

Universidade de Brasília - UnB  
Faculdade de Ciências e Tecnologias em Engenharia - FCTE

Algiz - EcoNet

# DOCUMENTO DE ARQUITETURA

Versão 1.3.0

Autores: Ana Beatriz Cunha Wirthmann, Arthur de Lima Sobreira, Caua Nicolas Pereira de Oliveira, Diogo Oliveira Ferreira, Gabriel Augusto Vilarinho Viana Rocha, Gabriel Pereira da Silva, João Victor Sousa Soares e Silva, João Pedro Sampaio Maciel, Julia Oliveira Patricio, Mariana Ribeiro Santana Gonzaga, Rafael Siqueira Soares, Tiago Scherrer Tavares de Lyra  
Orientador: Prof. Ricardo Ajax Dias Kosloski

Brasília, DF  
2025



Ana Beatriz Cunha Wirthmann, Arthur de Lima Sobreira, Caua Nicolas Pereira de Oliveira, Diogo Oliveira Ferreira, Gabriel Augusto Vilarinho Viana Rocha, Gabriel Pereira da Silva, João Victor Sousa Soares e Silva, João Pedro Sampaio Maciel, Julia Oliveira Patricio, Mariana Ribeiro Santana Gonzaga, Rafael Siqueira Soares, Tiago Scherrer Tavares de Lyra

# DOCUMENTO DE ARQUITETURA

Trabalho submetido à disciplina de Métodos  
De desenvolvimento de Software da Universidade  
de Brasília

Universidade de Brasília - UnB  
Faculdade de Ciências e Tecnologias em Engenharia - FCTE

Orientador: Prof. Ricardo Ajax Dias Kosloski

Brasília, DF  
2025

# Resumo

Este documento descreve a arquitetura do sistema EcoNet, uma aplicação desenvolvida para promover e otimizar práticas sustentáveis, facilitando o cadastro e a gestão de ações ecológicas, e apoiando uma economia circular eficiente e segura entre empresas. A arquitetura do sistema adota o padrão MVC (Modelo, Visão, Controller) combinado com o estilo N-Camadas, especificamente em três níveis: Apresentação, Negócio e Persistência (ou Dados). Essa escolha visa assegurar uma clara separação de responsabilidades, otimizando a organização do código, a manutenção, a escalabilidade e a evolução do software.

O processo de desenvolvimento do EcoNet é gerenciado pela metodologia SCRUM, com sprints semanais que permitem entregas iterativas, avaliação contínua e adaptação aos requisitos evolutivos do projeto. As metas arquitetônicas propostas incluem, garantir uma vida útil prolongada com capacidade de evolução tecnológica, responsividade em navegadores como Google Chrome e Microsoft Edge, e criptografia de senhas. A implementação do software utiliza JavaScript e React para o frontend, MySQL como banco de dados, e Docker para implantação, assegurando portabilidade e facilidade de manutenção e evolução. O sistema atende a diversos perfis de usuários (como Administrador, Dono de Empresa e Gestor de Contratos) com permissões específicas, facilitando a interação e a gestão de funcionalidades.

**Palavras-chave:** EcoNet, Arquitetura de Software, MVC, N-Camadas, SCRUM, Sustentabilidade, Economia Circular, MySQL, React, Docker.

## Lista de ilustrações

Figura 1: Diagrama esquemático da representação arquitetural.....	15
Figura 2: Diagrama de caso de uso para os cargos de dono de empresa, vice-dono de empresa, gestor, comunicador e usuário comum.....	17
Figura 3: Diagrama de caso de uso para o cargo de administrador.....	18
Figura 4: Diagrama de estados.....	20
Figura 5: Diagrama de atividades.....	22
Figura 6: Diagrama de classes.....	25
Figura 7: Desenho esquemático da visão de implementação.....	27
Figura 8: Diagrama de Pacotes do Projeto Econet.....	28
Figura 9: Protótipo da Página Home.....	29
Figura 10: Continuação do Protótipo da Página Home.....	29
Figura 11: Protótipo da Página Login.....	30
Figura 12: Protótipo da Página Sobre Nós.....	30

## Lista de tabelas

Tabela 1: Tabela de Integrantes do Grupo.....	9
Tabela 2: Tabela de Contribuição.....	10
Tabela 3: Tabela de histórico de revisões.....	11

## Lista de abreviaturas e siglas

UnB:	Universidade de Brasília
FCTE:	Faculdade de Ciências e Tecnologias em Engenharia
MVC:	Modelo, Visão, Controller
N-Camadas:	Estilo arquitetural de sistema em N níveis (Apresentação, Negócio e Persistência)
SCRUM:	Metodologia ágil de desenvolvimento de software
MySQL:	Sistema de gerenciamento de banco de dados
Docker:	Plataforma de contêineres para implantação
UML:	Unified Modeling Language (Linguagem de Modelagem Unificada)
API:	Application Programming Interface (Interface de Programação de Aplicações)
REST:	Representational State Transfer (Estilo arquitetural para APIs, (REST APIs))
HTTP:	Hypertext Transfer Protocol
JDBC:	Java Database Connectivity (Mencionado como forma de acesso a banco de dados)
ORM:	Object-Relational Mapping (Mapeamento Objeto-Relacional)
CSS:	Cascading Style Sheets
MDS:	Métodos de Desenvolvimento de Software
DF:	Distrito Federal

# Sumário

<b>1 ORGANIZAÇÃO DO GRUPO.....</b>	<b>8</b>
1.1 Tabela de integrantes.....	9
Tabela 1: Tabela de Integrantes do Grupo.....	9
1.2 Tabela de contribuição.....	10
Tabela 2: Tabela de Contribuição.....	10
1.3 Histórico de revisões.....	11
Tabela 3: Tabela de histórico de revisões.....	11
<b>2 INTRODUÇÃO.....</b>	<b>12</b>
2.1 Propósito.....	12
2.2 Escopo.....	12
<b>3 REPRESENTAÇÃO ARQUITETURAL.....</b>	<b>12</b>
3.1 Definições.....	12
3.2 Justifique sua escolha.....	13
3.3 Detalhamento.....	13
Figura 1: Diagrama esquemático da representação arquitetural.....	15
3.4 Metas e restrições arquiteturais.....	16
3.5 Visão de Casos de uso (escopo do produto).....	16
Figura 2: Diagrama de caso de uso para os cargos de dono de empresa, vice-dono de empresa, gestor, comunicador e usuário comum.....	17
Figura 3: Diagrama de caso de uso para o cargo de administrador.....	18
3.6 Visão lógica.....	18
3.6.1 Diagrama de estados.....	20
Figura 4: Diagrama de estados.....	20
3.6.2 Diagrama de atividades.....	22
Figura 5: Diagrama de atividades.....	22
3.6.3 Diagrama de classes.....	25
Figura 6: Diagrama de classes.....	25
3.7 Visão de Implementação.....	27
Figura 7: Desenho esquemático da visão de implementação.....	27
Figura 8: Diagrama de Pacotes do Projeto Econet.....	28
Figura 9: Protótipo da Página Home.....	29
Figura 10: Continuação do Protótipo da Página Home.....	29
Figura 11: Protótipo da Página Login.....	30
Figura 12: Protótipo da Página Sobre Nós.....	30
3.8 Visão de Implantação.....	31
3.9 Restrições adicionais.....	31
<b>4.0 BIBLIOGRAFIA.....</b>	<b>32</b>

