**ECO DASHBOARD**

**ECO DASHBOARD**

**Arquitetura de Software**

**Versão <1.0>**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 11/05/2025 | 1.0 | Elaboração do documento | Luiz Guilherme |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Sumário

[1.](#_heading=h.5oouda3r8lqv) Introdução 4

[1.1](#_heading=h.ar6pxklz5np1) Finalidade 4

[1.2](#_heading=h.wmqefyfmy8ri) Escopo 4

[1.3](#_heading=h.q87kk2in6td8) Definições, Acrônimos e Abreviações 4

[1.4](#_heading=h.27mw26nfmtta) Visão Geral 4

[2.](#_heading=h.9qi0gcwx4fck) Representação Arquitetural 4

[3.](#_heading=h.zbguxpirpm52) Metas e Restrições da Arquitetura 5

[4.](#_heading=h.lr33qdztkdm) Visão de Casos de Uso 5

[5.](#_heading=h.btm89jtuqa8k) Visão Lógica 5

[5.1](#_heading=h.21awggrgm7wi) Visão Geral 5

[5.2](#_heading=h.681rlwnhc4t8) Pacotes de Design Significativos do Ponto de Vista da Arquitetura 7

[6.](#_heading=h.dx7elpe4johm) Visão de Implantação 7

[7.](#_heading=h.efzyk7w7vhrx) Visão da Implementação 7

1. Tamanho e Desempenho ....................................................................................................... 9
2. Qualidade .............................................................................................................................. 9

# Introdução

O presente documento tem como objetivo descrever a arquitetura do sistema **ECO DASHBOARD**, uma aplicação web voltada para o monitoramento da qualidade da água e do ar. Este sistema tem como finalidade proporcionar uma plataforma acessível e eficiente para a análise de dados ambientais, facilitando a visualização de informações relevantes e auxiliando na tomada de decisões por parte da Secretaria de Meio Ambiente.

## Finalidade

Este documento oferece uma visão arquitetural geral do sistema, utilizando diferentes perspectivas para representar os diversos componentes e interações da aplicação. O objetivo é registrar e comunicar as decisões arquiteturais mais relevantes tomadas durante o desenvolvimento, garantindo que todos os envolvidos compreendam a estrutura e o funcionamento do sistema.

## Escopo

O sistema **ECO DASHBOARD** permite o upload de arquivos CSV contendo dados ambientais, como pH, turbidez e níveis de CO₂, armazenando essas informações em um banco de dados relacional (SQLite) e oferecendo ferramentas para visualização e análise por meio de gráficos e tabelas. Este documento é direcionado a desenvolvedores, arquitetos de software, servindo como base para implementação, manutenção e evolução do sistema.

## Descrição do Problema

A Secretaria de Meio Ambiente enfrenta dificuldades na análise de dados ambientais, que atualmente são recebidos em planilhas e analisados manualmente. A ausência de ferramentas visuais e automatizadas compromete a interpretação dos dados e dificulta a identificação de padrões e anomalias. O sistema **ECO DASHBOARD** foi concebido para resolver esse problema, oferecendo uma solução digital que automatiza o processamento, visualização e interpretação de dados ambientais.

|  |  |
| --- | --- |
| **Problema** | Dificuldade na visualização e análise dos dados de qualidade da água e do ar, dados coletados manualmente ou recebidos em planilhas (CSV) desestruturadas |
| **Afeta** | Secretaria de Meio Ambiente, Equipe técnica e analistas ambientais, Gestores e responsáveis pela fiscalização |
| **Impacto** | Tomada de decisões ambientais mais lenta e menos precisa, aumento do tempo de análise e maior chance de erro na interpretação |
| **Solução** | Desenvolvimento de uma aplicação web com visualização gráfica com Matplotlib para análise dos dados ambientais, upload de arquivos CSV e armazenamento estruturado em banco de dados (SQLite) |

## Definições, Acrônimos e Abreviações

**MVC**: Padrão de arquitetura de software que divide a aplicação em três componentes principais:

**Model**: Responsável pelas regras de negócio e acesso ao banco de dados (como o SQLite).

**View**: Responsável pela interface do usuário, onde as informações e gráficos são exibidos (usando HTML/CSS).

**Controller**: Gerencia as interações entre o model e a view, controlando as ações do usuário (feito pelo Django no projeto).

## Visão Geral

Este documento apresenta as visões arquiteturais do ECO DASHBOARD, abordando seu comportamento nos principais processos, formas de implantação e implementação, além das restrições de desempenho e qualidade do sistema.

**1.5.1 Tabela de Funcionalidades**

|  |
| --- |
| **Funcionalidades** |
| * Permite autenticação de usuários com nome de usuário e senha. * Permite o envio de planilhas CSV com dados ambientais. * Salva os dados recebidos no banco de dados SQLite. * Exibe os dados enviados em formato tabular para facilitar a leitura. * Permite filtrar os dados por período e local de coleta. * Permite salvar os gráficos gerados como imagens PNG. |

## 2. Levantamento e Análise de Requisitos

### 2.1 Entendimento do Problema

* Técnicos ambientais precisam analisar dados de diferentes regiões e períodos, mas não possuem uma ferramenta visual para isso.
* Falta de histórico visual dificulta o acompanhamento da evolução da qualidade ambiental ao longo do tempo.

