UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

JOÃO VITOR PEDROSA FARIA

DESENVOLVIMENTO DO PROTÓTIPO DE UM APLICATIVO PARA GERENCIAMENTO DE REPÚBLICAS ESTUDANTIS EM OURO PRETO

Ouro Preto, MG
2025

JOÃO VITOR PEDROSA FARIA

DESENVOLVIMENTO DO PROTÓTIPO DE UM APLICATIVO PARA GERENCIAMENTO DE REPÚBLICAS ESTUDANTIS EM OURO PRETO

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Daniel Ludovico Guidoni

Ouro Preto, MG
2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

João Vitor Pedrosa Faria

DESENVOLVIMENTO DO PROTÓTIPO DE UM APLICATIVO PARA GERENCIAMENTO DE REPÚBLICASESTUDANTIS EM OURO PRETO

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 26 de Agosto de 2025.

Membros da banca

Daniel Ludovico Guidoni (Orientador) - Doutor - Universidade Federal de Ouro Preto
Fernanda Sumika Hojo de Souza (Examinador) - Doutora - Universidade Federal de Ouro Preto
Alex Vitorino (Examinador) - Graduado - Universidade Federal de Ouro Preto

Daniel Ludovico Guidoni, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 28/08/2025.

Documento assinado eletronicamente por **Daniel Ludovico Guidoni**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 05/09/2025, às 14:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0974072** e o código CRC **D04D6847**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Resumo

Este trabalho propõe o desenvolvimento de um protótipo de aplicativo web para gerenciamento de repúblicas estudantis em Ouro Preto, com foco em organização colaborativa, praticidade e transparência. A solução busca atender às demandas específicas dessas moradias, oferecendo funcionalidades como controle de tarefas, divisão de despesas e envio automático de notificações. A metodologia adotada envolve levantamento de requisitos com usuários reais, modelagem UML e uso de abordagens ágeis para garantir flexibilidade no desenvolvimento. A implementação utiliza Python e o framework Django, priorizando segurança e escalabilidade. O sistema visa reduzir conflitos, equilibrar responsabilidades e melhorar a comunicação entre moradores. Espera-se que a aplicação contribua de forma prática para a gestão cotidiana das repúblicas e sirva como referência para projetos similares em contextos de moradia compartilhada.

Palavras-chave: Repúblicas estudantis. Aplicativo web. Gestão de tarefas. Controle financeiro. Engenharia de software.

Abstract

This project presents the development of a web application to support the management of student housing communities, known as "repúblicas", in Ouro Preto. Focused on practicality and collaborative organization, the system provides features such as task scheduling, shared expense tracking, and automatic notifications. The methodology includes user interviews for requirements gathering, UML modeling, and the use of agile approaches to ensure adaptability during development. The application is being built with Python and the Django framework, aiming for a secure, scalable, and user-friendly solution. The expected outcomes include improved communication, balanced responsibilities among residents, and increased transparency in daily operations. This system may also serve as a reference for future initiatives targeting shared housing management in academic contexts.

Keywords: Student housing. Web application. Collaborative management. Agile development. Software engineering.

Lista de Ilustrações

Lista de Tabelas

Sumário

1 Introdução

As repúblicas estudantis desempenham um papel central na vida acadêmica, especialmente em cidades como Ouro Preto, onde a Universidade Federal de Ouro Preto (UFOP) reúne milhares de estudantes. Além de funcionarem como moradia, esses espaços promovem o desenvolvimento social, a convivência coletiva e a autogestão entre os moradores. Essa dinâmica de cooperação e tomada de decisão conjunta se alinha com o conceito de governança colaborativa, que, segundo (TORFING, 2020), demonstra como a colaboração entre diferentes partes interessadas pode gerar valor e eficiência. A importância de ferramentas digitais para fortalecer essa colaboração já é evidenciada na literatura, onde a tecnologia é vista como um facilitador para o compartilhamento de informações e a coordenação de esforços.

No entanto, a administração dessas moradias apresenta diversos desafios. A divisão de tarefas, o controle de despesas compartilhadas e a comunicação interna são frequentemente organizados de forma manual ou improvisada, gerando sobrecarga para alguns moradores, conflitos e falhas de organização. Tais dificuldades são agravadas pela ausência de ferramentas tecnológicas específicas para esse contexto. A resolução desses problemas exige uma abordagem que vá além da tecnologia, focando em como ela pode mediar e estruturar a interação humana (ANSELL; GASH, 2018).

Diante dessa realidade, este trabalho propôs o desenvolvimento de um aplicativo web voltado à gestão colaborativa de repúblicas estudantis. A proposta foi criar uma solução integrada que facilite a organização de tarefas, o acompanhamento de despesas e o envio de notificações, promovendo uma convivência mais transparente e eficiente.

O projeto adotou princípios da engenharia de software e segue uma abordagem baseada em metodologias ágeis. A modelagem do sistema foi feita com base em diagramas UML, garantindo clareza no planejamento e suporte às decisões técnicas.

A implementação foi realizada com uso da linguagem Python e do framework Django, priorizando segurança, escalabilidade e usabilidade. O sistema buscou atender inicialmente às repúblicas de Ouro Preto, mas poderá ser adaptado para outros contextos de moradia estudantil coletiva.

Ao unir vivência prática, fundamentação teórica e aplicação de boas práticas de desenvolvimento de software, este trabalho buscou preencher uma lacuna tecnológica ainda pouco explorada e contribuir com soluções inovadoras no campo da gestão colaborativa.

1.1 Justificativa

A gestão de repúblicas estudantis envolve a divisão de responsabilidades entre moradores com rotinas, perfis e prioridades distintas. Sem ferramentas específicas para esse tipo de organização coletiva, é comum que atividades do dia a dia — como o cumprimento de tarefas domésticas, o pagamento de contas e a comunicação interna — sejam realizadas de forma desorganizada ou concentradas em poucas pessoas.

Embora existam aplicativos para controle de finanças, como Splitwise (Splitwise, 2024) ou de tarefas, como Todoist (CONQUER, 2019) e Trello (Atlassian, 2024), essas ferramentas não foram concebidas para o contexto das repúblicas. Elas carecem de funcionalidades integradas que levem em conta a natureza compartilhada, rotativa e autogerida dessas moradias. Além disso, muitas soluções exigem múltiplos aplicativos para tarefas distintas, o que fragmenta ainda mais o processo de gestão e comunicação.

Nesse cenário, surge a necessidade de uma aplicação web que unifique essas funções em um único ambiente, desenhado especificamente para repúblicas estudantis. O sistema proposto busca preencher essa lacuna ao oferecer uma solução centralizada, adaptável e acessível, capaz de promover a colaboração entre moradores e evitar sobrecargas e conflitos recorrentes.

Além de seu impacto prático imediato, o projeto também contribui para o campo da engenharia de software, explorando o desenvolvimento de soluções colaborativas aplicadas a comunidades autogeridas. Espera-se que a ferramenta possa servir de modelo para projetos semelhantes voltados à moradia compartilhada em outros contextos acadêmicos ou sociais.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um aplicativo web colaborativo que auxilie na gestão de repúblicas estudantis, integrando funcionalidades de organização de tarefas, controle financeiro e comunicação interna, com foco em usabilidade, eficiência e adequação ao contexto coletivo das moradias em Ouro Preto.

1.2.2 Objetivos Específicos

- Levantar os requisitos necessários para um sistema de gerenciamento de repúblicas estudantis;
- Modelar o sistema utilizando diagramas UML (casos de uso, classes, sequências), com base nos requisitos levantados;
- Implementar o backend do sistema utilizando a linguagem Python e o framework Django;

- Desenvolver a interface do usuário com tecnologias web (HTML, CSS, JavaScript e Bootstrap);
- Criar e integrar módulos para:
  - Cadastro e gerenciamento de tarefas com prazos e recorrências;
  - Registro e divisão de despesas entre os moradores;
  - Dashboard para visualização dos dados;

1.3 Organização do Trabalho

A estrutura deste trabalho está dividida nos seguintes capítulos:

1.3.1 Estrutura da Monografia

- **Capítulo 1 – Introdução:** apresenta o tema, a contextualização do problema, os objetivos da pesquisa, a justificativa e a organização geral do trabalho.
- **Capítulo 2 – Revisão Bibliográfica:** aborda os fundamentos teóricos relevantes para o desenvolvimento do sistema, incluindo a gestão de repúblicas estudantis, modelagem UML, desenvolvimento web e metodologias ágeis, além da análise de trabalhos relacionados.
- **Capítulo 3 – Desenvolvimento:** descreve o processo de construção do sistema proposto, contemplando o levantamento de requisitos, a modelagem com diagramas UML, as tecnologias utilizadas e as decisões de projeto.
- **Capítulo 4 – Resultados:** apresenta os resultados da implementação do sistema, com prints da interface final e a discussão sobre como as funcionalidades desenvolvidas atendem aos requisitos do projeto.
- **Capítulo 5 – Conclusão e Trabalhos Futuros:** apresenta uma síntese do trabalho realizado, indicando se os objetivos propostos foram atendidos e destacando as contribuições do sistema desenvolvido, bem como sugestões para trabalhos futuros.

2 Revisão Bibliográfica

Este capítulo aborda os referenciais teóricos, conceitos e as ferramentas utilizadas no desenvolvimento do sistema, além dos trabalhos relacionados ao tema.

2.1 Fundamentação Teórica

Nesta seção são apresentados os principais fundamentos teóricos utilizados no desenvolvimento do sistema, organizados por temas centrais para o projeto: gestão de repúblicas estudantis, modelagem UML, desenvolvimento web e metodologias ágeis.

2.1.1 Repúblicas Estudantis e sua Gestão

Ouro Preto, localizada em Minas Gerais, teve relevância histórica no Ciclo do Ouro e na Inconfidência Mineira. Após a mudança da capital para Belo Horizonte em 1897, a cidade experimentou uma diminuição populacional. Nesse contexto, imóveis desocupados foram ocupados por estudantes, originando as repúblicas estudantis, que passaram a preservar o patrimônio histórico e a sustentar uma cultura de autogestão e solidariedade.

Com a criação da UFOP em 1969, as repúblicas se consolidaram como a principal forma de moradia estudantil. Dividem-se em federais, administradas pelos próprios estudantes sem aluguel, e particulares, mantidas por aluguéis. Esses espaços desempenham papel essencial na vida acadêmica e cultural, promovendo autonomia e cooperação entre os estudantes.

As repúblicas também estiveram no centro de movimentos estudantis e eventos históricos, como o período militar e protestos por melhores condições acadêmicas. Hoje, elas são reconhecidas como parte significativa da memória coletiva da cidade e da universidade (MACHADO).

Atualmente, a UFOP possui 59 casas que se localizam no entorno do campus Morro do Cruzeiro e no centro histórico de Ouro Preto. (Universidade Federal de Ouro Preto, 2014)

A gestão das repúblicas é conduzida pelos próprios moradores, seguindo uma hierarquia em que os residentes mais antigos assumem responsabilidades administrativas. As tarefas a serem executadas são distribuídas entre todos os moradores.

2.1.2 Diagramas UML

A Unified Modeling Language (UML) é uma linguagem de modelagem padronizada amplamente utilizada na engenharia de software para especificar, visualizar, desenvolver e documentar sistemas de software complexos. Por meio de uma representação gráfica clara e bem definida, os diagramas UML ajudam a compreender melhor os requisitos, o design e a arquitetura

de sistemas, promovendo comunicação eficiente entre as equipes de desenvolvimento. (BOOCH; RUMBAUGH; JACOBSON, 2005)

Os diagramas UML podem ser divididos em Estruturais e Comportamentais. Se tratando dos Estruturais, temos:

1. Diagrama de Classe: Representa as classes do sistema, seus atributos, métodos e relações como herança, associação e agregação (FOWLER, 2004).
2. Diagrama de Componentes: Modela a organização dos componentes do sistema e suas dependências (Object Management Group (OMG), 2023).
3. Diagrama de Implantação: Mostra como o sistema será implantado em um ambiente físico, incluindo servidores e conexões (Object Management Group (OMG), 2023).

Em relação aos Comportamentais, temos:

1. Diagrama de Casos de Uso: Identifica os atores e os casos de uso de um sistema, descrevendo o que o sistema deve fazer (FOWLER, 2004).
2. Diagrama de Sequência: Mostra a interação entre objetos em uma sequência de eventos cronológicos (BOOCH; RUMBAUGH; JACOBSON, 2005).
3. Diagrama de Atividades: Representa fluxos de trabalho ou processos dentro do sistema, destacando as atividades e decisões (Object Management Group (OMG), 2023).