# 1 ORGANIZAÇÃO DO GRUPO

## 1.1 Tabela de integrantes

Tabela 1: Tabela de Integrantes do Grupo

<b>Matrícula</b>	<b>Nome</b>	<b>Função (responsabilidade)</b>
190139048	Ana Beatriz Cunha Wirthmann	Documentação, Desenvolvedor Front-End
231026797	Arthur de Lima Sobreira	Desenvolvedor Front-End
231026670	Caua Nicolas Pereira de Oliveira	Desenvolvedor Front-End
231011266	Diogo Oliveira Ferreira	Desenvolvedor Back-End
221022533	Gabriel Augusto Vilarinho Viana Rocha	Project Owner, Desenvolvedor Full-Stack
221008641	Gabriel Pereira da Silva	Documentação
221008196	João Victor Sousa Soares e Silva	Banco de Dados, Desenvolvedor Back-End
211039528	João Pedro Sampaio Maciel	Banco de Dados, Desenvolvedor Back-End
231027140	Julia Oliveira Patricio	Testes de Software, Desenvolvedor Back-End
231026993	Mariana Ribeiro Santana Gonzaga	Documentação, Desenvolvedor Back-End
221022195	Rafael Siqueira Soares	Testes de Software, Desenvolvedor Back-End
190048549	Tiago Scherrer Tavares de Lyra	Desenvolvedor Front-End



## 1.2 Tabela de contribuição

Tabela 2: Tabela de Contribuição

<b>Matrícula</b>	<b>Nome</b>	<b>Pontos de contribuição</b>
190139048	Ana Beatriz Cunha Wirthmann	8
231026797	Arthur de Lima Sobreira	8
231026670	Caua Nicolas Pereira de Oliveira	8
231011266	Diogo Oliveira Ferreira	8
221022533	Gabriel Augusto Vilarinho Viana Rocha	10
221008641	Gabriel Pereira da Silva	10
221008196	João Victor Sousa Soares e Silva	8
211039528	João Pedro Sampaio Maciel	8
231027140	Julia Oliveira Patricio	8
231026993	Mariana Ribeiro Santana Gonzaga	8
221022195	Rafael Siqueira Soares	8
190048549	Tiago Scherrer Tavares de Lyra	8

### 1.3 Histórico de revisões

Tabela 3: Tabela de histórico de revisões

<b>Data</b>	<b>Versão</b>	<b>Descrição</b>	<b>Autor</b>
28/05/2025	1.0.0	Versão inicial do documento de arquitetura	Todos integrantes
10/06/2025	1.1.0	Adição de detalhes sobre MVC, aprimoramento de restrições arquitetônicas e refinamento de diagramas e textos, mantendo a estrutura base.	Gabriel Augusto Vilarinho Viana Rocha e Gabriel Pereira da Silva
12/06/2025	1.1.1	Refinamento dos diagramas de Estados e Classes para maior clareza visual, sem alterações na arquitetura ou requisitos.	Gabriel Augusto Vilarinho Viana Rocha e Gabriel Pereira da Silva
13/06/2025	1.2.0	Adição de novo artefato (Diagrama de Atividades) e expansão significativa do Diagrama de Classes.	Gabriel Augusto Vilarinho Viana Rocha e Gabriel Pereira da Silva
04/07/2025	1.3.0	Esta versão introduz aprimoramentos significativos no detalhamento técnico e visual do documento, incluindo novas seções como organização da equipe, ferramentas adicionais e protótipos melhor descritos.	Gabriel Augusto Vilarinho Viana Rocha, Gabriel Pereira da Silva, Diogo Oliveira Ferreira

## 2 INTRODUÇÃO

### 2.1 Propósito

- Este documento descreve a arquitetura do sistema EcoNet, sendo desenvolvido pelo grupo, na disciplina de MDS – Métodos de Desenvolvimento de Software – edição do primeiro semestre de 2025. O objetivo é fornecer uma visão abrangente da arquitetura do sistema para desenvolvedores, testadores e demais interessados, principalmente no que se refere às tecnologias adotadas, às decisões arquitetônicas e à organização dos componentes do sistema.

### 2.2 Escopo

- O detalhamento do escopo encontra-se no documento "Visão do Produto - EcoNet", já entregue, juntamente com a Declaração de Escopo do Produto. De maneira geral, o escopo do EcoNet compreende o desenvolvimento de uma aplicação destinada a promover e otimizar práticas sustentáveis, facilitando o cadastro e a gestão de ações ecológicas, além de fornecer ferramentas que ajudam em áreas relacionadas à sustentabilidade. O sistema buscará automatizar processos, fornecendo modelos de contratos e apoiando a tomada de decisões por parte dos usuários e administradores.

## 3 REPRESENTAÇÃO ARQUITETURAL

### 3.1 Definições

- O sistema seguirá a arquitetura MVC (Modelo, Visão, Controller), que é amplamente utilizada em frameworks de sucesso por promover uma clara separação de responsabilidades. No padrão MVC, o **Modelo** gerencia os dados e as regras de negócio, a **Visão** é responsável pela interface com o usuário, e o **Controller** atua como intermediário, interpretando as entradas do usuário e coordenando as interações entre as camadas.

Adicionalmente, a escolha pelo estilo arquitetural em N-Camadas, com  $N = 3$ , reforça essa divisão estrutural do sistema em três níveis principais:

- **Camada de Apresentação:** Responsável por exibir a interface e interagir com o usuário, integrando os aspectos visuais e de usabilidade.

- **Camada de Negócio:** Onde as regras, a lógica de validação e o processamento dos dados são definidos, garantindo que a aplicação cumpra seus propósitos funcionais.
- **Camada de Persistência:** Dedicada ao acesso, armazenamento e recuperação dos dados, assegurando a integridade e a persistência das informações.

Essa abordagem modular não apenas facilita a manutenção e a escalabilidade do sistema, como também permite que equipes trabalhem de forma paralela em diferentes camadas, reduzindo o acoplamento e aumentando a coesão. A representação arquitetural pode ser complementada por diagramas UML (como diagramas de componentes e de classes) que auxiliam na visualização e na comunicação das decisões de design junto aos stakeholders.

### 3.2 Justifique sua escolha

- A utilização da arquitetura MVC combinada com o estilo N-Camadas (com  $N = 3$ ) é justificada pela sua capacidade de promover uma clara separação de responsabilidades dentro do sistema. Essa divisão em camadas — Apresentação, Negócio e Persistência(Dados) — facilita a manutenção, a escalabilidade e a evolução do software, pois permite que alterações em uma camada possam ser realizadas sem impactos diretos nas demais. Além disso, a separação entre o Modelo, a Visão e o Controller (no padrão MVC) garante que a lógica de negócios, a interface com o usuário e o controle das interações fiquem isolados, otimizando a organização do código e favorecendo o trabalho colaborativo entre equipes especializadas. Assim, essa abordagem modular melhora a eficiência do desenvolvimento e a adaptabilidade do sistema às mudanças.