### 2.2 Requisitos Funcionais e Não-Funcionais

* **Funcionais**: Login de usuários, upload de arquivos CSV, armazenamento em banco de dados, visualização em tabela, filtros por data e local, geração de gráficos e exportação em PNG
* **Não-Funcionais**: desempenho com resposta em até 5 segundos, autenticação para segurança, interface simples com HTML/CSS nativo e compatibilidade com arquivos CSV em UTF-8.

# Resumo dos Usuários

|  |  |  |
| --- | --- | --- |
| **Nome** | **Responsabilidade** | **Descrição** |
| Usuários | Gerar tabelas e excluir apenas os arquivos que enviaram | Podem enviar arquivos CSV, gerar gráficos e excluir somente os dados que eles próprios enviaram. |
| Moderadores | Gerar tabelas e excluir quaisquer arquivos postados | Têm permissão para gerar gráficos e excluir qualquer arquivo enviado por qualquer usuário. |
| Administradores | Todas as permissões anteriores + gerenciar usuários | Além de gerar e excluir arquivos, podem adicionar novos usuários e criar grupos com diferentes papéis. |

# Necessidades do Cliente

|  |  |  |  |
| --- | --- | --- | --- |
| **Identificador** | **Necessidade** | **Prioridade** | **Solução Atual** |
| Cliente | Visualizar dados rapidamente | Alta. | Análise manual em planilhas |
| Usuário | Gerar gráficos de tendências | Alta. | Nenhuma ferramenta automatizada |

# Representação Arquitetural

* **Visão de Caso de Uso:** Apresenta as funcionalidades principais do ECO DASHBOARD, como upload de arquivos CSV, geração de gráficos, aplicação de filtros e controle de acesso via login, além dos perfis de usuários (Usuários, Moderadores e Administradores) que interagem com o sistema..
* **Visão Lógica:** Descreve as classes que compõem o sistema, organizadas segundo o padrão MVC do Django, incluindo modelos para os dados ambientais, views para controle das requisições e templates para a interface. Essa organização facilita manutenção e escalabilidade.
* Visão de processos: Mostra como o sistema responde às ações do usuário, como envio de arquivos, aplicação de filtros e geração de gráficos, garantindo que cada etapa seja processada em sequência para manter a integridade dos dados e a fluidez da interface.

* Visão de implantação: Descreve o ambiente de execução do ECO DASHBOARD, que pode ser um servidor local ou na nuvem, com suporte a Python, Django e banco SQLite, garantindo fácil instalação e baixo custo de infraestrutura.
* Visão de implementação**:** Ilustra a distribuição dos processos do sistema, onde o backend Django processa as requisições, manipula os dados e gera os gráficos, enquanto o frontend HTML/CSS exibe as informações ao usuário. O sistema utiliza threads para processar uploads e gerar gráficos de forma assíncrona, melhorando o desempenho.

# Metas e Restrições da Arquitetura

Existem algumas restrições de requisito e de sistema principais que têm uma relação significativa com a arquitetura, sendo elas:

* Utilização do paradigma **Orientado a Objetos** para garantir modularidade, reutilização e organização do código.
* Estrutura MVC;
* Linguagem de programação **Python 3**, compatível com bibliotecas para análise de dados e geração de gráficos;
* Framework *Django*
* O sistema em questão deverá ser multiplataforma; Windows, Linux e MacOS  Bancos de dados *SQLite*.

## 6. Arquitetura do Sistema e Design de Software

### 6.1 Visão Geral da Arquitetura

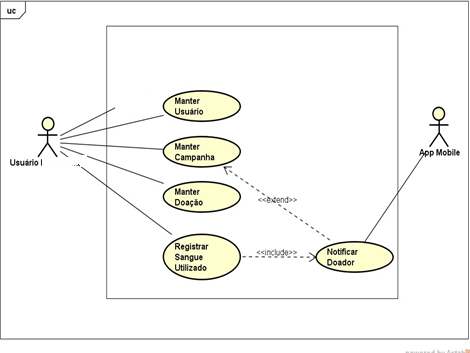
* **Padrão**: Arquitetura monolítica modular, planejada para ser migrada para uma arquitetura de microserviços em versões futuras.
* **Tecnologias**: Python 3.0+, Django (backend), HTML/CSS nativo (frontend), SQLite.