Os diagramas UML foram aplicados neste trabalho para modelar a estrutura de dados, os fluxos de interação e as funcionalidades do sistema proposto.

2.1.3 Diagrama ERE

A modelagem de dados para o sistema foi conduzida por meio do Diagrama de Entidade-Relacionamento (ER). O ER foi essencial para o planejamento da base de dados, permitindo a expressão precisa das restrições de cardinalidade e das regras de negócio. Com essa abordagem, foi possível definir de forma clara as relações de um para um (1:1) entre User e UserProfile, e de um para muitos (1:N) entre República e Moradores, assegurando a integridade e a consistência dos dados. A utilização do ER serviu como uma ponte entre a fase de conceituação do sistema e a implementação das tabelas e chaves estrangeiras no banco de dados, garantindo que a estrutura da aplicação fosse robusta e alinhada às suas funcionalidades.

2.1.4 Desenvolvimento Web

O desenvolvimento web é o processo de criação, estruturação e manutenção de aplicações acessíveis por meio da internet, um campo que evolui continuamente para atender às demandas

por sistemas mais interativos e robustos. Conforme (FREEMAN; FREEMAN, 2014), o desenvolvimento moderno abrange desde a análise de requisitos até a implantação, com o objetivo de proporcionar uma experiência de uso fluida, segura e eficiente. Para este trabalho, o sistema foi concebido a partir de uma arquitetura em camadas, dividida em duas áreas principais: o front-end e o back-end, que se complementam para entregar a solução proposta.

A camada de front-end é a interface direta com a qual o usuário interage. Para sua construção, foram utilizadas as tecnologias fundamentais da web (HTML, CSS, JavaScript) e o framework Bootstrap. O uso de um framework como o Bootstrap é crucial para otimizar o processo de desenvolvimento, pois ele oferece um conjunto de componentes pré-estilizados e um sistema de grid responsivo. Essa abordagem permitiu focar na usabilidade e na estética do aplicativo, garantindo uma interface consistente em diferentes dispositivos sem a necessidade de construir cada elemento do zero, o que alinha o projeto com o requisito de usabilidade.

Em contrapartida, a camada de back-end é responsável pelo processamento de dados, lógica de negócio e segurança do sistema. Para esta camada, a linguagem Python e o framework Django foram as escolhas tecnológicas. Frameworks de back-end, como o Django, provêm uma estrutura organizada e ferramentas que simplificam tarefas complexas, como a gestão de bancos de dados (ORM), o roteamento de URLs e o sistema de autenticação de usuários. A escolha por Python e Django assegura que o sistema seja escalável e robusto, priorizando a segurança e a integridade das informações dos moradores. A comunicação eficiente entre essas duas camadas, a do front-end e a do back-end, é o que possibilita a integração entre a lógica de aplicação e a experiência do usuário.

2.1.5 Metodologias Ágeis

As metodologias ágeis priorizam a entrega contínua de valor e a adaptação a mudanças ao longo do ciclo de vida do projeto. Fundamentadas no Manifesto Ágil (BECK, 2001), essas abordagens valorizam interações humanas, software funcional, colaboração com o cliente e flexibilidade. A aplicação desses princípios foi crucial para o sucesso deste trabalho, pois permitiu uma resposta eficaz às necessidades dinâmicas do projeto. O Scrum foi a abordagem adotada, utilizada para organizar o trabalho em ciclos curtos e iterativos, conhecidos como sprints, com um backlog de funcionalidades bem definido. Essa metodologia assegurou que as entregas fossem incrementais e que o progresso do projeto pudesse ser medido e acompanhado de forma contínua, permitindo a evolução do sistema de forma coesa e eficiente.

2.2 Trabalhos Relacionados

Nesta seção, são apresentados alguns trabalhos acadêmicos e soluções existentes que tratam de temas relacionados ao desenvolvimento de sistemas web, especialmente na área de gerenciamento de repúblicas, tarefas e finanças compartilhadas. Os estudos apresentados fornecem

base conceitual e prática para o desenvolvimento do sistema proposto, ao mesmo tempo em que revelam lacunas que a nossa solução pretende preencher.

A monografia de (EMILIANO, 2023) propõe uma arquitetura e implementação de um sistema web para o gerenciamento de repúblicas estudantis. O autor apresenta uma análise detalhada sobre as necessidades específicas dos estudantes que vivem em repúblicas, com foco em recursos, controle de acesso e comunicação. Apesar de ter servido como a principal inspiração teórica para o nosso projeto, a solução não aprofunda na integração de funcionalidades como finanças e tarefas em um fluxo unificado e automatizado.

O trabalho de (MIRANDA, 2022) apresenta um sistema web para gestão administrativa de escritórios de arquitetura. Destaca-se pela implementação de um módulo de tarefas com recursos visuais de arrastar-e-soltar, permitindo aos usuários manipular status como "A fazer", "Em andamento"e "Feito". O uso de diagramas de casos de uso e entidade-relacionamento fortalece a arquitetura do sistema. No entanto, sua aplicação é restrita ao ambiente corporativo, e as funcionalidades não são adaptadas para a dinâmica informal e multifuncional de uma república estudantil.

O estudo de (NEVES, 2023) aborda o desenvolvimento de uma plataforma para gestão de condomínios de pequeno porte, com funcionalidades de finanças, controle de inadimplência e comunicação entre moradores e síndicos. Com uso de metodologias ágeis (Scrum) e tecnologias como Java e Spring Boot, o trabalho apresenta uma solução escalável e segura. Apesar da relevância, a natureza do condomínio (com gestão vertical e hierárquica) difere da gestão horizontal e colaborativa de uma república, onde todos os membros têm voz ativa nas decisões.

O aplicativo Todoist (CONQUER, 2019) é amplamente utilizado para organização de tarefas e projetos. Oferece funcionalidades como criação de subtarefas, definição de prioridades, notificações por e-mail e colaboração entre usuários. A interface limpa e o uso de gamificação para estimular produtividade são pontos de destaque que inspiraram funcionalidades de engajamento do sistema proposto, mas seu foco é individual ou em grupos genéricos, sem integração nativa com o gerenciamento financeiro.

O Tarefaça (RODRIGUES, 2021) é um sistema focado na gamificação de tarefas domésticas em moradias compartilhadas. Apresenta ranking entre os usuários, sistema de pontuação e alarmes para avisos de contas e tarefas. No entanto, o trabalho não inclui mecanismos de cálculo automático de débitos, um recurso essencial para a transparência financeira

O Bills Organizer (SMOBILEAPPS, 2013) é um aplicativo mobile para organização de contas e finanças pessoais. Possui cadastro por categorias, definição de datas de vencimento, relatórios e visualizações por tipo de despesa. Apesar da interface limitada, sua funcionalidade de relatórios financeiros reforça a importância de visualização clara das obrigações, o que foi incorporado no sistema com geração de relatórios automáticos.

O Splitwise (Splitwise, 2024) é uma referência importante em controle de despesas

compartilhadas. Permite criar grupos, dividir contas, acompanhar saldos e registrar pagamentos. Porém, operam isoladamente do gerenciamento de tarefas e do fluxo de comunicação, exigindo o uso de múltiplas plataformas.

O Trello (Atlassian, 2024) é uma plataforma visual baseada em quadros e cartões, bastante utilizada com metodologias ágeis. Suas listas e fluxos de tarefas representam bem a dinâmica colaborativa desejada em repúblicas. A lógica de Kanban visual adotada no projeto foi diretamente inspirada nessa ferramenta. Porém, é uma ferramenta genérica que requer customizações manuais para atender às necessidades específicas de uma república.

A seguir, a Tabela 2.1 apresenta um resumo comparativo entre os trabalhos analisados.

Tabela 2.1 – Comparativo entre trabalhos relacionados

Trabalho	**Tecnologias / Métodos**	**Contribuições Relevantes**
(EMILIANO, 2023)	Sistema web para repúblicas, foco em usabilidade	Interface adaptada à gestão estudantil
(MIRANDA, 2022)	Gestão de tarefas com arrastar-e-soltar	Organização visual com modelagem DER
(NEVES, 2023)	Java, Spring Boot, Scrum	Comunicação interna e controle de finanças em condomínios
(CONQUER, 2019)	App de tarefas com gamificação	Produtividade e organização individual
(RODRIGUES, 2021)	Gamificação, rankings, alarmes	Engajamento em tarefas domésticas
(SMOBILEAPPS, 2013)	App de contas com relatórios	Controle individual de despesas
(Splitwise, 2024)	Divisão de despesas em grupo	Gestão colaborativa de finanças
(Atlassian, 2024)	Kanban visual de tarefas	Organização ágil e colaborativa

Os trabalhos apresentados oferecem importantes contribuições, mas não contemplam de forma integrada e adaptada o contexto específico das repúblicas estudantis. O sistema proposto neste trabalho busca preencher essa lacuna ao reunir, em um único ambiente, funcionalidades de organização de tarefas, controle financeiro e notificações automáticas voltadas à moradia estudantil compartilhada.

3 Desenvolvimento

Com base nos conceitos apresentados na revisão bibliográfica, este capítulo descreve a aplicação prática das metodologias, técnicas de modelagem e ferramentas adotadas no desenvolvimento do sistema proposto. O trabalho segue uma abordagem aplicada e exploratória, com o objetivo de desenvolver uma solução prática para um problema real vivenciado em repúblicas estudantis.

3.1 Levantamento de Requisitos

Não se preocupe, é possível reescrever esse trecho para refletir sua experiência pessoal, mantendo a formalidade acadêmica.

Aqui está uma sugestão de como você pode adaptar o texto:

O levantamento de requisitos foi conduzido com base na experiência vivenciada em repúblicas estudantis da cidade de Ouro Preto. Essa abordagem, que se apoia em um conhecimento aprofundado do cotidiano e das dinâmicas internas desses ambientes, permitiu identificar de forma precisa as necessidades e os desafios enfrentados pelos moradores.

Com base nesse entendimento, os requisitos do sistema foram definidos para atender às demandas reais, especialmente no que se refere à divisão de tarefas, controle de despesas e comunicação entre os membros. As percepções obtidas serviram de base para a elaboração dos requisitos funcionais (Tabela 3.1) e não funcionais (Tabela 3.2).

Os requisitos funcionais do sistema foram definidos com base nas necessidades identificadas, e incluem a capacidade de realizar operações de CRUD (do inglês Create, Read, Update e Delete), que significam, respectivamente, a criação, leitura, atualização e exclusão de dados. Essa terminologia é fundamental para descrever as interações básicas do usuário com as informações do sistema.

Tabela 3.1 – Requisitos Funcionais

Requisitos Funcionais		
Código	**Descrição do Requisito**	**Módulo Responsável**
RF01	O sistema permitirá o CRUD das informações dos usuários (moradores).	Usuários
RF02	O sistema permitirá a autenticação dos usuários.	Usuários
RF03	O sistema permitirá o CRUD e atribuição de tarefas.	Tarefas
RF04	Definição de prazos, prioridades e recorrências para tarefas.	Tarefas
RF05	O sistema permitirá o CRUD de despesas.	Despesas
RF06	O sistema calculará e dividirá automaticamente os valores.	Despesas
RF07	Notificações sobre vencimentos de contas e tarefas pendentes.	Notificações

Tabela 3.2 – Requisitos Não Funcionais

Requisitos Não Funcionais		
Código	**Requisito**	**Justificativa**
RNF01	A interface deve ser agradável e de fácil utilização.	Melhoria da experiência do usuário.
RNF02	Os dados devem ser armazenados em servidor externo.	Garantir acesso remoto e segurança.

3.2 Modelagem com UML

A arquitetura do sistema foi concebida como uma aplicação web colaborativa, dividida em camadas de frontend e backend. O sistema foi estruturado pra ser escalável e seguro, utilizando o padrão MVT(Model-View-Template) do framework Django para gerenciar a lógica de negócio e a renderização das interfaces. A modelagem do sistema foi realizada utilizando os diagramas UML descritos na Revisão Bibliográfica. Foram elaborados diagramas de classes, casos de uso e sequência, aplicados para representar os fluxos e estruturas dos módulos de tarefas, despesas e usuários.