### 3.3 Detalhamento

- Para o EcoNet, combinamos SCRUM (gestão ágil), MVC (organização do código) e uma arquitetura em 3 camadas (apresentação, negócio e persistência). Essa abordagem garante desenvolvimento iterativo, código modular e fácil manutenção, com entregas alinhadas aos requisitos evolutivos do projeto.

- **Organização por Sprints e Flexibilidade:**

- **Sprints Semanais:**

O desenvolvimento ocorre em ciclos de 1 semana (sprints), permitindo que entregas parciais do sistema sejam avaliadas, revisadas e refinadas com base nos feedbacks recebidos. Esse ciclo curto possibilita uma resposta rápida a eventuais mudanças ou ajustes necessários, mantendo o projeto alinhado às expectativas e necessidades do produto.

- **Atribuição de Tarefas:**

Cada sprint tem objetivos bem definidos, com tarefas atribuídas a cada membro da equipe de acordo com suas competências. Isso torna o trabalho mais organizado, permitindo que os desenvolvedores se concentrem em partes específicas da aplicação, seja na elaboração de componentes da camada de apresentação, na definição da lógica de negócios ou na implementação dos mecanismos de persistência.

- **Integração com a Estrutura Técnica do EcoNet:**

- **Arquitetura em Camadas:**

Embora o SCRUM regule o processo, a implementação do EcoNet está estruturada em três camadas principais:

- **Camada de Apresentação:**

Responsável pela interface e pela interação com o usuário, garantindo que todas as entradas e saídas sejam devidamente tratadas.

- **Camada de Negócio:**

Focaliza a implementação das regras, validações e operações, como o processamento dos dados e fluxos internos do sistema.

- **Camada de Persistência:**

Cuida do armazenamento, recuperação e gerenciamento dos dados, interagindo com o banco de dados (MySQL).

Essa divisão modular facilita a manutenção e os testes independentes de cada componente, o que se alinha perfeitamente com a filosofia de entregas incrementais do SCRUM.

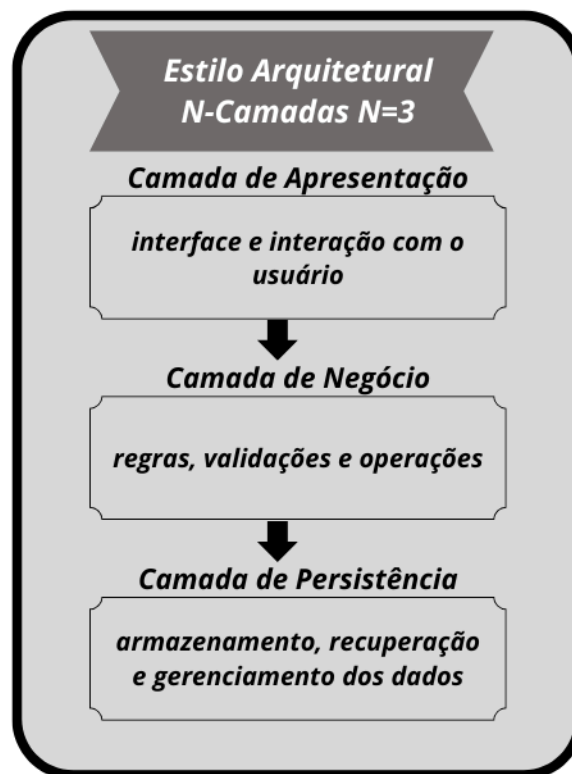
- **Ferramentas e Comunicação:**

- **Colaboração e Feedback Contínuo:**

Utilizamos ferramentas como o Microsoft Teams para reuniões diárias e revisão semanal do progresso (dailys, planning, review e retrospective). Isso possibilita um fluxo constante de feedback, essencial para que cada sprint seja ajustado de acordo com os desafios encontrados, assegurando a consistência e a evolução técnica do sistema.

### **Representação Visual da Arquitetura:**

Figura 1: Diagrama esquemático da representação arquitetural.



Fonte: De autoria própria.

Foi elaborado um diagrama arquitetural que ilustra a divisão em camadas, os fluxos de dados e a comunicação entre os componentes (Figura 1). Portanto, nosso modelo arquitetural pode ser resumido como:

### **"Uma arquitetura em 3 camadas (apresentação, negócio e persistência) integrada a um processo de desenvolvimento ágil via SCRUM".**

Isso permite que cada sprint contribua para a evolução do sistema de forma incremental, com entregas parciais (e testáveis) que estão alinhadas à organização modular proposta, resultando em um sistema fácil de manter e adaptável às mudanças de requisitos.

Em resumo, a combinação do processo ágil proporcionado pelo SCRUM com uma arquitetura técnica em três camadas permite que o EcoNet se desenvolva de maneira organizada, flexível e adaptável. Cada sprint contribui incrementalmente para o produto final, refletindo ajustes constantes no design e na implementação, enquanto a divisão modular garante que os diversos componentes do sistema possam ser desenvolvidos, testados e mantidos com maior clareza e eficiência.

### **3.4 Metas e restrições arquiteturais**

- Foram propostas algumas metas que visam assegurar que a solução atenda às expectativas de qualidade e confiabilidade desde sua implantação até futuras atualizações.
  - O sistema deverá ser capaz de suportar pelo menos 10 acessos simultâneos de usuários, sem apresentar lentidão ou falhas.
  - O sistema deve ter vida útil prolongada, sendo capaz de evoluir tecnologicamente.
  - O site precisa ser responsivo, funcionando adequadamente em navegadores como, Google Chrome e Microsoft Edge.
  - O sistema deve funcionar em rede local.
  - O sistema deve ter criptografia de senhas.

### **3.5 Visão de Casos de uso (escopo do produto)**

- Os casos de uso do sistema **EcoNet** estão centrados na promoção de uma economia circular eficiente e segura entre empresas. O sistema possui diferentes perfis de usuários, cada um com permissões específicas, garantindo segurança e organização no uso da plataforma (Figura 2).

O **Administrador** é responsável pela gestão técnica e completa da aplicação, podendo acessar logs, editar empresas e usuários, além de configurar o sistema. Já o **Dono de Empresa** tem

autoridade legal para editar os dados da empresa, iniciar contatos, assinar contratos e distribuir cargos. O **Vice-Dono** possui as mesmas permissões, exceto a criação/distribuição de cargos.

O **Gestor de Contratos** foca na formalização de acordos, sendo responsável por iniciar contatos e gerenciar os contratos com outras empresas. O **Comunicador** atua no diálogo entre empresas, sem poderes legais, enquanto o **Visitante** apenas visualiza empresas e realiza buscas, sem possibilidade de interação contratual ou de contato.