### 6.2 Design de Software (Diagramas)

* **UML de Classes**:
* **Diagrama de Sequência**:
* **Diagrama de Componentes**:
* Modelos Conceitual, Lógico do Banco de Dados

# Visão de Casos de Uso

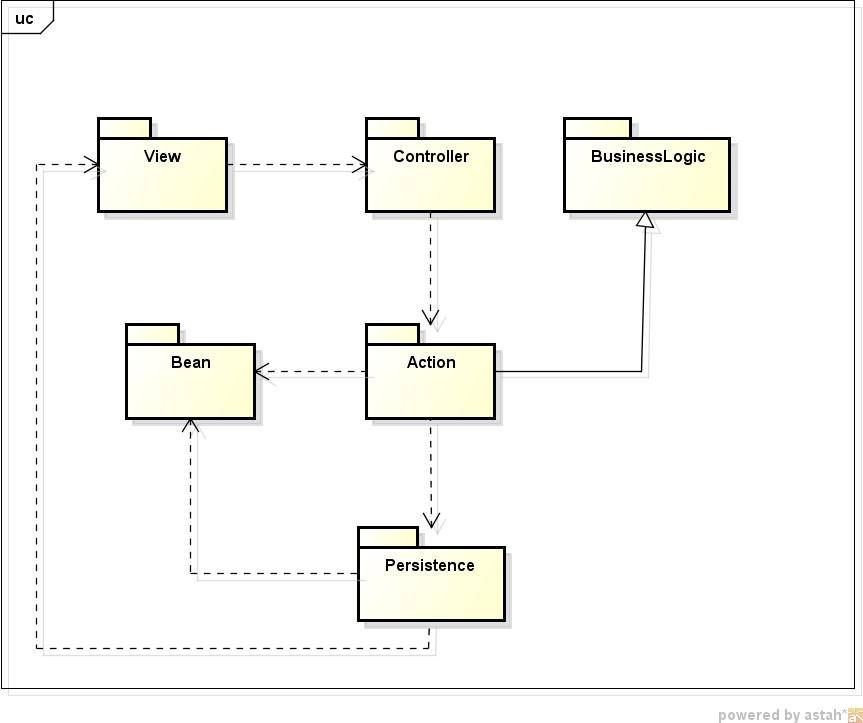
Os casos de uso do sistema xxxxx serão listados abaixo:



# Visão Lógica

## Visão Geral

A visão lógica define a estrutura da arquitetura. Abaixo será especificado o padrão utilizado para o desenvolvimento do sistema, no caso, MVC.



* *View*: Componente que contém as visões do projeto, ou seja, as interfaces, formulários e etc.;
* *Controller*: Componente que recebe as informações e requisições do pacote *view* e os despacha para devida classe de controle, o *controller* se comunica com classes *BusinessLogic* que contém as regras de negócio do sistema;
* *Action - Model*: pacote que recebe as informações e requisições do pacote *controller* e os atribui às respectivas classes do pacote *bean* e do pacote *persistence,* utilizando dados de seus modelos;
* *Bean*: pacote que recebe e armazena as informações referentes a seus respectivos objetos;
* *Persistence*: pacote que recebe as requisições e realiza as operações relacionadas ao Banco de Dados, utilizando dados provindos do pacote *bean*. O pacote *persistence* também pode enviar dados para o pacote *view*.

## Pacotes de Design Significativos do Ponto de Vista da Arquitetura

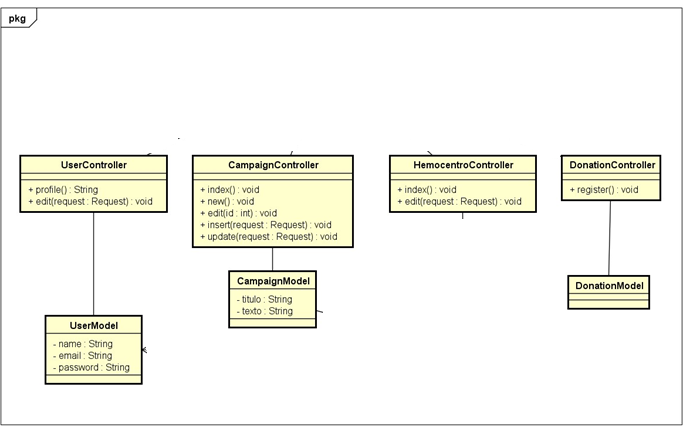


Diagrama de Classe geral do sistema.

# Visão de Implantação

O sistema será implantado quando for validado entre os *stakeholders*.

# Visão da Implementação

O sistema será implementado utilizando conceitos de Programação Orientada a Objetos através do framework *XXXX*, estrutura MVC, , *Web Container JBOSS,* Linguagem de Programação Java e Bancos de Dados *xxxx*.

1. **Tamanho e Desempenho**

A aplicação deve rodar através de um site próprio na versão web para os xxxx e por meio de dispositivos móveis na versão mobile que será para o uso do usuário xxxxx.

1. **Qualidade**

O padrão de arquitetura adotado no projeto tem como finalidade garantir uma melhor organização do código-fonte, o que auxilia na manutenibilidade do software, bem como a portabilidade do mesmo.