3.2.1 Diagrama de Classes

O diagrama de classes apresentado na Figura 3.1 ilustra a estrutura de dados de um sistema de gerenciamento de repúblicas estudantis, com foco nos relacionamentos entre as principais entidades: `Republica`, `User`, `UserProfile`, `Tarefa` e `Despesa`. O diagrama é organizado para refletir a lógica de negócio do sistema. A entidade `Republica` "possui" múltiplos `UserProfile` (moradores), o que é modelado como uma relação de um para muitos (1:N). Da mesma forma,

a entidade `User` “possui” um `UserProfile`, estabelecendo uma relação de um para um (1:1), o que permite estender o modelo de usuário padrão do Django para incluir informações como telefone e tipo de morador.

As entidades `Tarefa` e `Despesa` também se conectam com a `Republica` e o `UserProfile`. Cada `Tarefa` “pertence” a uma `Republica` (1:N) e é “responsável por” um `UserProfile` (N), indicando quem deve executá-la. Similarmente, cada `Despesa` pertence a uma `Republica` (1:N) e é paga por um `UserProfile`. A entidade `Despesa` também tem uma relação de muitos para muitos (N:N) com `UserProfile`, representando a divisão de uma despesa entre vários moradores. Essa estrutura de relacionamentos entre as entidades assegura a consistência dos dados, permitindo a correta atribuição de tarefas e o controle transparente de despesas em cada moradia.

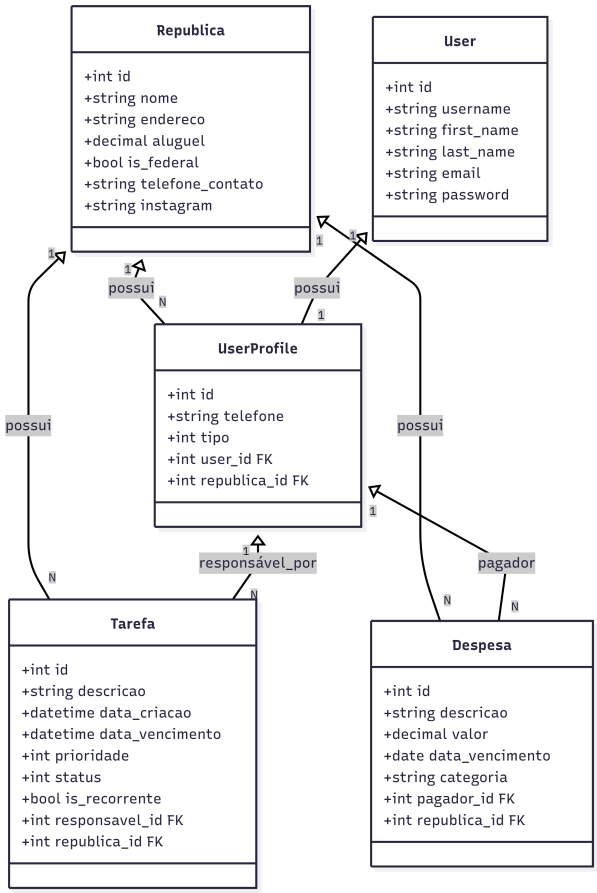

Figura 3.1 – Diagrama de Classes

3.2.2 Diagramas Comportamentais

A Figura 3.2, que representa o diagrama de casos de uso para o módulo de Tarefas, mostra as funcionalidades disponíveis para os diferentes perfis de usuário. O "Morador Comum"pode "Criar Tarefas", "Editar Tarefas"e "Ver Tarefas". Já o "Administrador Sistema/República"possui um conjunto de permissões mais amplo, incluindo a capacidade de "Apagar Tarefas"além das ações já permitidas ao morador comum.

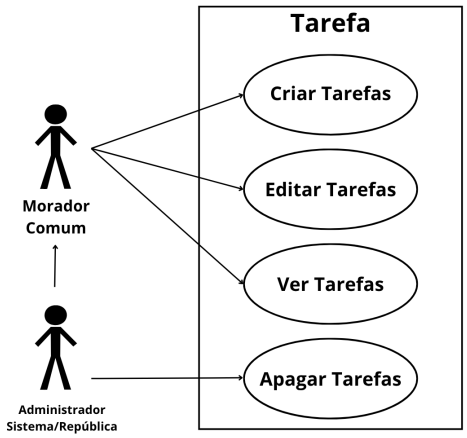

Figura 3.2 – Casos de Uso – Tarefas

Similarmente, a Figura 3.3 exibe o diagrama de casos de uso para o módulo de Despesas. Assim como no módulo de tarefas, o "Morador Comum"pode "Criar Despesas,Editar Despesas"e "Ver Despesas". O "Administrador Sistema/República"também tem a permissão adicional de "Apagar Despesas,"garantindo o controle total sobre os registros financeiros da república.

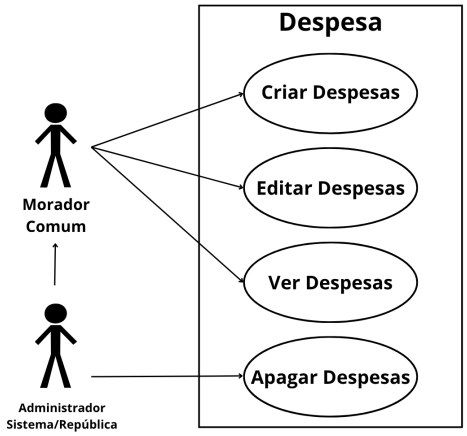

Figura 3.3 – Casos de Uso – Despesas

Por fim, a figura 3.4 ilustra o diagrama de casos de uso para o módulo de Moradores e as interações entre os usuários e o sistema para gerenciar os perfis dos membros da república. O "Morador Comum"tem a permissão de "Ver Moradores", o que garante a transparência ao permitir que todos os membros visualizem os perfis uns dos outros. Já o "Administrador Sistema/República"possui um conjunto de permissões mais abrangente. Além de "Ver Moradores", este perfil pode "Criar Moradores", "Editar Moradores"e "Apagar Moradores", o que lhe confere o controle total sobre o cadastro e a gestão dos usuários da república

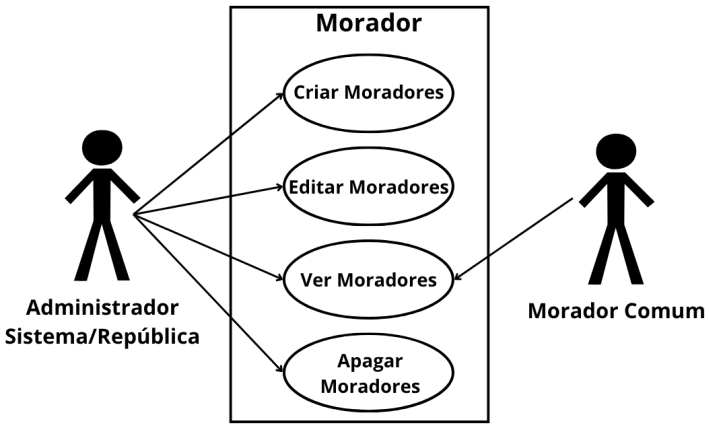

Figura 3.4 – Casos de Uso – Moradores

3.2.3 Diagramas de sequência

O Diagrama de Sequência — Registro de Tarefas (Figura 3.5) descreve o fluxo de interação para a criação de uma nova tarefa. O fluxo começa com o "Morador"selecionando a opção "Registrar Tarefa"na interface do usuário. Em seguida, o morador insere os dados da tarefa e confirma a ação. O "Sistema"recebe esses dados, valida-os e os insere no "Banco de Dados". Uma vez que o banco de dados confirma o registro, o sistema, por sua vez, confirma o registro da tarefa para o morador, finalizando a interação.

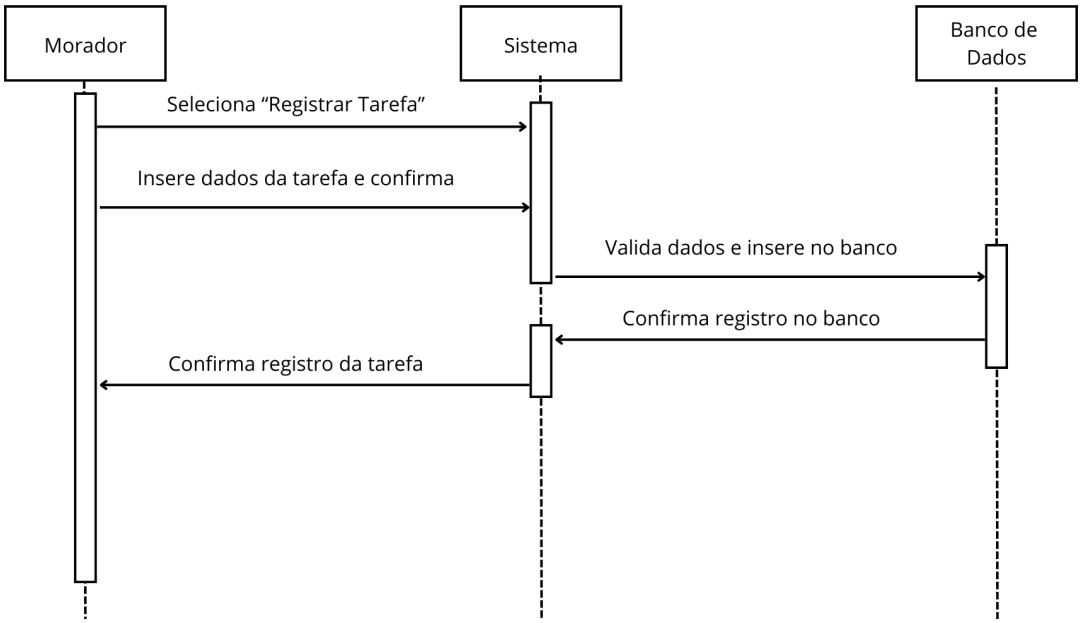

Figura 3.5 – Diagrama de Sequência – Registro de Tarefas

O Diagrama de Sequência — Registro de Despesas (Figura 3.6) detalha o processo de cadastro de uma despesa. O fluxo é similar ao de registro de tarefas, mas com foco em finanças compartilhadas. O "Morador"inicia o processo selecionando a opção "Registrar Despesa", insere os dados necessários (como valor e descrição) e confirma a ação. O "Sistema"recebe esses dados, realiza a validação e armazena as informações no "Banco de Dados". Após o banco de dados confirmar que o registro foi realizado com sucesso, o sistema informa ao morador que o registro da despesa foi concluído, assegurando a transparência e a organização financeira.

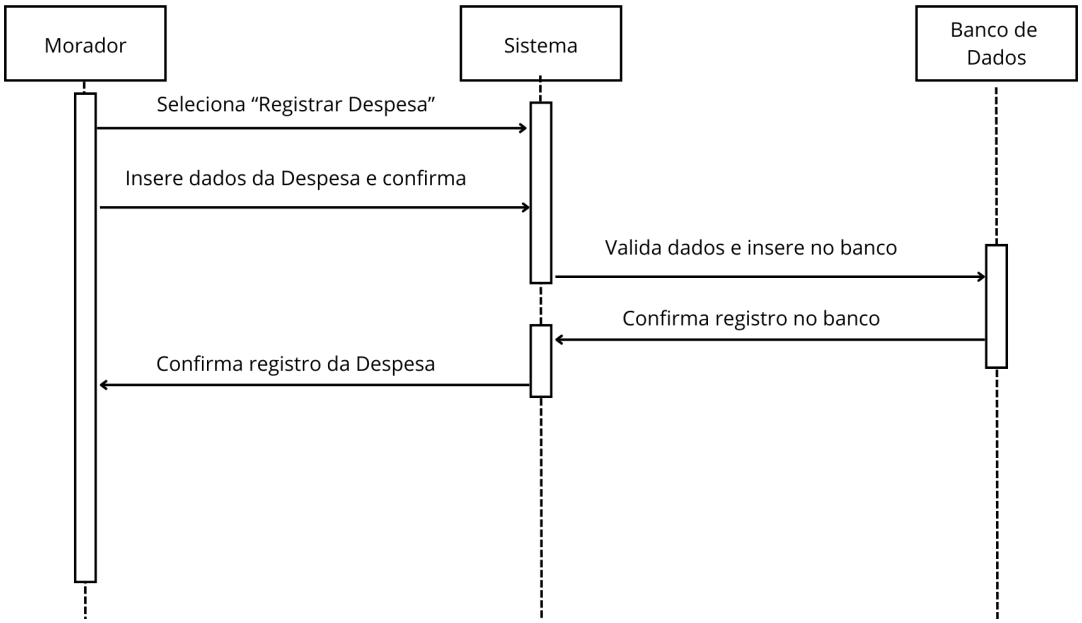

Figura 3.6 – Diagrama de Sequência – Registro de Despesas

Por fim, o Diagrama de Sequência — Registro de Usuários (Figura 3.7) ilustra o fluxo de interações para o cadastro de novos usuários no sistema. O processo começa quando o "Administrador da República"seleciona a opção para registrar um novo usuário. Em seguida, o administrador insere os dados do novo usuário e confirma a ação. O "Sistema"recebe esses dados, realiza a validação necessária e os insere no "Banco de Dados". Após o banco de dados confirmar o registro, o sistema, por sua vez, informa ao administrador da república que o registro do usuário foi concluído com sucesso.