Cada tipo de usuário tem acesso personalizado a funcionalidades como login, cadastro, busca por empresas, criação de contratos e chats, promovendo uma interação segura e colaborativa. O fluxo de uso da plataforma, portanto, reflete um ecossistema digital bem estruturado para facilitar a gestão de resíduos e matérias-primas reutilizáveis (Figura 3).

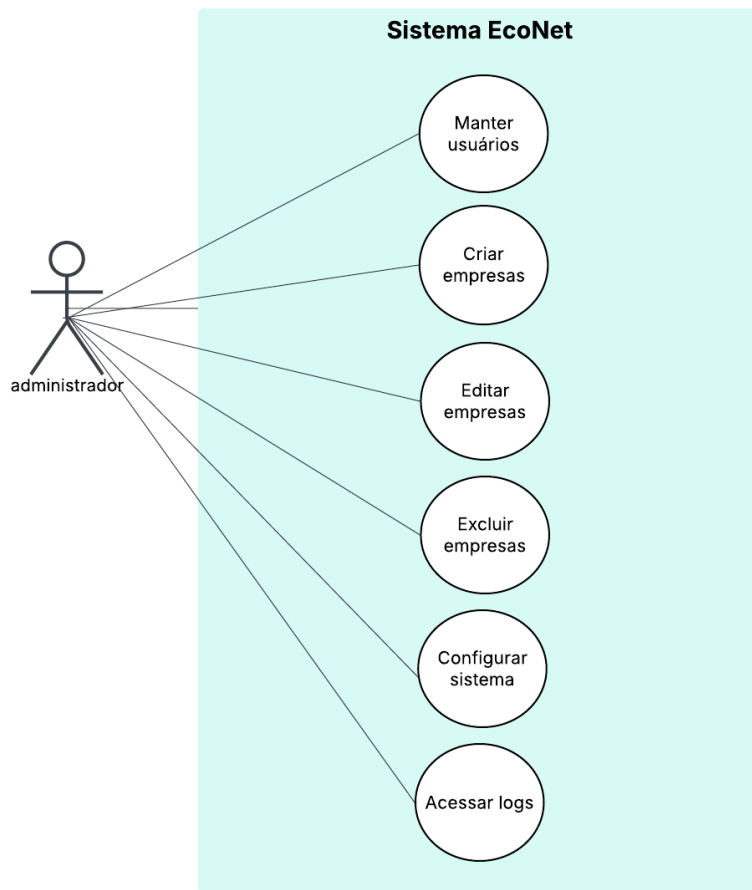
Figura 2: Diagrama de caso de uso para os cargos de dono de empresa, vice-dono de empresa, gestor, comunicador e usuário comum.



Fonte: De autoria própria.



Figura 3: Diagrama de caso de uso para o cargo de administrador



Fonte: De autoria própria.

### 3.6 Visão lógica

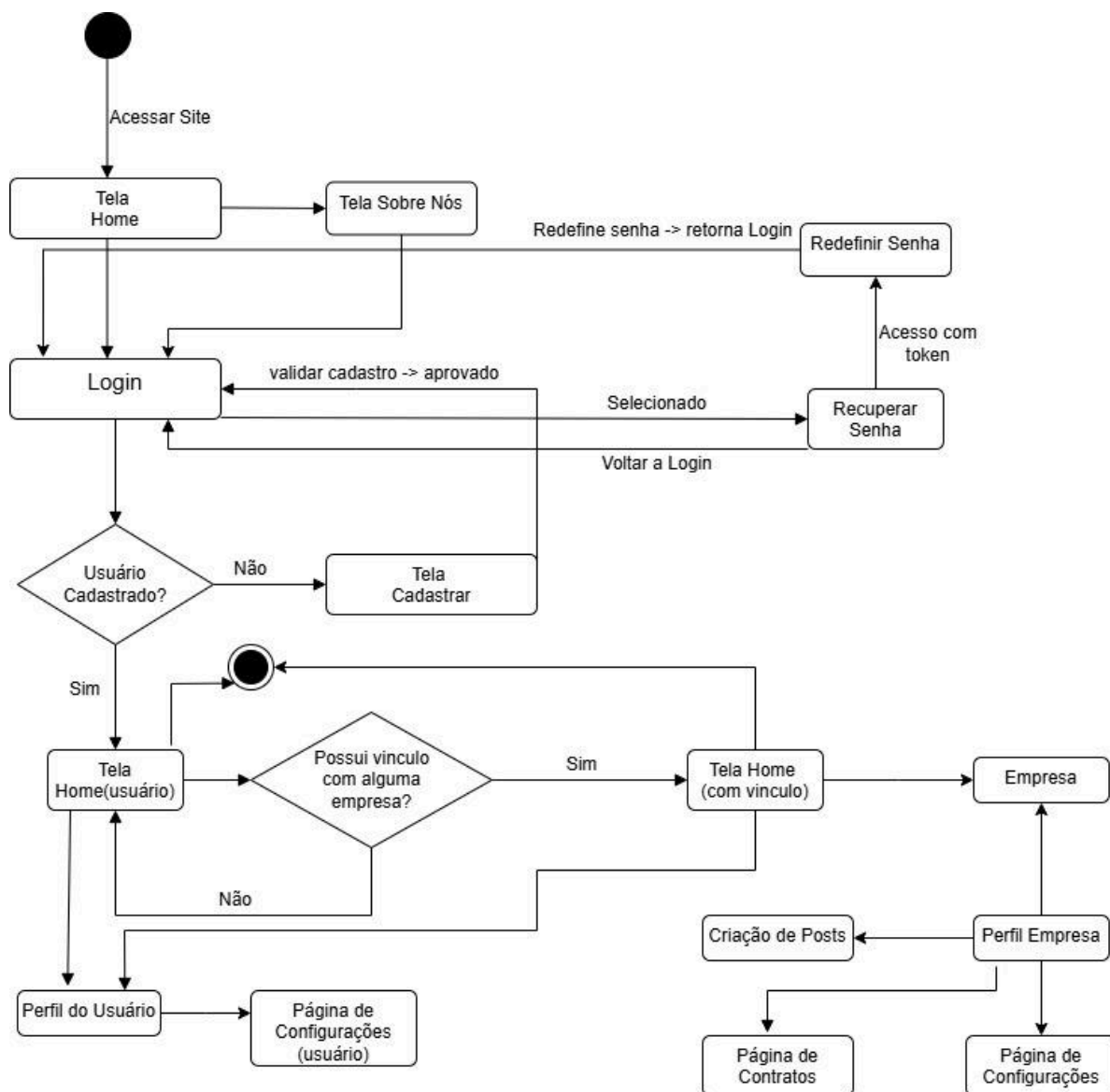
- A Arquitetura em Camadas adotada no sistema EcoNet organiza os módulos com funcionalidades similares em camadas verticais distintas, organizando o sistema em níveis de responsabilidade e promovendo um baixo acoplamento entre seus componentes. Esta abordagem facilita a manutenção, a escalabilidade e a especialização das responsabilidades dentro do sistema. A camada de apresentação lida com a interface do usuário e a interação com o navegador, a camada de negócios é responsável pela lógica da aplicação referente às práticas sustentáveis e interações entre empresas, e a camada de dados/persistência gerencia o armazenamento e a recuperação das informações sobre materiais, usuários e contratos.

