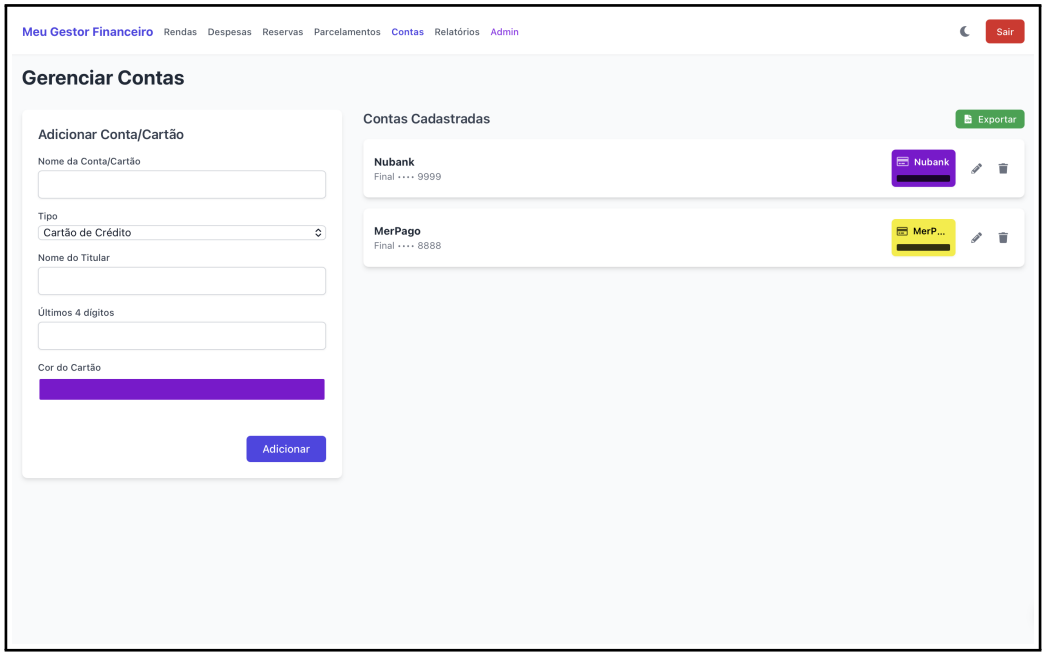

Figura 8: Gerenciamento de Cartões

3.6 Módulo 6: Relatórios Estratégicos

- **Relatório Financeiro Anual:**
  - Filtro seletor de Ano de Referência.
  - Resumo Anual: Balanço consolidado de rendas, despesas e saldo final.
  - Gráfico de Barras: Evolução mensal (Janeiro a Dezembro).
  - Detalhamento: Lista de rendas do ano e agrupamento de despesas por conta/cartão.
  - Exportação: Função para gerar o relatório final consolidado.

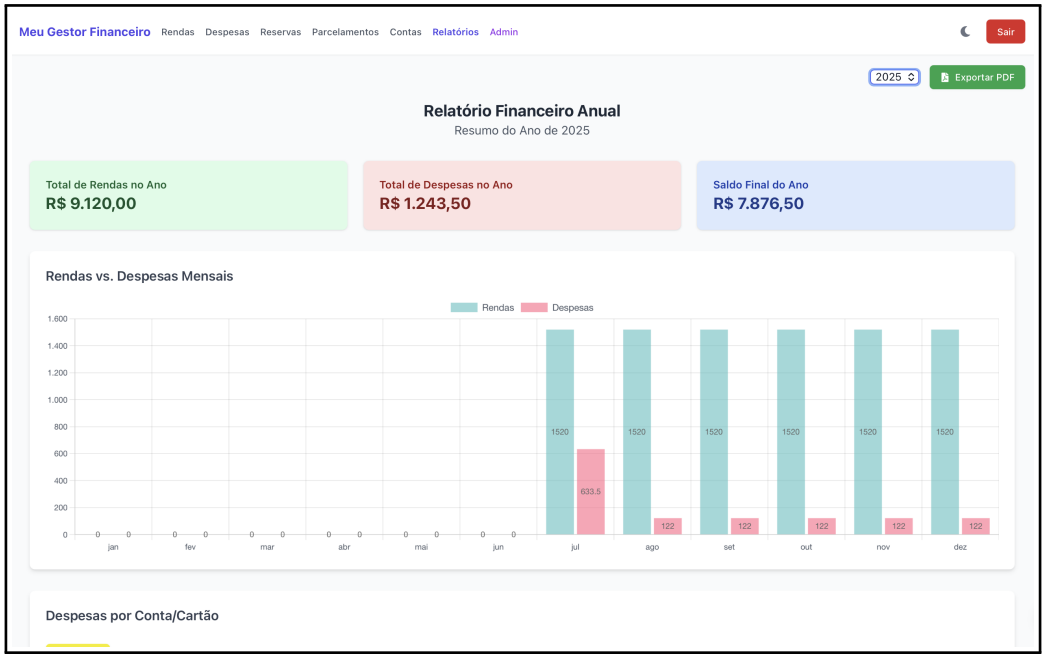

Figura 9: Relatórios (Parte 1)