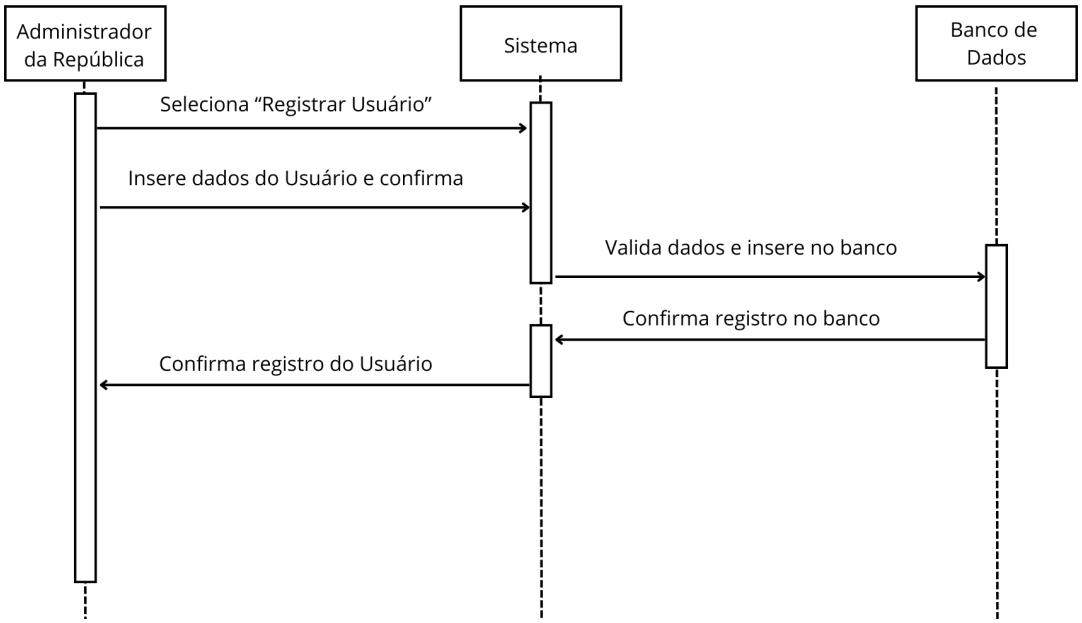

Figura 3.7 – Diagrama de Sequência – Registro de Usuários

3.3 Diagramas ER

A modelagem da arquitetura de dados foi realizada utilizando a notação de Diagramas de Entidade-Relacionamento (ER). Essa abordagem serviu como um guia visual e técnico para a implementação das classes de modelo no framework Django, assegurando que o banco de dados fosse estruturado de maneira lógica e consistente para suportar as funcionalidades do sistema. Os EREs permitiram expressar com precisão as restrições de cardinalidade, como a relação de um para muitos entre a *República* e seus *Moradores*, e a relação de um para um entre o *User* e seu *UserProfile*, garantindo a integridade dos dados desde a sua concepção.

3.3.1 Diagrama de Entidade-Relacionamento (DER) Geral

O Diagrama ER ilustrado na Figura 3.8 apresenta a arquitetura de dados do sistema, detalhando as entidades e seus relacionamentos para gerenciar informações de uma república estudantil. O diagrama é composto pelas entidades `REPUBLICA`, `USER`, `USER_PROFILE`, `TAREFA` e `DESPESA`. A entidade `USER`, que é o modelo padrão do Django, está ligada a `USER_PROFILE` por uma relação de um para um (1:1), permitindo estender os dados do usuário com informações como *telefone*, *tipo* e o ID da república (*republica_id FK*). A entidade `REPUBLICA` é o centro da organização, possuindo uma relação de um para muitos (1:N) com `USER_PROFILE`, o que permite que uma república tenha vários moradores.

O diagrama também detalha a gestão de tarefas e despesas. A entidade `TAREFA` está

relacionada a REPUBLICA por uma chave estrangeira (*republica_id FK*), indicando a qual moradia a tarefa pertence. Ela também possui uma relação de muitos para um com USER_PROFILE, através do campo *responsavel_id FK*, que identifica o morador responsável pela tarefa. De forma semelhante, a entidade DESPESA está ligada a REPUBLICA (*republica_id FK*) e a USER_PROFILE através do campo *pagador_id FK*, indicando quem pagou a despesa. Além disso, a DESPESA tem uma relação de muitos para muitos com USER_PROFILE (*envolvido_em*), representando a divisão de uma despesa entre múltiplos moradores, e inclui campos como *valor_por_pessoa* para gerenciar a transparência financeira.

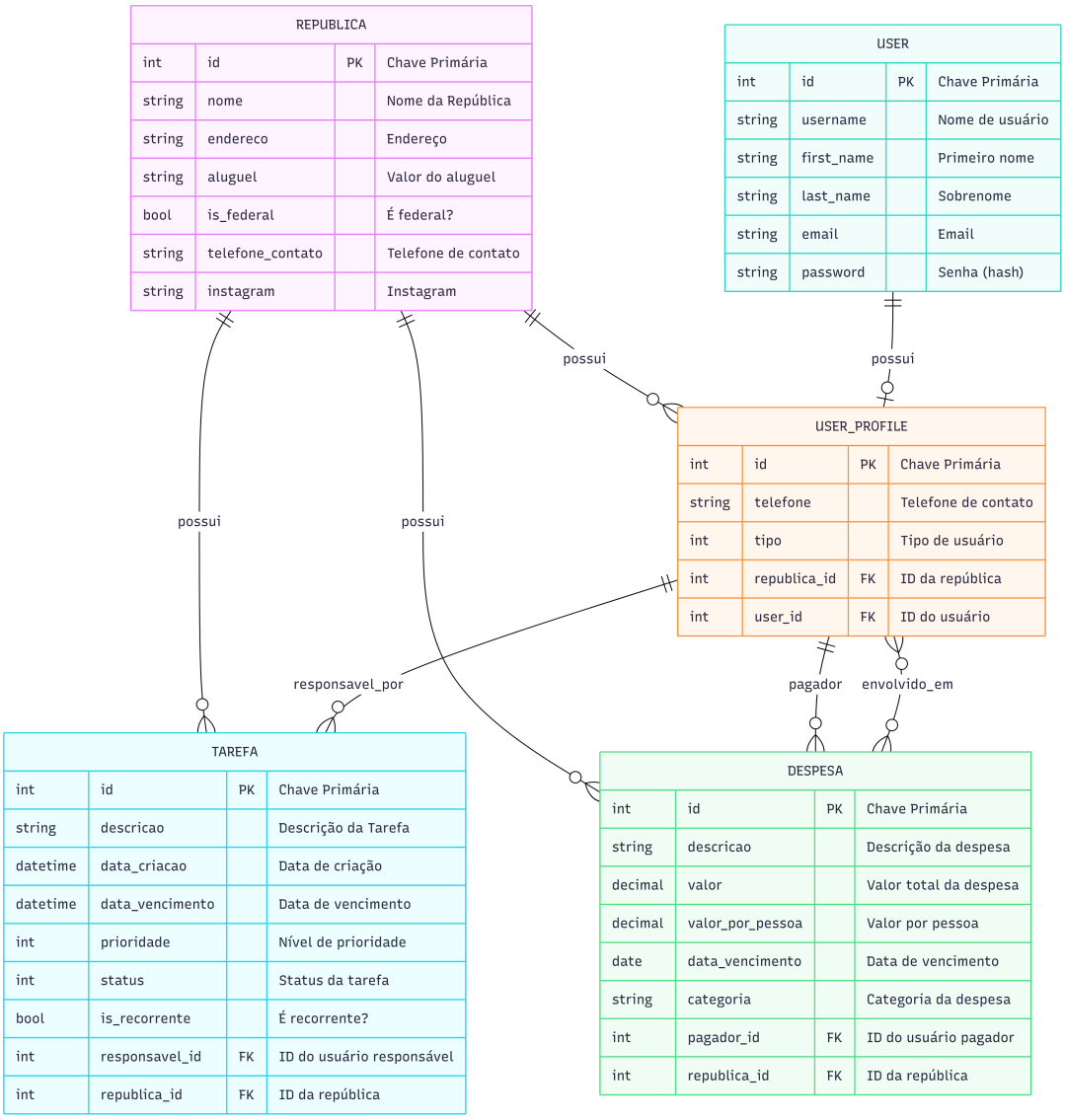

Figura 3.8 – Diagrama de Entidade-Relacionamento Estendido (ERE) Geral.

3.3.2 Módulo de Repúblicas

A Figura 3.9, que representa o DER do Módulo de Repúblicas e Usuários, ilustra a relação de um para muitos entre a entidade REPUBLICA e a entidade USER_PROFILE. Essa modelagem é

fundamental para a gestão do sistema, pois estabelece o vínculo de cada morador com uma única república. A entidade `REPUBLICA` é o ponto central, conectando-se a `USER_PROFILE`, `TAREFA` e `DESPESA`, o que indica que as tarefas e despesas são diretamente associadas à moradia em que ocorrem.

A entidade `REPUBLICA` contém atributos como *nome* e *endereco*, além de campos para informações de contato e redes sociais como *telefone_contato* e *instagram*. Um campo booleano *is_federal* distingue entre repúblicas federais e particulares. Essa estrutura de dados assegura que as tarefas e despesas sejam corretamente registradas sob a república à qual pertencem, e que os moradores estejam vinculados à sua respectiva moradia. A relação de um para muitos (*1:N*) entre `REPUBLICA` e `USER_PROFILE` garante a consistência, pois um morador só pode pertencer a uma república por vez.

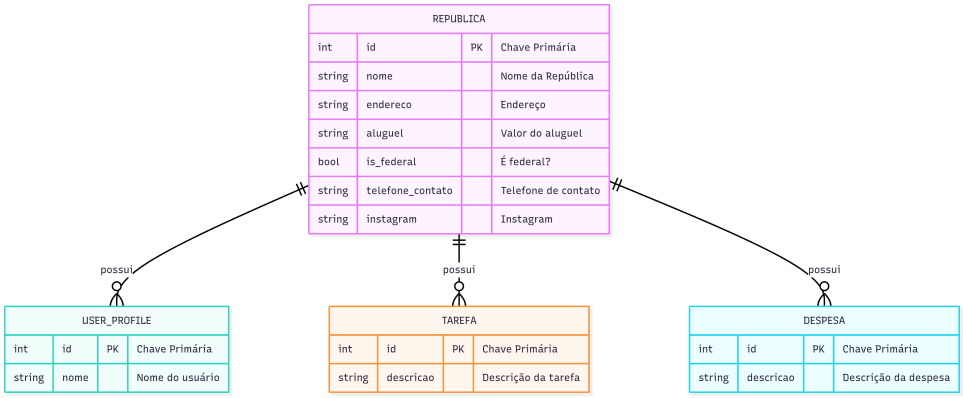

Figura 3.9 – DER: Módulo de Repúblicas e Usuários.