- O sistema EcoNet é subdividido nos seguintes módulos principais, que representam suas camadas:
  - **Camada de Apresentação (Frontend):** Responsável por toda a interface com o(a) usuário(a) do EcoNet e pelas lógicas de comunicação com os navegadores. Esta camada constrói e exibe as telas de login, o painel principal com o feed de postagens de materiais e conteúdos, os formulários para criação e edição de anúncios de materiais disponíveis ou necessários, os perfis de empresas e as ferramentas de interação, garantindo uma experiência de usuário intuitiva e responsiva.
  - **Camada de Negócios (Backend/API):** Responsável por executar as operações e fluxos de negócios específicos associados a uma requisição vinda da camada de apresentação. No EcoNet, isso inclui validar dados de cadastro de usuários e empresas, processar a criação e o gerenciamento de postagens (anúncios de materiais), gerenciar o sistema de seguir empresas, facilitar os contatos e a organização de contratos de reaproveitamento, aplicar regras de permissão baseadas nos papéis dos usuários e orquestrar as interações com a camada de persistência.
  - **Camada de Persistência (Dados):** Esta camada é responsável por gerenciar todas as operações de armazenamento, acesso e recuperação dos dados do EcoNet. Ela centraliza a interação com o sistema de gerenciamento de banco de dados, abstraindo os detalhes técnicos sobre onde e como os dados são fisicamente salvos. Nela, são gerenciadas informações essenciais como dados dos usuários (tanto físicos quanto jurídicos), cadastro de empresas, postagens (ofertas e procura de materiais), tags, curtidas, materiais salvos, contratos firmados e configurações do sistema. Essa abordagem garante que os dados sejam organizados, armazenados de forma segura e recuperados de maneira eficiente, facilitando a manutenção e a evolução do sistema e permitindo que as demais camadas interajam com os dados sem se preocupar com os detalhes do armazenamento físico.

- Todas estas camadas no EcoNet são projetadas para serem o mais independentes possíveis, formando uma abstração em torno de suas responsabilidades e do trabalho a ser realizado dentro de uma requisição específica. A comunicação entre as camadas é bem definida, permitindo que cada uma evolua sem impactar diretamente as outras, e podendo ser configurada para ser aberta ou fechada conforme as necessidades de interação e segurança estabelecidas pela equipe de desenvolvimento do EcoNet.

### 3.6.1 Diagrama de estados

Figura 4: Diagrama de estados



Fonte: De autoria própria.

## Descrição Geral do Diagrama de Estados

- O diagrama representa o fluxo de navegação de um sistema web desde o acesso inicial até as diferentes telas conforme o tipo de usuário e seu vínculo com empresas (Figura 4).

### 1. Acesso Inicial:

- O usuário acessa o site e é direcionado para a **Tela Home**.
- Da Home, pode ir para a **Tela Sobre Nós**, **Login**, ou recuperar a senha.

### 2. Login e Cadastro:

- O usuário tenta fazer login.
- Se não for cadastrado, é direcionado para a **Tela Cadastrar**.
- Após o cadastro e validação, retorna ao login.
- Caso esqueça a senha, pode usar **Recuperar Senha**, acessando por token e chegando até **Redefinir Senha** (e retorna ao login).

### 3. Pós-Login:

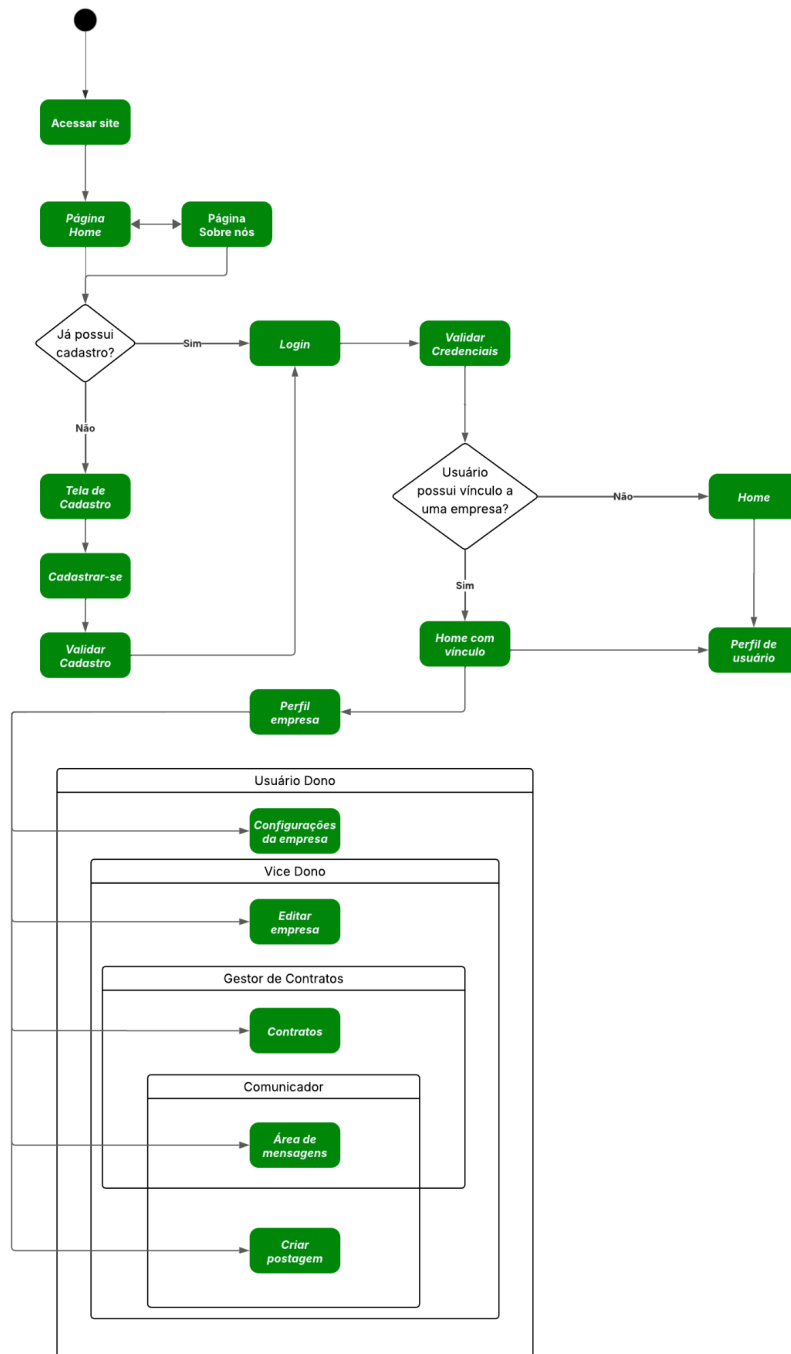
- Se o login for bem-sucedido, verifica-se se o usuário possui vínculo com alguma empresa.
- **Se sim:** é direcionado à **Tela Home (com vínculo)**.
- **Se não:** vai para a **Tela Home (usuário)**.