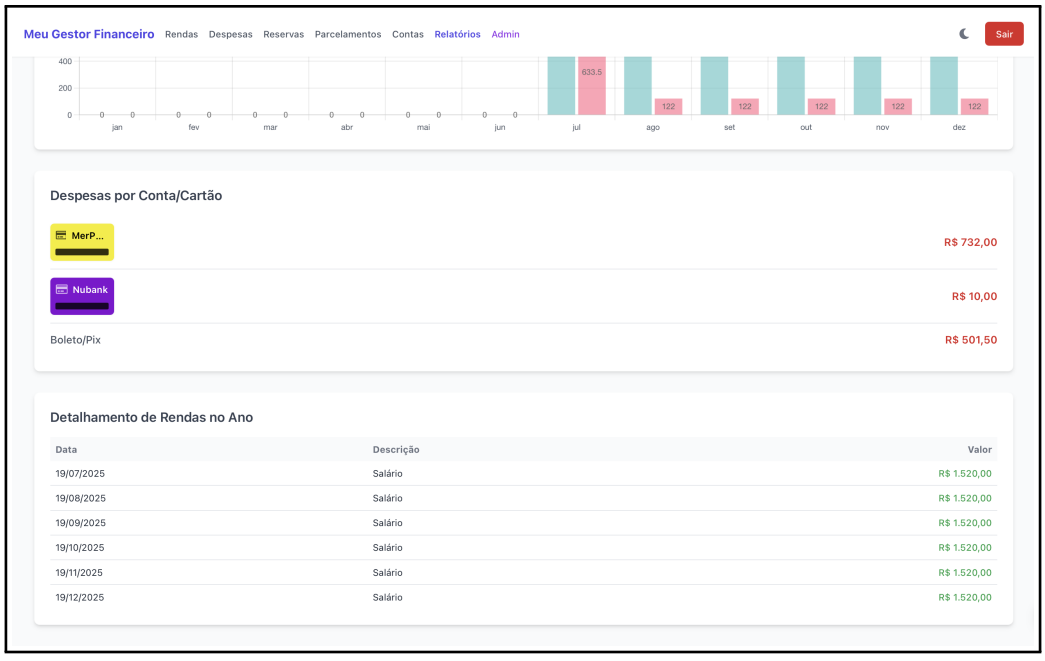

Figura 10: Relatórios (Parte 2)

4 Requisitos Técnicos

4.1 Arquitetura e Backend

- **Tecnologias:** O projeto deve ser desenvolvido utilizando as tecnologias estudadas durante a disciplina, isto é, React/Node.js com o framework Next.js.
- **Padrão MVC:** O código deve ser organizado utilizando o padrão de *Models*, *Controllers* e *Routes*.
- **Persistência:** Implementação utilizando MongoDB com Mongoose. A modelagem deve permitir que despesas parceladas sejam consultadas individualmente por mês.
- **Segurança:** Hashing de senhas via `bcryptjs` e proteção de endpoints através de middlewares que validem o token `JWT`.

4.2 Front-End e UX

- **Hooks:** Uso obrigatório de `useState` para controle de estados locais/fluxos e `useEffect` para chamadas assíncronas à API.
- **Gráficos:** Implementação visual através das bibliotecas `Chart.js` ou `Recharts`.

5 Dicas de Implementação

5.1 Lógica de Parcelamento

Ao salvar uma despesa com $n\_parcelas > 1$, o sistema deve gerar n registros no banco de dados.

$$Valor\_Parcela = \frac{Valor\_Total}{n\_parcelas} \tag{1}$$

Utilize um laço de repetição no *backend* para incrementar o mês de cada registro subsequente. Recomenda-se o uso de um `parcelaId` único para agrupar estas transações.

5.2 Navegação Temporal

A API deve ser capaz de filtrar transações com base no mês e ano de referência enviados pelo *frontend*, garantindo que o *dashboard* seja reativo à navegação do usuário.

6 Avaliação e Critérios

O formato de entrega será composto por três partes. **Nota: A ausência da apresentação resultará em nota ZERO.**

1. **Código Fonte (50%):** Organização no padrão MVC, código limpo e funcionamento sem erros críticos, bem como a aplicação correta dos conceitos vistos durante a disciplina.
2. **Relatório Técnico (30%):** Documentação da modelagem de dados, arquitetura da API, decisões de projeto e dificuldades encontradas.
3. **Apresentação (20%):** Demonstração das funcionalidades e arguição técnica sobre o código implementado.

7 Prazos e Entrega

- **Data Limite:** 18/03/2026.
- **Formato:** Link do repositório no GitHub e arquivo PDF do relatório (via SUAP).

8 Política de Plágio e Uso de IA

- **Plágio:** Cópias idênticas entre grupos ou da internet serão zeradas.
- **IA:** O uso de ferramentas como ChatGPT é permitido como auxílio. Entretanto, o aluno deve ser capaz de explicar 100% da lógica apresentada. A incapacidade de explicar o próprio código resultará em penalização severa.