3.3.3 Módulo de Usuários

A modelagem deste módulo, apresentada na Figura 3.10, detalha a relação de um para um entre as entidades `User` (do Django) e `UserProfile`. Essa modelagem permite estender o modelo de usuário padrão do Django, que já gerencia a autenticação e dados básicos como *username*, *first_name*, *last_name*, *email* e *password*. A entidade `UserProfile` complementa esses dados com informações específicas do contexto do sistema, como *telefone* e *tipo* de usuário. A relação de muitos para um entre `UserProfile` e `REPUBLICA` (*republica_id FK*) assegura que cada morador esteja vinculado a uma república, garantindo que as informações do perfil do usuário sejam mantidas de forma organizada e consistente.

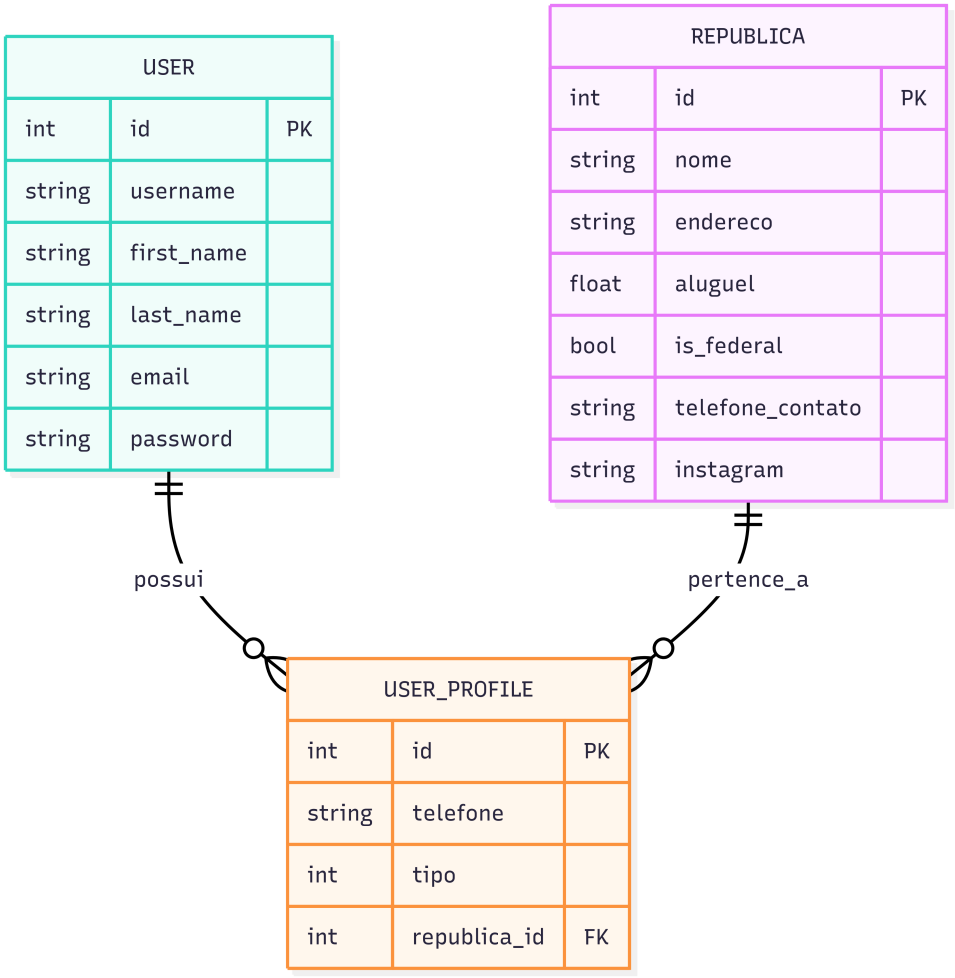

Figura 3.10 – DER: Módulo de Usuários.

3.3.4 Módulo de Tarefas

O DER do Módulo de Tarefas, apresentado na Figura 3.11, demonstra a estrutura de dados da entidade `TAREFA` e suas relações com a `REPUBLICA` e `USER_PROFILE`. A entidade `TAREFA` possui atributos como `id`, *descricao*, *data_criacao*, *data_vencimento*, *prioridade* e *status*. Ela está diretamente relacionada à entidade `REPUBLICA` através da chave estrangeira *republica_id* FK, o que garante que cada tarefa esteja vinculada à república correta. Além disso, a `TAREFA` se conecta a `USER_PROFILE` através da chave estrangeira *responsavel_id* FK, identificando o morador que é responsável por sua execução. Essa modelagem permite organizar as responsabilidades de forma clara, assegurando que as tarefas estejam corretamente atribuídas e sejam facilmente rastreáveis dentro do contexto de cada moradia.

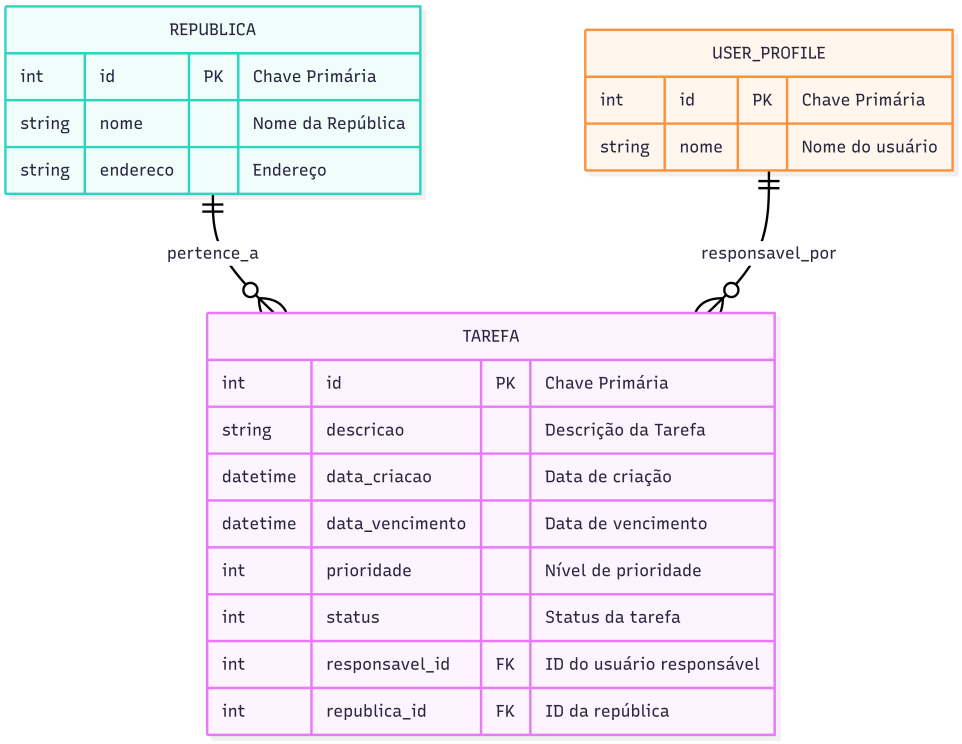

Figura 3.11 – DER: Módulo de Tarefas.

3.3.5 Módulo de Despesas

A modelagem do Módulo de Despesas, ilustrada na Figura 3.12, detalha a estrutura de dados da entidade `DESPESA` e suas relações com as entidades `REPUBLICA` e `USER_PROFILE`. A entidade `DESPESA` inclui atributos como *id*, *descricao*, *valor* (valor total da despesa), *valor_por_pessoa* (valor por pessoa) e *data_vencimento*. Ela se relaciona a uma `REPUBLICA` através da chave estrangeira *republica_id* FK, indicando a qual moradia a despesa pertence. Além disso, a `DESPESA` está conectada à entidade `USER_PROFILE` de duas formas: primeiramente, por uma chave estrangeira *pagador_id* FK que identifica o morador que realizou o pagamento, e, em segundo lugar, por uma relação de muitos para muitos, *envolvido_em*, que permite rastrear todos os moradores que devem participar da despesa. Essa estrutura robusta garante a transparência e a precisão no controle financeiro da moradia.

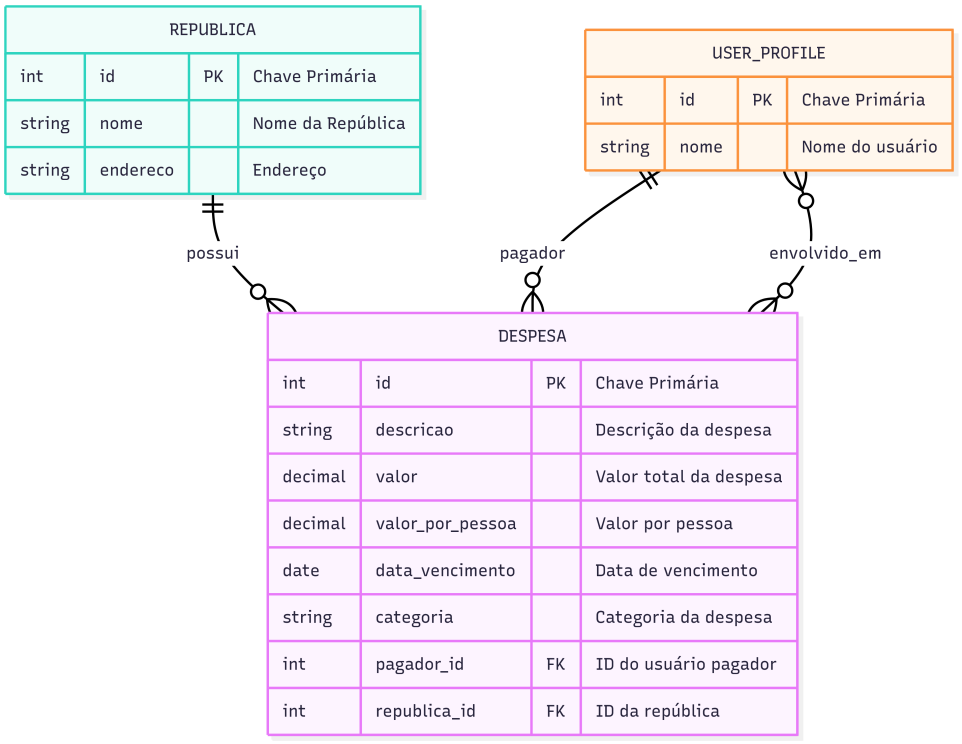

Figura 3.12 – DER: Módulo de Despesas.

3.4 Ferramentas e Tecnologias Utilizadas

O sistema foi implementado com a linguagem Python e o framework Django, selecionados por sua robustez e segurança, além de seu suporte para a organização modular do projeto. Para o frontend, foram utilizadas tecnologias como HTML, CSS e JavaScript, com o suporte do framework Bootstrap para garantir uma interface responsiva e amigável. A combinação dessas ferramentas permitiu uma construção ágil e eficiente, priorizando a usabilidade e a funcionalidade. Para a persistência de dados, o banco de dados SQLite foi empregado no ambiente de desenvolvimento, com a possibilidade de migração para o PostgreSQL em cenários de produção.

3.5 Processo de Desenvolvimento

O processo de desenvolvimento seguiu a abordagem ágil com base no framework Scrum, conforme discutido na revisão bibliográfica. A gestão das atividades foi estruturada em sprints, com um backlog de funcionalidades priorizadas que serviram como guia para cada ciclo de trabalho. Para o acompanhamento e a visualização do progresso, foi utilizado um quadro de Scrum, que permitiu monitorar o status das tarefas em diferentes etapas, como "A fazer", "Em andamento"e "Concluído". A visualização em quadro proporcionou clareza sobre o fluxo de trabalho e facilitou a comunicação e a colaboração durante a execução do projeto.

Tabela 3.3 – Cronograma de atividades propostas e realizadas para a Monografia II.

Mês	Atividade	Status
Maio	Ajustes recorrentes da Monografia 1 e Desenvolvimento dos diagramas	Realizado
Junho	Desenvolvimento dos módulos Usuários e Repúblicas	Realizado
Julho	Desenvolvimento dos Módulos de Despesas e Tarefas	Realizado
Agosto	Desenvolvimento do Dashboard, Redação final da Monografia II e entrega	Planejado

3.6 Detalhamento e Implementação dos Módulos

Nesta seção, é apresentado o detalhamento de cada módulo implementado no aplicativo, demonstrando como os requisitos funcionais foram traduzidos em soluções práticas e funcionais. Inicialmente, a arquitetura previu módulos isolados para cada funcionalidade. No entanto, durante o processo de desenvolvimento, a necessidade de uma tela central que oferecesse uma visão consolidada e imediata das métricas mais importantes da república se fez evidente. Essa adaptação, alinhada com os princípios de flexibilidade das metodologias ágeis, levou à criação do módulo de Dashboard. Ele atua como a tela principal do sistema, integrando visualizações e dados de todos os outros módulos para uma experiência de usuário mais eficiente e intuitiva.