### 4. Funcionalidades pós-login:

- Usuários sem vínculo podem acessar:
  - **Perfil do Usuário**
  - **Página de Configurações (usuário)**
- Usuários com vínculo podem acessar:
  - **Empresa**
  - **Perfil Empresa**
  - **Página de Configurações**
  - **Página de Contratos**
  - **Criação de Posts**

### 3.6.2 Diagrama de atividades

Figura 5: Diagrama de atividades



Fonte: De autoria própria.

O diagrama (Figura 5) de atividade apresentado mapeia de forma clara e dinâmica o fluxo de interações de um usuário dentro do sistema web, desde o acesso inicial até a execução de funcionalidades específicas pós-login.

- **Fluxo de Navegação Inicial:**

- **Acesso ao Site:**

- O fluxo inicia com o usuário acessando o site, o que resulta na apresentação da **Página Home**. A partir desta tela inicial, o usuário pode optar por navegar para a **Página Sobre Nós** para conhecer mais informações sobre o sistema.

- **Processo de Autenticação e Cadastro:**

- **Verificação de Cadastro:**

- O sistema verifica se o usuário já possui cadastro.

- **Usuário sem cadastro:**

O usuário é direcionado para a **Tela de Cadastro**, onde pode realizar a criação de uma conta. Durante este processo, é realizada a validação dos dados inseridos e verificado se o usuário deseja estabelecer um vínculo com uma empresa.

- **Usuário com cadastro:**

O fluxo avança para o **login**, onde as credenciais fornecidas são validadas.

- **Pós-Authenticação: Home com ou sem Vínculo:**

Após a validação das credenciais, o diagrama destaca uma nova decisão baseada no vínculo do usuário:

- **Usuário com vínculo empresarial:**

- Caso o usuário esteja vinculado a uma empresa, ele é direcionado para a **Home com vínculo**. Nesta interface, além de acessar seu perfil pessoal, o usuário pode interagir com recursos específicos do perfil empresarial.

- **Usuário sem vínculo:**

- Se não houver vínculo, o usuário permanece na interface padrão, a **Home**.

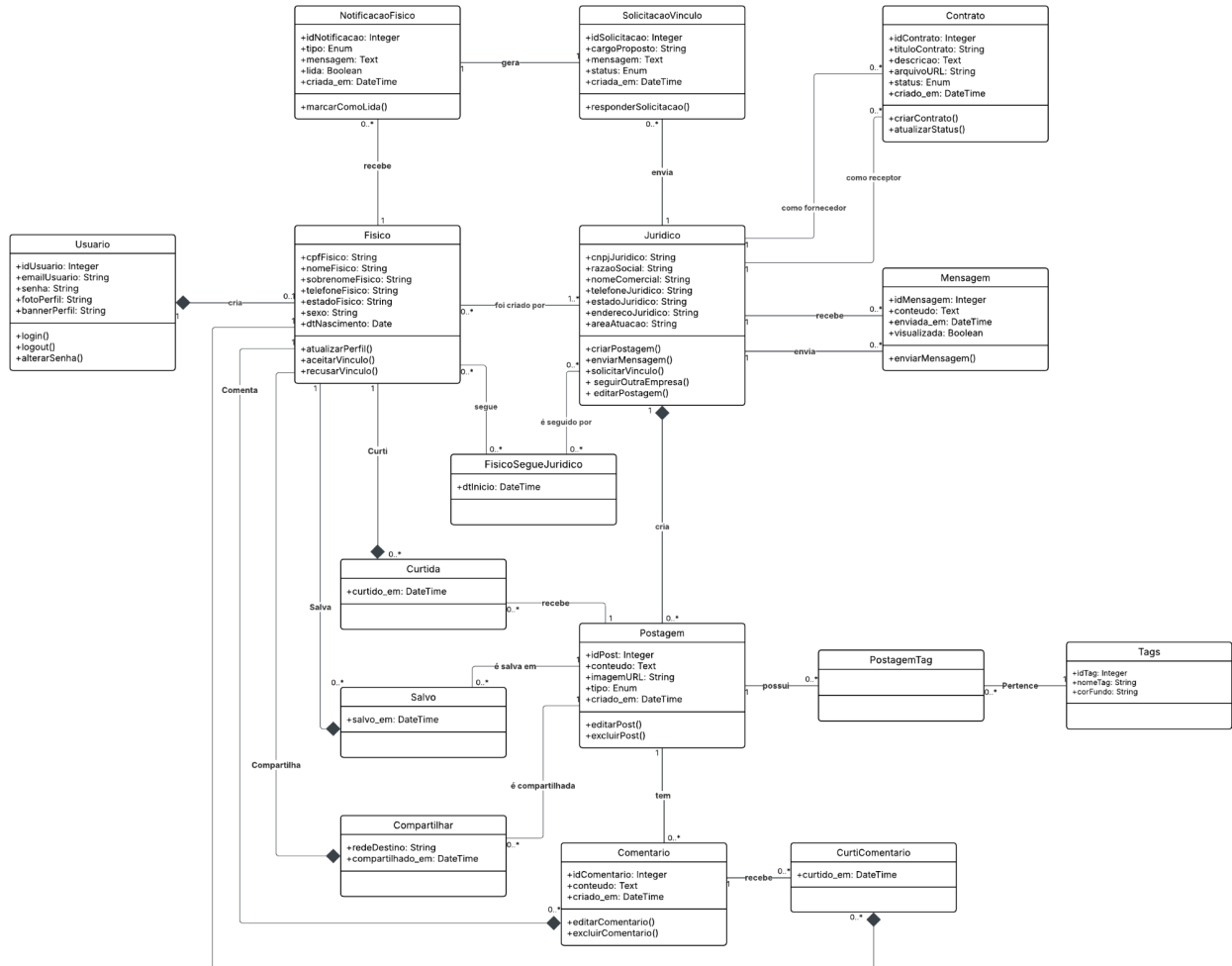
- **Funcionalidades Específicas do Perfil Empresarial:**

Dentro da **Home com vínculo**, o fluxo detalha as funcionalidades disponíveis conforme o tipo de perfil empresarial:

- **Usuário Dono:**  
Possui acesso às **Configurações da empresa** para gerenciar e definir parâmetros essenciais.
- **Vice Dono:**  
Tem a capacidade de **Editar empresa**, permitindo atualizações e ajustes nas informações corporativas.
- **Gestor de Contratos:**  
Responsável pelo **Gerenciamento de Contratos**, podendo acessar e controlar os acordos firmados.
- **Comunicador:**  
Acessa a **Área de mensagens** e tem a permissão para **Criar postagem**, facilitando a comunicação interna e externa.

### 3.6.3 Diagrama de classes

Figura 6: Diagrama de classes



Fonte: De autoria própria.

O diagrama ilustrado (Figura 6) apresenta diversas classes fundamentais organizadas de forma a refletir a complexidade e a modularização do sistema. Entre as classes principais, destacam-se:

- **Usuário:** Central para o sistema, esta classe contém atributos como `idUsuario`, `emailUsuario`, `senhaUsuario`, `fotoPerfil` e `bannerPerfil`, e métodos que suportam funções de autenticação e gerenciamento do perfil (por exemplo, `login()`, `logou()` e `alterarSenha()`).



- **Físico:** Destinada a representar usuários com perfil individual, esta classe inclui atributos como `cpfFísico`, `nomeFísico`, `sobrenomeFísico`, `telefoneFísico`, `emailFísico`, `sexoFísico` e `dtNascimento`. Seus métodos, como `utilizarPerfil()`, `aceitarVinculo()` e `recusarVinculo()`, demonstram interações típicas de redes sociais ou sistemas colaborativos.
- **Jurídico:** Voltada para contas empresariais ou institucionais, esta classe possui atributos específicos como `cnpjJurídico`, `razaoSocial`, `nomeComercial`, `telefoneJurídico`, `emailJurídico`, `sexoJurídico`, `dtNascimento` e `areaAtuacao`. Os métodos `criarPostagem()`, `enviarMensagem()`, `solicitarVinculo()`, `seguirOutroEmpresa()` e `odiarPostagem()` ressaltam as funcionalidades exclusivas deste tipo de perfil.
- **Postagem, Comentário e Tags:** Estas classes mapeiam os elementos de conteúdo e interação. A classe `Postagem` se relaciona com `Comentario` (permitindo avaliações e discussões) e com `Tags` (por meio da classe associativa `PostagemTag`), o que organiza e categoriza o conteúdo do sistema.
- Outras classes, como `NotificacaoFísico`, `SolicitacaoVinculo`, `Contrato`, `Mensagem`, `FísicoSegueJurídico`, `Curtida`, `Salvo`, `Compartilhar` e `CurtiComentario`, ampliam a funcionalidade do sistema, cobrindo aspectos como notificações, vínculos entre usuários e reações diversas a conteúdos compartilhados.

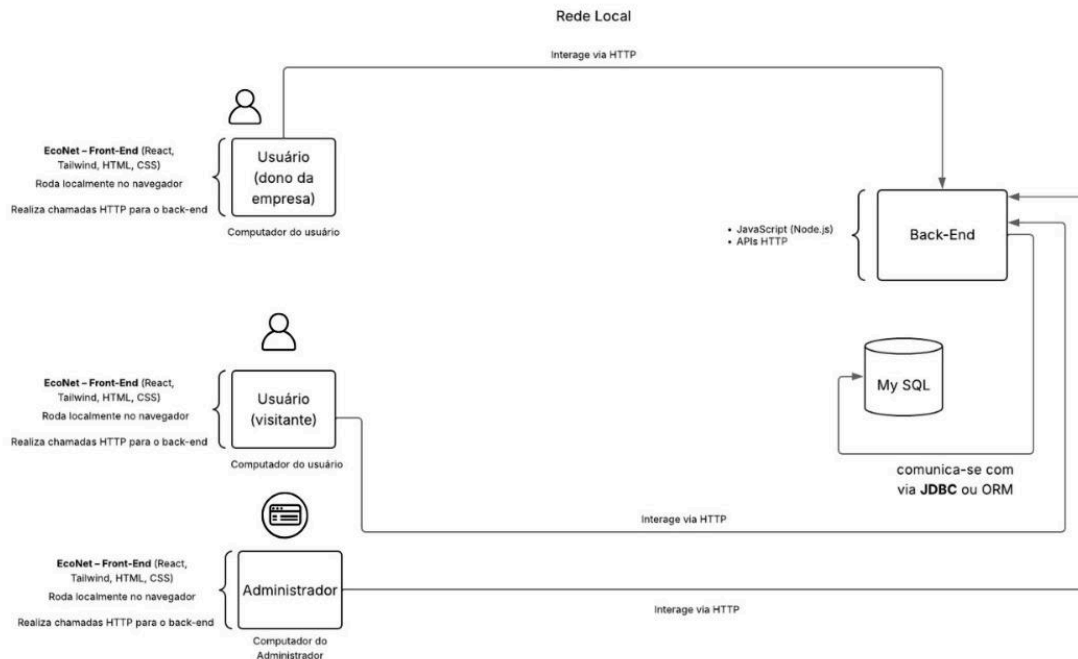
## Relações e Interações

As conexões entre as classes, representadas por linhas no diagrama, indicam como os componentes do sistema interagem. Por exemplo, a associação entre a classe `Usuario` e `Físico` demonstra a especialização ou extensão de comportamentos, enquanto a ligação entre `Postagem` e `Salvo` evidencia a funcionalidade de armazenamento ou bookmark de conteúdo. Essa rede de relacionamentos é fundamental para:

- Visualizar a dependência entre os módulos do sistema.
- Identificar pontos de integração e interface entre componentes.
- Assegurar que a implementação respeite as diretrizes da orientação a objetos, promovendo baixo acoplamento e alta coesão.

### 3.7 Visão de Implementação

Figura 7: Desenho esquemático da visão de implementação.



Fonte: De autoria própria.

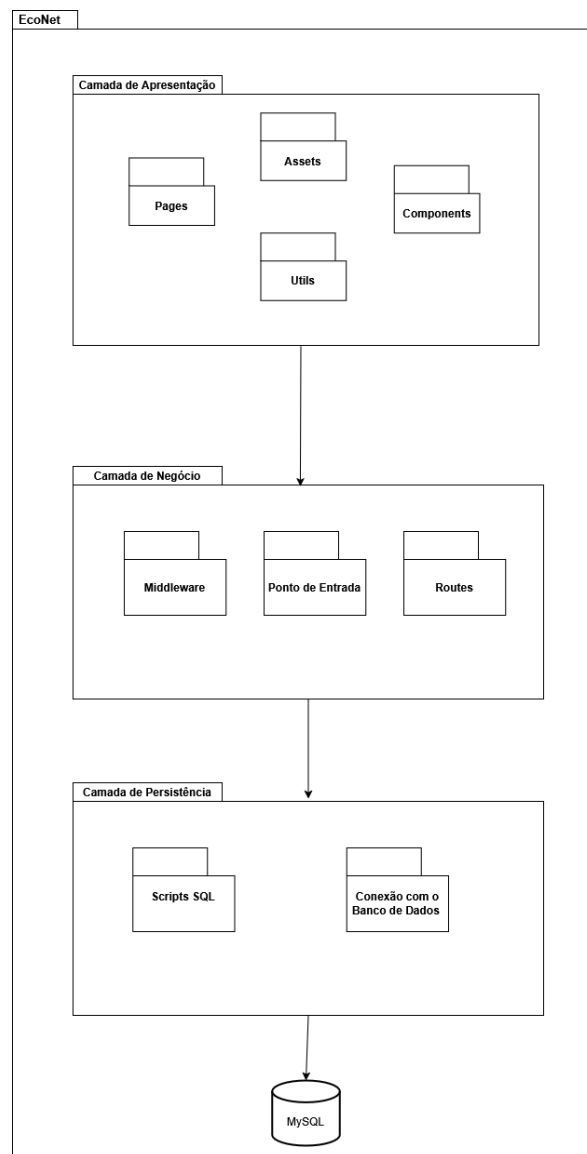
O Desenho esquemático (Figura 7) apresenta um diagrama esquemático da visão de implementação do EcoNet em rede local. Os principais elementos são:

- Três perfis de usuários, cada um representado por um ícone de computador:
  - Dono de Empresa
  - Visitante
  - Administrador
- Interface de usuário (front-end) construída com React, Material-UI e CSS, indicada como camada de apresentação. Essa camada faz requisições HTTP ao back-end.
- Back-end implementado em Node.js, expondo APIs REST para tratar as requisições vindas do front-end.
- Persistência de dados em um banco MySQL, acessado via JDBC ou ORM pelo back-end.