3.6.1 Módulo de Gerenciamento de Repúblicas

Este módulo constitui a base do sistema, permitindo a gestão das repúblicas cadastradas. A criação de novas repúblicas é uma funcionalidade restrita aos administradores do sistema, sendo realizada exclusivamente através do painel de administração do Django, o que garante um controle centralizado e seguro. A edição pode ser realizada apenas pelos administradores da República, enquanto os demais usuários têm acesso somente à listagem, permitindo a visualização de informações como nome, endereço, contato e Instagram. A modelagem da entidade República foi realizada para refletir essas características, incluindo um campo para distinguir repúblicas federais de particulares. O controle de acesso foi implementado para permitir que apenas administradores do sistema e administradores de república (para sua própria república) possam editar os dados, garantindo a integridade das informações.

Para refletir a estrutura de dados planejada, o modelo `Republica` foi definido com atributos para armazenar as informações essenciais de cada moradia, conforme a figura 3.13. O uso de um `DecimalField` para o valor do aluguel e um `BooleanField` para identificar se a república é federal demonstra a adequação do modelo às regras de negócio específicas do contexto de Ouro Preto.

```python
class Republica(models.Model):
    nome = models.CharField(max_length=100)
    endereco = models.CharField(max_length=255)
    aluguel = models.DecimalField(max_digits=10, decimal_places=2)
    is_federal = models.BooleanField(default=False)
    telefone_contato = models.CharField(max_length=20, blank=True, null=True)
    instagram = models.CharField(max_length=50, blank=True, null=True)
    def __str__(self):
        return self.nome
```

Figura 3.13 – Código do modelo Republica

A interface de listagem de repúblicas foi concebida para exibir as informações de forma clara e organizada, utilizando um layout de cards que destaca os dados mais relevantes de cada moradia. O controle de acesso é aplicado diretamente na interface, exibindo o botão de "Editar"somente para os usuários que possuem a permissão necessária, conforme as regras de negócio.

3.6.2 Módulo de Gerenciamento de Moradores (UserProfile)

Este módulo é essencial para a organização da moradia, permitindo a gestão de usuários e seus perfis, e vinculando-os a repúblicas específicas. As funcionalidades de listagem de moradores, cadastro de novos usuários, edição de perfis e a exclusão de usuários foram desenvolvidas, com a lógica de permissões assegurando um controle rigoroso. A seção "Meu Perfil"foi implementada como um recurso para o próprio usuário gerenciar seus dados básicos, como nome e telefone, de forma segura e autônoma.

A modelagem de dados para os moradores utiliza o modelo `User` padrão do Django para autenticação, estendido pelo `UserProfile` para campos específicos como telefone, tipo de morador e vínculo com a república, conforme a Figura 3.14. Essa abordagem garante a segurança na gestão de senhas e permissões, ao mesmo tempo em que adiciona flexibilidade para os dados personalizados do perfil.

```python
class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='profile')
    telefone = models.CharField(max_length=20, blank=True, null=True)

    TIPO_USUARIO_CHOICES = [
        (1, 'Administrador da República'),
        (2, 'Morador Comum'),
    ]
    tipo = models.IntegerField(choices=TIPO_USUARIO_CHOICES, default=2)

    republica = models.ForeignKey(Republica, on_delete=models.CASCADE, related_name='moradores_associados')

    def __str__(self):
        return self.user.username
```

Figura 3.14 – Código do modelo UserProfile.

Para o cadastro e edição de usuários, foram desenvolvidos formulários Django específicos (`UserRegistrationForm` para criação e edição do perfil de terceiros e `ProfileEditForm` para

edição do próprio perfil). Esses formulários automatizam a validação dos dados e a interação com os modelos `User` e `UserProfile`, garantindo a consistência das informações.

A interface de listagem de moradores exibe os perfis em um layout de cards, organizados por república, proporcionando uma visão clara dos membros de cada moradia. Os botões de ação, como "Editar"e "Remover", são exibidos condicionalmente, refletindo as regras de controle de acesso que permitem que apenas administradores do sistema e administradores de república (para sua própria república) gerenciem os usuários.

3.6.3 Módulo de Gerenciamento de Tarefas

O módulo de tarefas é uma ferramenta central para a organização colaborativa, permitindo aos moradores gerenciar as responsabilidades diárias de forma eficiente. As funcionalidades de listagem, cadastro, edição e exclusão de tarefas foram implementadas, com a possibilidade de definir descrição, prioridade, status, data de vencimento e atribuir um responsável.

A entidade `Tarefa` foi modelada para capturar todos os atributos necessários, como descrição e datas, e se relaciona com a `República` (indicando a qual moradia a tarefa pertence) e com o `UserProfile` (identificando o morador responsável), , conforme a figura 3.15. Essa estrutura permite um acompanhamento detalhado das obrigações e facilita a atribuição de responsabilidades.

```python
class Tarefa(models.Model):
    descricao = models.TextField()
    data_criacao = models.DateTimeField(default=timezone.now)
    data_vencimento = models.DateTimeField(blank=True, null=True)

    PRIORIDADE_CHOICES = [
        (1, 'Baixa'),
        (2, 'Média'),
        (3, 'Alta'),
        (4, 'Urgente'),
    ]
    prioridade = models.IntegerField(choices=PRIORIDADE_CHOICES, default=2)

    STATUS_CHOICES = [
        (1, 'Pendente'),
        (2, 'Em Andamento'),
        (3, 'Concluída'),
        (4, 'Atrasada'),
        (5, 'Cancelada'),
    ]
    status = models.IntegerField(choices=STATUS_CHOICES, default=1) # Padrão: Pendente
    responsavel = models.ForeignKey(UserProfile, on_delete=models.SET_NULL, related_name='tarefas_responsavel', blank=True, null=True)
    republica = models.ForeignKey(Republica, on_delete=models.CASCADE, related_name='tarefas')

    def __str__(self):
        return self.descricao[:50]

    class Meta:
        verbose_name = "Tarefa"
        verbose_name_plural = "Tarefas"
        ordering = ['data_vencimento', 'prioridade']
```

Figura 3.15 – Código do modelo Tarefa.

Para a interação com as tarefas, o `TarefaForm` foi desenvolvido, oferecendo uma interface intuitiva para a criação e modificação dos registros. O formulário inclui campos para seleção de prioridade e status, além de um seletor para o responsável.

A interface de listagem de tarefas apresenta os itens em um formato de cards, destacando a descrição, data de vencimento, responsável e status, com indicadores visuais para a prioridade.

O controle de acesso é rigoroso, exibindo os botões de "Editar"e "Remover"apenas para administradores do sistema e administradores de república (para tarefas de sua própria república), garantindo que as responsabilidades sejam gerenciadas por quem tem autoridade.

3.6.4 Módulo de Controle de Despesas

Este módulo é crucial para a transparência financeira da república, permitindo o registro e o acompanhamento de despesas compartilhadas. As funcionalidades de listagem de gastos, cadastro de novos itens, edição de detalhes e exclusão foram implementadas, com a capacidade de calcular e dividir os valores automaticamente entre os moradores. Essa funcionalidade não é futura, mas parte do sistema já desenvolvido, tornando o controle financeiro mais preciso e alinhado aos objetivos do trabalho.

A entidade `Despesa` foi modelada para capturar informações como descrição, valor total, categoria e o morador pagador, conforme a figura 3.16. Para suportar a divisão automática, o modelo inclui o campo `valor_por_pessoa`, que armazena o valor calculado individualmente, e o relacionamento `ManyToManyField` `pessoas_envolvidas`, que rastreia os moradores que devem participar da despesa. Essa estrutura robusta permite uma organização financeira detalhada e serve como base para a visualização de gastos no Dashboard.

```python
class Despesa(models.Model):
    descricao = models.TextField()
    valor = models.DecimalField(max_digits=10, decimal_places=2)
    valor_por_pessoa = models.DecimalField(max_digits=10, decimal_places=2, blank=True, null=True)
    data_vencimento = models.DateField(blank=True, null=True)

    CATEGORIA_CHOICES = [
        ('Moradia', 'Moradia'),
        ('Contas', 'Contas'),
        ('Alimentação', 'Alimentação'),
        ('Transporte', 'Transporte'),
        ('Lazer', 'Lazer'),
        ('Outros', 'Outros'),
    ]
    categoria = models.CharField(max_length=50, choices=CATEGORIA_CHOICES, default='Outros')

    pagador = models.ForeignKey(UserProfile, on_delete=models.SET_NULL, related_name='despesas_pagas', blank=True, null=True)
    republica = models.ForeignKey(Republica, on_delete=models.CASCADE, related_name='despesas')

    # Campo para as pessoas envolvidas na despesa (Many-to-many)
    pessoas_envolvidas = models.ManyToManyField(UserProfile, related_name='despesas_envolvidas')

    data_registro = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return f"{self.descricao} - R$ {self.valor}"

    class Meta:
        verbose_name = "Despesa"
        verbose_name_plural = "Despesas"
        ordering = ['-data_registro']
```

Figura 3.16 – Código do modelo Despesa.

O formulário `DespesaForm` foi desenvolvido para facilitar a entrada de dados, incluindo campos para o valor, categoria e a seleção do morador pagador. Para a divisão, o formulário utiliza um campo `CheckboxSelectMultiple`, que permite ao usuário selecionar facilmente as pessoas envolvidas na despesa. Essa interface intuitiva simplifica o processo de registro financeiro e garante que a divisão seja feita de forma precisa.

A interface de listagem de despesas apresenta os registros de forma clara, com detalhes sobre o valor total, o valor individual por pessoa e a lista de moradores envolvidos na divisão. O controle de acesso para este módulo segue o padrão dos demais, permitindo que apenas usuários com permissão adequada possam gerenciar os registros financeiros, assegurando a integridade e a segurança das informações.

3.6.5 Módulo de Dashboard

O Dashboard atua como a tela principal e a central de comando do sistema, oferecendo aos usuários uma visão consolidada e imediata das métricas mais importantes da república. Foi modelado conforme a Figura 3.17. Diferentemente de uma lista estática, o painel foi concebido para ser uma ferramenta de consulta rápida, apresentando dados essenciais em cartões de resumo que destacam o número de tarefas pendentes, o total de despesas do mês e a data do próximo vencimento. Essa visualização imediata do status da república contribui para uma melhor organização do dia a dia e para a redução de conflitos, pois todos os moradores têm acesso às mesmas informações de forma transparente.

```python
@login_required
def dashboard_view(request, filter_by='all'):
    user = request.user
    user_profile = None
    republica_do_usuario = None

    try:
        user_profile = UserProfile.objects.get(user=user)
        republica_do_usuario = user_profile.republica
    except UserProfile.DoesNotExist:
        messages.error(request, 'Seu perfil de usuário não foi encontrado. Por favor, entre em contato com um administrador.')
        return redirect('logout')

    republica_foco = None
    if user.is_superuser or user.is_staff:
        republica_foco = republica_do_usuario if republica_do_usuario else (Republica.objects.first() if Republica.objects.exists() else None)
    else:
        republica_foco = republica_do_usuario

    dashboard_data = {}

    if republica_foco:
        dashboard_data['republica_nome'] = republica_foco.nome
        dashboard_data['republica_id'] = republica_foco.id
        dashboard_data['filter_by'] = filter_by

        tarefas_base = Tarefa.objects.filter(republica=republica_foco)
        despesas_base = Despesa.objects.filter(republica=republica_foco)

        if filter_by == 'mine':
            tarefas_base = tarefas_base.filter(responsavel=user_profile)
            despesas_base = despesas_base.filter(pagador=user_profile)
            dashboard_data['is_filtered_by_user'] = True
        else:
            dashboard_data['is_filtered_by_user'] = False

        dashboard_data['total_tarefas'] = tarefas_base.count()
        dashboard_data['tarefas_pendentes'] = tarefas_base.filter(status=1).count()
```

Figura 3.17 – Parte da interface do Dashboard com resumo de tarefas e despesas.