As setas direcionais mostram o fluxo de comunicação HTTP do front-end para as APIs REST e, em seguida, o acesso ao banco de dados. A separação clara de camadas reforça o isolamento de responsabilidades, facilitando a manutenção, escalabilidade e segurança da aplicação.

O EcoNet adota uma arquitetura baseada em três camadas principais: apresentação, negócio e persistência. Essa abordagem favorece a separação de responsabilidades, tornando o sistema mais modular e de fácil manutenção. A seguir, é apresentado um diagrama (Figura 08) de pacotes representando as camadas do projeto e os pacotes que a compõem.

Figura 8: Diagrama de Pacotes do Projeto Econet



Fonte: De autoria própria.

## Camada de Apresentação:

- A camada de apresentação é responsável por fornecer a interface gráfica da aplicação, permitindo a interação entre os usuários e o sistema. Os pacotes desta camada concentram a estrutura de páginas e componentes visuais reutilizáveis, além de arquivos estáticos e utilitários relacionados à interface. A comunicação com as camadas inferiores se dá por meio de requisições HTTP. A seguir são apresentados alguns protótipos de baixa fidelidade (Figura 9), (Figura 10), (Figura 11) e (Figura 12), desenvolvidos para representar as principais interfaces do EcoNet.

Figura 9: Protótipo da Página Home.



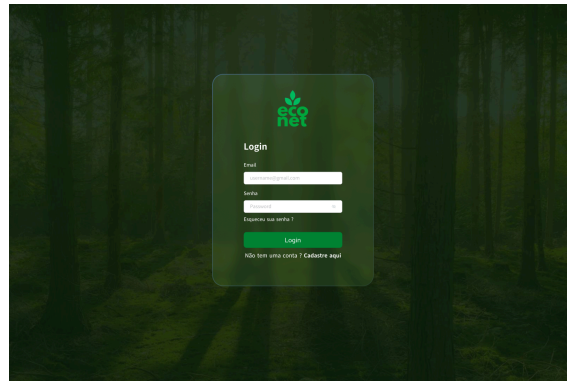
Fonte: De autoria própria.

Figura 10: Continuação do Protótipo da Página Home.



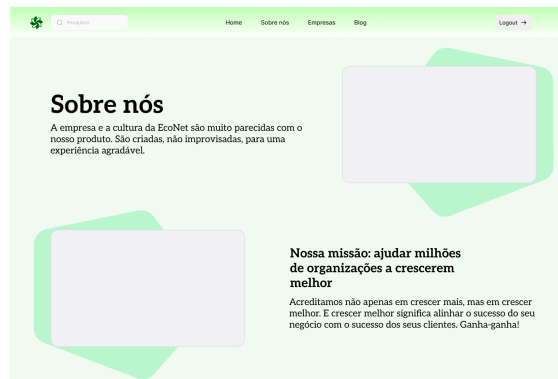
Fonte: De autoria própria.

Figura 11: Protótipo da Página Login.



Fonte: De autoria própria.

Figura 12: Protótipo da Página Sobre Nós.



Fonte: De autoria própria.

### **Camada de Negócio:**

A camada de negócios é o núcleo funcional da aplicação. Os pacotes desta camada são responsáveis por receber as requisições vindas da interface de usuário, processar os dados conforme as regras definidas (como autenticação, validação e autorização) e interagir com a camada de persistência para armazenar ou recuperar dados.

### **Camada de Persistência:**

- A camada de persistência tem como principal função gerenciar a comunicação com o banco de dados da aplicação. Os pacotes desta camada realizam a conexão com o banco, executam instruções SQL e fornecem à camada de negócios os dados necessários para o funcionamento da aplicação. São incluídas aqui as operações de leitura, inserção, atualização e exclusão de dados.

### 3.8 Visão de Implantação

- O software será implantado em uma infraestrutura baseada em contêineres utilizando Docker, o que garante portabilidade, escalabilidade e facilidade de manutenção do ambiente. Essa abordagem permite que o sistema seja executado de forma consistente em diferentes ambientes, facilitando a integração contínua e o deploy automatizado via GitHub. A aplicação será desenvolvida com JavaScript e React, garantindo uma interface responsiva e de alta performance, utilizada com Tailwind CSS para facilitar a construção de layouts customizáveis e eficientes. O backend será conectado a um banco de dados MySQL, escolhido por sua robustez, desempenho em operações relacionais e ampla compatibilidade com ferramentas de gestão. Para a administração interna do sistema, será utilizado o AdminJS, permitindo uma interface moderna, intuitiva e altamente personalizável para o gerenciamento de dados e usuários diretamente a partir do backend. A documentação do projeto será gerada com Docusaurus, que terá um papel essencial na centralização e organização do conhecimento técnico do sistema, facilitando a navegação e a compreensão tanto por desenvolvedores quanto por usuários técnicos. Essa infraestrutura tecnológica garante que o software seja implementado de forma segura, com alta disponibilidade e facilidade de manutenção e evolução.

### 3.9 Restrições adicionais

- O Software será voltado para rede global, onde qualquer usuário ou empresa poderão se conectar e comunicar-se através de um chat disponibilizado pelo produto, sendo necessário apenas as credenciais e senha do usuário para acessar as demais funcionalidades do website.
  - **Confiabilidade:** Deve proporcionar uma segurança promissora, a fim de garantir que apenas usuários logados possam desfrutar das funcionalidades do software.
  - **Interface:** Contará com padronizações em todas as áreas estéticas, ou seja, cores das páginas, fontes, botões, ícones e caixas.
  - **Eficiência:** Deve apresentar um ambiente totalmente fluido e que minimize os atrasos no processamento de respostas.
  - **Portabilidade:** Deve apresentar compatibilidade com mais de um navegador, sendo eles Google Chrome e Microsoft Edge.

## **4.0 BIBLIOGRAFIA**

SOMMERVILLE, Ian. Engenharia de Software.

FURLAN, José D. Modelagem de Dados: A Base para o Sucesso dos Seus Projetos.

MANIFESTO ÁGIL. 2001. Disponível em:

<https://agilemanifesto.org/iso/ptbr/manifesto.html>SOMMERVILLE, Ian.