Para proporcionar uma análise financeira visual e intuitiva, o dashboard incorpora um gráfico de despesas por categoria. Esta visualização dinâmica, alimentada por dados reais do banco de dados, permite que os moradores compreendam rapidamente a distribuição dos gastos. A interação com o gráfico é facilitada por uma tooltip que exibe o valor exato de cada categoria, tornando a consulta de dados mais eficiente. A interface também permite que o usuário alterne

entre uma visão geral da república e uma visão pessoal, filtrando os dados para exibir apenas suas próprias tarefas e despesas.

A seção de próximas tarefas no dashboard foi aprimorada com uma visualização em formato de calendário em lista, onde as tarefas são agrupadas por data de vencimento. Essa abordagem oferece uma organização temporal clara e direta, similar a um calendário, sem a necessidade de uma implementação complexa de uma biblioteca de calendário. Cada item da lista apresenta detalhes importantes, como a descrição, o responsável e a prioridade visualmente destacada por meio de badges coloridos, facilitando a identificação das tarefas mais urgentes.

A estrutura do dashboard, com seus cartões de resumo, gráficos e listas de tarefas, foi implementada para que o sistema seja robusto e responsivo. A interface se adapta a diferentes tamanhos de tela, garantindo uma experiência de usuário consistente em dispositivos móveis e desktops. A combinação desses elementos visuais e a lógica de filtragem de dados em tempo real demonstram a aplicação prática dos conceitos de engenharia de software para criar uma solução funcional, usável e alinhada com as necessidades da comunidade estudantil.

4 Resultados e Discussão

4.1 Resultados da Implementação

O resultado do desenvolvimento é uma aplicação web funcional, coerente com os requisitos levantados e a arquitetura planejada, conforme os princípios de usabilidade e eficiência da engenharia de software. A interface do usuário foi construída para ser intuitiva e adaptada ao contexto das repúblicas estudantis, com um design que melhora a experiência e a acessibilidade da informação. A seguir, são apresentadas as interfaces principais do sistema, demonstrando como as funcionalidades foram implementadas para atender aos objetivos do projeto.

4.1.1 Dashboard Interativo

A tela inicial da aplicação, o Dashboard (Figura 4.1), atua como a central de comando, oferecendo aos usuários uma visão consolidada e imediata das métricas mais importantes da república. Essa arquitetura de painel foi escolhida para otimizar o fluxo de trabalho e reduzir a carga cognitiva, permitindo que os usuários compreendam o status da moradia em um único olhar. A visualização utiliza cards de resumo para destacar o número de tarefas pendentes, o total de despesas do mês e a data do próximo vencimento, alinhando-se ao requisito não funcional de usabilidade (RNF01) ao tornar a informação facilmente acessível.

Figura 4.1 – Interface do Dashboard, com resumo de tarefas e despesas.

Para proporcionar uma análise financeira visual e intuitiva, o dashboard incorpora um gráfico de despesas por categoria (Figura 4.2). Esta visualização, alimentada por dados reais do banco de dados, permite que os moradores compreendam rapidamente a distribuição dos gastos, um recurso que atende diretamente ao requisito funcional RF05.

Figura 4.2 – Detalhe da seção "Despesas por Categoria"no Dashboard.

A seção de próximas tarefas foi aprimorada com uma visualização em formato de calendário em lista, onde os itens são agrupados por data de vencimento, conforme a Figura 4.3. Essa solução de design, inspirada em ferramentas como o Trello, demonstra a aplicação de padrões de usabilidade para facilitar o acompanhamento eficiente das responsabilidades.

Figura 4.3 – Detalhe da seção "Próximas Tarefas"no Dashboard.

4.1.2 Listagem de Repúblicas

O módulo de gerenciamento de repúblicas é apresentado por meio de uma interface de listagem clara e organizada, conforme a Figura 4.4. A visualização em cards, com um design que destaca os dados mais relevantes, como nome e endereço, é uma solução que prioriza a estética e a eficiência da comunicação visual, conforme o requisito RNF01. O controle de acesso, implementado de forma rigorosa na camada de backend, é demonstrado pela presença condicional do botão "Editar", que só é exibido para usuários com permissão, alinhando o sistema às boas práticas de segurança e autorização.

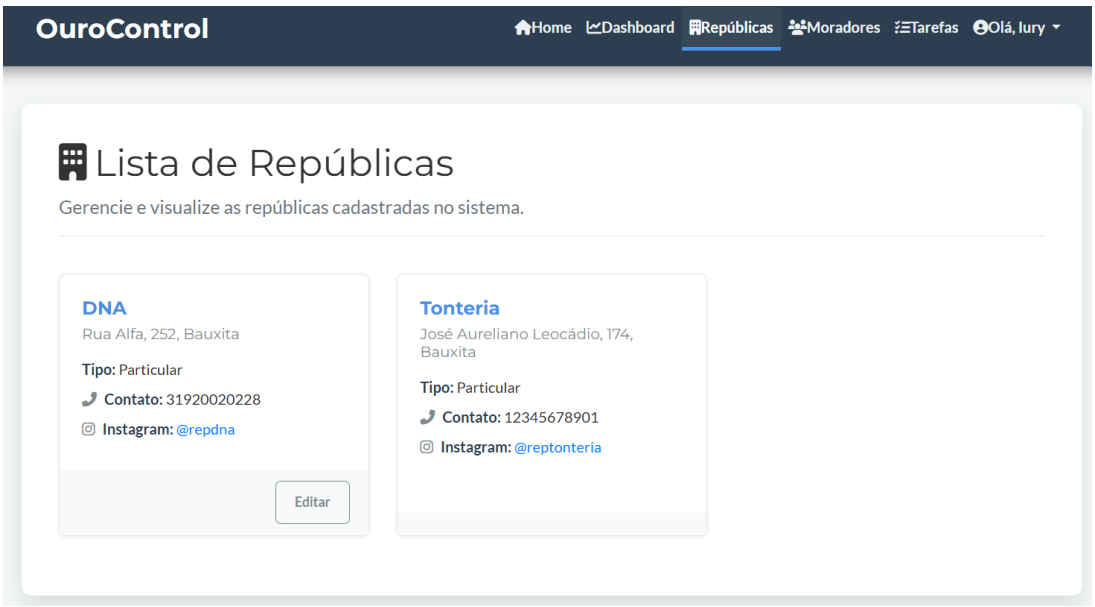

Figura 4.4 – Interface de listagem de repúblicas, com a visualização em cards.

4.1.3 Listagem de Moradores

A listagem de moradores foi concebida para refletir a estrutura organizacional das repúblicas, conforme a Figura 4.5. Os perfis são apresentados em cards, agrupados por moradia, o que facilita a identificação dos membros de cada comunidade. O layout e a organização das informações, que incluem nome, email e tipo de usuário, visam melhorar a navegabilidade e a compreensão da dinâmica entre os moradores. A implementação de controle de acesso nos botões de "Editar"e "Remover"garante que o gerenciamento de usuários seja realizado apenas por quem tem a devida autoridade, atendendo ao requisito funcional RF01.

Figura 4.5 – Interface de listagem de moradores, com os perfis agrupados por república.

4.1.4 Módulo de Gerenciamento de Tarefas

A interface do módulo de tarefas (Figura 4.6) é uma ferramenta central para a organização colaborativa, implementada para atender ao requisito funcional RF03. A listagem de tarefas é apresentada em cards que destacam a descrição, o responsável, a data de vencimento e o status. Indicadores visuais em forma de badges coloridos são utilizados para sinalizar a prioridade de cada tarefa, facilitando a identificação rápida dos itens mais urgentes e assegurando um acompanhamento eficiente das responsabilidades. O design visualmente enriquecido contribui para o engajamento e a produtividade, aspectos considerados na revisão bibliográfica com base em trabalhos relacionados.

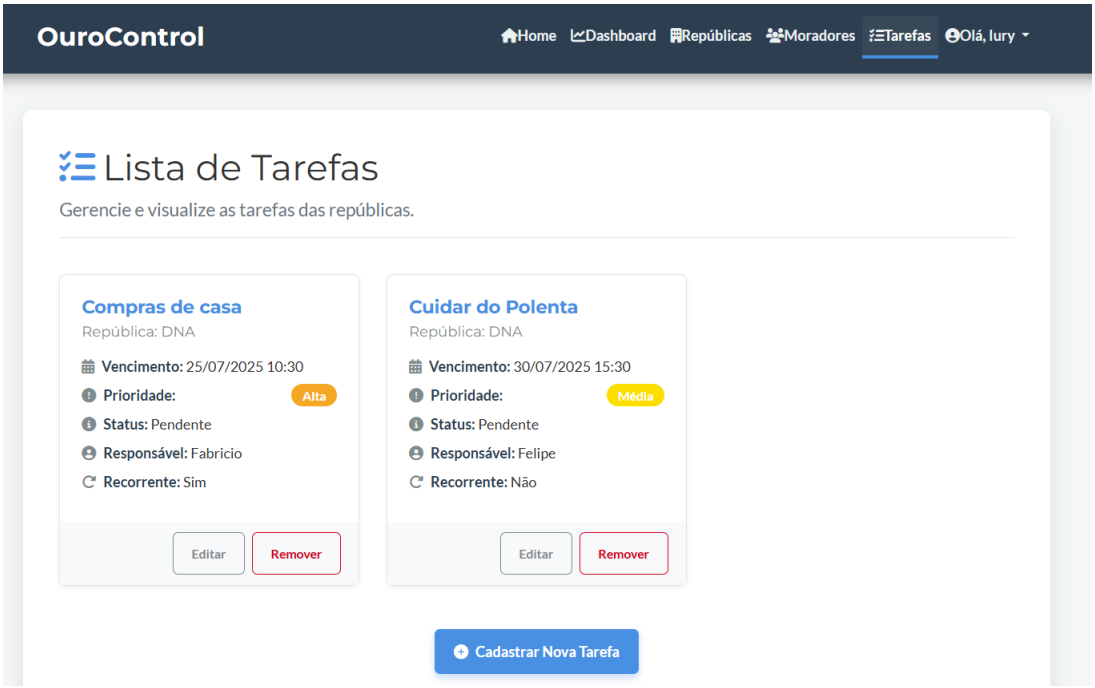

Figura 4.6 – Interface de listagem de tarefas, com indicadores visuais de prioridade.

4.1.5 Módulo de Controle de Despesas

A seção de despesas foi desenvolvida com foco na transparência financeira da república, atendendo ao requisito funcional RF05. A interface de listagem (Figura 4.7) apresenta os registros em cards que detalham o valor total e o valor individual por pessoa, além da categoria, data de vencimento e o morador pagador. Essa visualização clara e concisa é fundamental para o controle financeiro, permitindo uma gestão colaborativa e promovendo um ambiente de confiança entre os moradores. A lógica de divisão automática do valor, implementada no backend, demonstra uma aplicação prática dos conceitos de engenharia de software para resolver um problema recorrente em moradias compartilhadas.

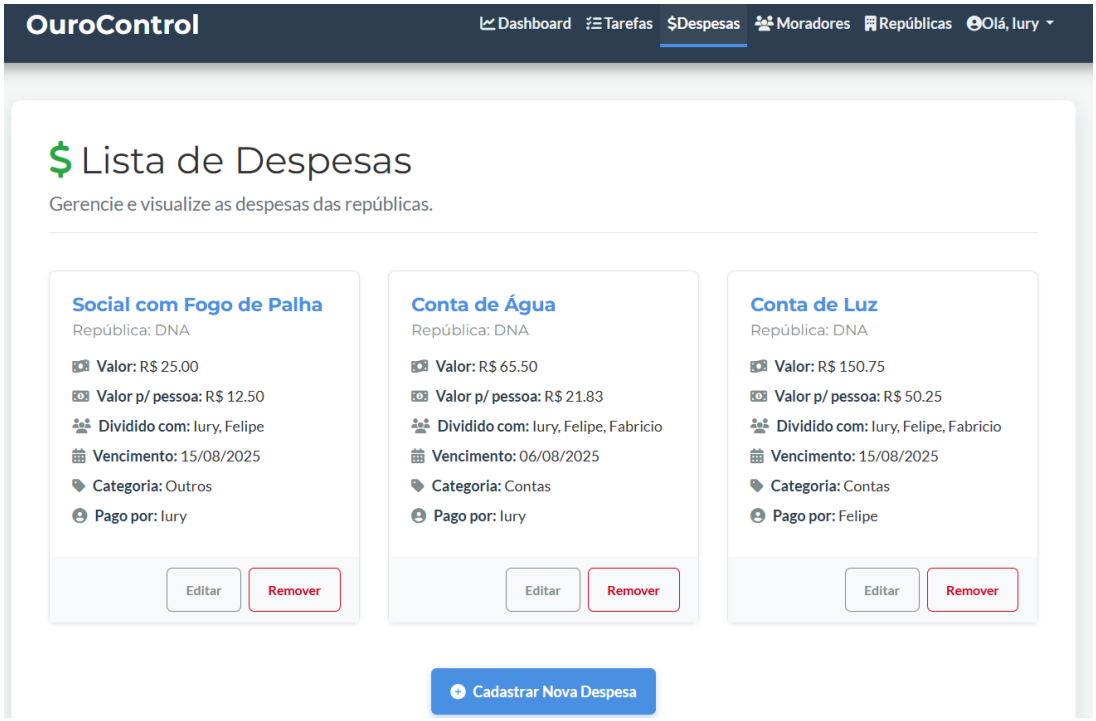

Figura 4.7 – Interface de listagem de despesas, com detalhamento dos valores e da divisão.

4.1.6 Formulários de Cadastro e Edição

Os formulários de cadastro e edição (Figura 4.8) foram projetados para serem intuitivos e eficientes, minimizando a curva de aprendizado do usuário. A validação dos dados, feita na camada do servidor conforme os princípios da engenharia de software, garante a integridade da base de dados. O design dos formulários, alinhado à estética do aplicativo, utiliza campos com labels claras, placeholders e botões de ação que guiam o usuário de forma lógica durante o processo de gerenciamento das informações, contribuindo diretamente para o requisito não funcional de usabilidade (RNF01).

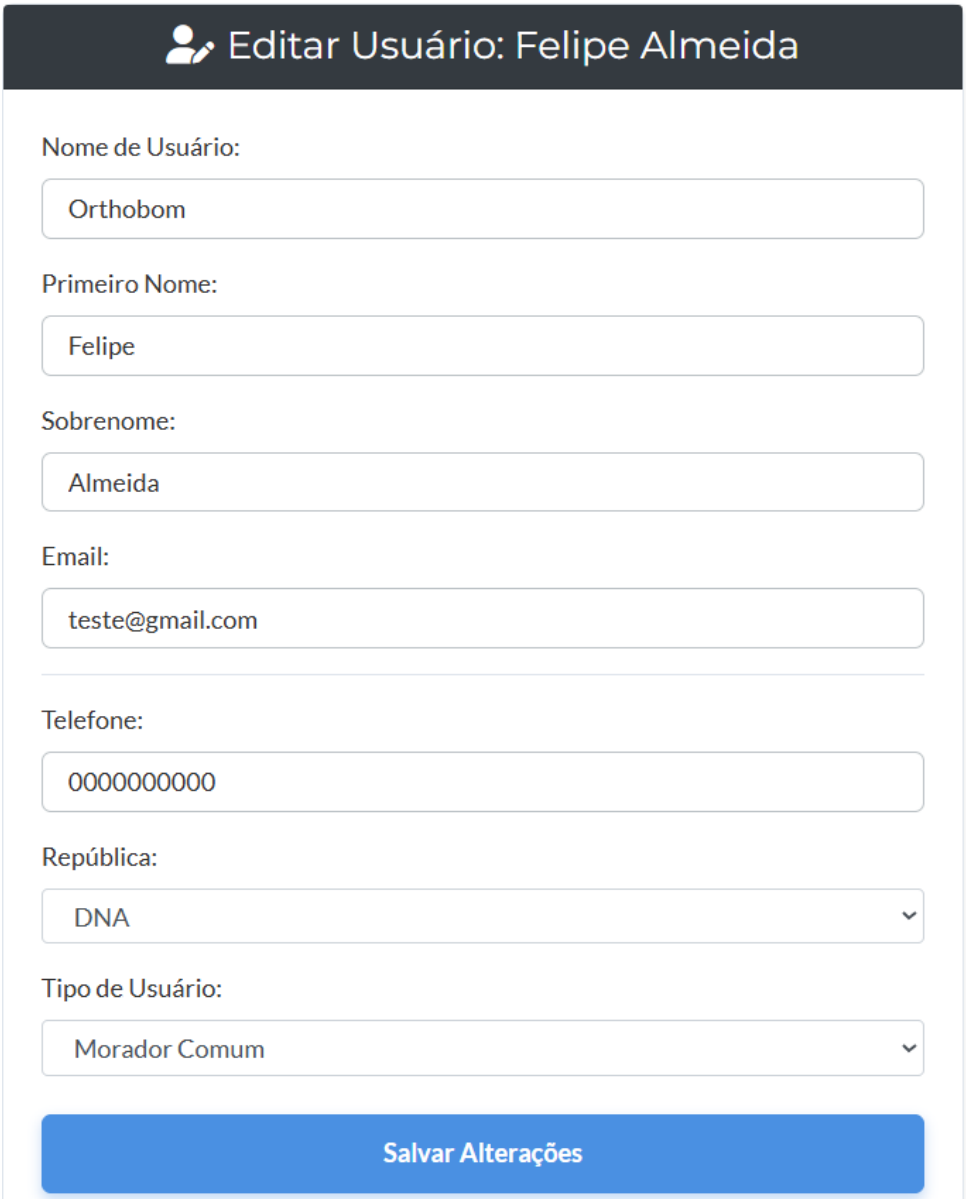

Figura 4.8 – Exemplo de formulário de edição de usuário.

5 Considerações Finais

Assim, segue-se as considerações finais sobre o trabalho realizado, sintetizando os principais pontos do desenvolvimento do sistema. A análise se concentra na conclusão obtida a respeito da construção e usabilidade da interface, avaliando como o design e as funcionalidades implementadas contribuíram para a experiência do usuário. Em seguida, são discutidos os possíveis desdobramentos e continuações deste trabalho, delineando caminhos para aprimoramentos futuros que podem expandir o escopo e o valor da aplicação.

5.1 Conclusão

Deste modo, o presente trabalho buscou, por meio dos princípios da engenharia de software, propor e desenvolver uma solução tecnológica para o gerenciamento de repúblicas estudantis na cidade de Ouro Preto. A proposta surgiu da observação de desafios recorrentes na organização de tarefas e no controle financeiro, evidenciando a necessidade de uma ferramenta que promovesse a colaboração e a transparência. Através da aplicação de metodologias ágeis e da modelagem com Diagramas de Entidade-Relacionamento Estendidos (ERE), foi possível estruturar um sistema robusto e funcional, alinhado com as boas práticas do desenvolvimento de software.

A implementação da aplicação, utilizando Python e o framework Django, demonstrou ser uma escolha adequada, provendo uma base segura e escalável para todos os módulos. O design da interface, com uma estética "Moderna e Clean", contribuiu para a usabilidade do sistema, atendendo ao requisito não funcional de experiência do usuário. O desenvolvimento dos módulos de Repúblicas, Moradores, Tarefas e Despesas, com o devido controle de acesso, garantiu que o sistema fosse capaz de atender aos requisitos funcionais levantados e resolver os problemas identificados no cotidiano das moradias.

Diante do que foi implementado, os objetivos propostos inicialmente foram alcançados. O aplicativo web desenvolvido é capaz de auxiliar na gestão das repúblicas, integrando as funcionalidades de organização de tarefas e controle financeiro em um ambiente único e centralizado. A contribuição prática do trabalho reside em oferecer uma ferramenta que minimiza conflitos, equilibra responsabilidades e melhora a comunicação entre os moradores.

5.2 Trabalhos Futuros

Embora o sistema se mostre funcional e coerente com os objetivos, existem caminhos para aprimoramentos e novas funcionalidades que podem ser explorados em trabalhos futuros. A

visualização de vencimentos no Dashboard atende parcialmente ao requisito funcional RF08. No entanto, a implementação de um sistema de notificações automáticas via e-mail ou SMS seria uma melhoria crucial para alertar proativamente os moradores sobre contas e tarefas pendentes. A criação de um módulo de relatórios financeiros mais detalhados e a possibilidade de exportação de dados também trariam um valor adicional significativo ao aplicativo. Adicionalmente, a funcionalidade de recorrência de tarefas, que foi parcialmente implementada, poderia ser estendida para um sistema de agendamento em segundo plano, utilizando ferramentas como o Celery.

A princípio, o projeto contemplava um módulo de relatórios como uma funcionalidade central. No entanto, a implementação desse módulo foi estrategicamente descontinuada para priorizar as funcionalidades de gestão que já foram desenvolvidas. A criação de relatórios de atividades e gastos, com a possibilidade de exportação de dados, permanece como uma importante sugestão para o futuro, permitindo uma análise mais aprofundada da dinâmica de cada república. Essa decisão de adaptação demonstra a aplicação de uma abordagem pragmática no desenvolvimento do projeto, garantindo a entrega de uma solução viável dentro do cronograma previsto.

Referências

ANSELL, C.; GASH, A. *The collaborative governance handbook*. [S.l.]: SAGE Publications, 2018.

Atlassian. *Trello: Gerenciamento de Projetos e Tarefas*. 2024. Acesso em: 4 dez. 2024. Disponível em: <https://trello.com/>.

BECK, K. e. a. *Manifesto Ágil*. 2001. Disponível em: <https://agilemanifesto.org/>.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *Unified Modeling Language User Guide*. [S.l.]: Pearson Education, 2005.

CONQUER, E. *TODOIST: A ferramenta que vai te ajudar a organizar a sua vida*. 2019. Acessado em: 28 de outubro de 2019. Disponível em: <https://escolaconquer.com.br/blog/todoist/>.

EMILIANO, M. G. B. *Implementação e análise de software com boa usabilidade para gerência financeira de repúblicas estudantis*. Dissertação (Mestrado) — Universidade Federal de Ouro Preto, 2023. Acesso em: 4 dez. 2024. Disponível em: <https://www.monografias.ufop.br/handle/35400000/6066?mode=full>.

FOWLER, M. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. [S.l.]: Addison-Wesley Professional, 2004.

FREEMAN, E.; FREEMAN, E. *Use a Cabeça! Desenvolvimento Web*. [S.l.]: Alta Books, 2014.

MACHADO, O. L. As repúblicas estudantis da universidade federal de ouro preto, brasil. *Revista Crítica de Ciências Sociais*, n. 66, p. 197–199, 2003.

MIRANDA, R. S. *Sistema Web para Gerenciamento de um Escritório de Arquitetura*. Dissertação (Mestrado) — Universidade Federal de Ouro Preto, 2022. Acesso em: 4 dez. 2024. Disponível em: <https://www.monografias.ufop.br/bitstream/35400000/6066/6/MONOGRAFIA_SistemaWebGerenciamento.pdf>.

NEVES, I. S. S. J. das. *Desenvolvimento de uma Plataforma Web para Gestão de Condomínios de Pequeno Porte*. Dissertação (Mestrado) — Universidade Federal de Ouro Preto, 2023. Acesso em: 4 dez. 2024. Disponível em: <https://www.monografias.ufop.br/handle/35400000/6066>.

Object Management Group (OMG). *Unified Modeling Language (UML) Specification*. 2023. Disponível em: <https://www.omg.org/spec/UML/>.

RODRIGUES, R. M. D. *Tarefaça: Ferramenta Gamificada para Gerenciamento de Tarefas*. 2021. Monografia - Universidade Federal de Ouro Preto. Disponível em: <https://www.monografias.ufop.br/bitstream/35400000/3018/9/MONOGRAFIA_Tarefa%c3%a7aFerramentaGamificada.pdf>.

SMOBILEAPPS. *Bills Organizer Free*. 2013. Disponível em: <https://play.google.com/store/apps/details?id=com.smapps.android.bills.trail>.

Splitwise. *Splitwise: Compartilhe contas e divida despesas*. 2024. Acesso em: 4 dez. 2024. Disponível em: <https://www.splitwise.com/>.

TORFING, J. *Collaborative governance as a source of public value: The value proposition of public-private-civil partnerships*. [S.l.]: Routledge, 2020.

Universidade Federal de Ouro Preto. *Repúblicas Estudantis - UFOP*. 2014. Acesso em: 4 dez. 2024. Disponível em: <https://web.archive.org/web/20180323195637/http://www.prace.ufop.br/index.php/assistencia-estudantil/2012-11-08-17-57-05/2014-09-18-19-04-10/2014-09-18-19-07-01>.