GREENLEENS

GREENLEENS

Arquitetura de Software

Versão <1.4>

Histórico da Revisão

Data	Versão	Descrição	Autor
29/05/2025	1.0	Elaboração do documento	Graziella Doche
05/06/2025	1.1	Parte Visual do Projeto	Thiago Alcino
06/06/2025	1.2	Inclusão do modelo 4+1 e definição do escopo	Thiago Ribeiro
07/06/2025	1.3	Adição dos casos de uso, visão lógica e visão de implantação	Graziella Doche
12/06/2025	1.4	Versão final validada com todos os requisitos completos	Samuel Monteiro

1. Introdução

O presente documento tem como objetivo descrever o documento de arquitetura do projeto GreenLeens, que utiliza o modelo arquitetural "4+1" Views, proposto por Philippe Kruchten (1995), para descrever a arquitetura do sistema GreenLeens. O modelo organiza a arquitetura em cinco visões complementares: lógica, processos, desenvolvimento, física e cenários.

Figura 1 – Arquitetura 4+1

O GreenLeens é um aplicativo sustentável voltado inicialmente à moda consciente. Permite que usuários visualizem diversos dados de sustentabilidade de produtos, incluindo a pegada de carbono, condições de trabalho na cadeia de produção, consumo de água e energia e entre outros, comparem alternativas sustentáveis, ganhem recompensas ecológicas (GreenCoins) e acompanhem relatórios de impacto ambiental. Além disso, permite que marcas parceiras gerem QR Codes com etiquetas digitais para impressão e fixação em seus produtos.

Figura 2 – Logo GreenLeens

1.1 Finalidade

Este documento oferece uma visão arquitetural geral do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do mesmo. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas no decorrer do processo de desenvolvimento do GreenLeens.

1.2 Escopo

Este documento tem como objetivo auxiliar todos os envolvidos no projeto a compreender os aspectos arquiteturais essenciais para o desenvolvimento de uma solução que atenda às necessidades dos usuários finais. Ele também serve como guia de referência para novos membros da equipe durante sua integração.

O escopo inicial do GreenLeens abrange a criação e operação de uma etiqueta digital inteligente para produtos de vestuário, com foco nas seguintes funcionalidades principais:

- ➔ Visualização de dados multifacetados de sustentabilidade dos produtos, incluindo:
  - Pegada de carbono;
  - Condições de trabalho na produção;
  - Consumo de água e energia.
- ➔ Comparação entre produtos, com base em indicadores de impacto ambiental.
- ➔ Sistema de gerenciamento de recompensas (GreenCoins), com acúmulo e resgate por boas práticas de consumo.
- ➔ Geração e acompanhamento de relatórios de impacto ambiental personalizados para os usuários.
- ➔ Geração automática de QR Codes contendo etiquetas digitais interativas para marcas parceiras aplicarem em suas peças.

[1]*Nota:* O escopo poderá ser expandido futuramente para incluir integrações com lojas físicas, certificadoras de sustentabilidade e novas categorias de produtos, além de funcionalidades analíticas para o mercado B2B.

1

1.3 Definições, Acrônimos e Abreviações

Termo	Descrição
MVC	Padrão de arquitetura de software onde M significa modelo sendo responsável pela parte de regras de negócio, V a visualização responsável pela parte de interfaces e C a parte de controle dos dados.
ESG	*Environmental, Social and Governance* (meio ambiente, social e governança).
IA	Inteligência artificial: usada para recomendações personalizadas no app.
API	*Application Programming Interface* (interface de comunicação entre sistemas).
QR CODE	*Quick response code*: código bidimensional escaneável para acesso rápido a dados.
GREENCOINS	Moeda virtual usada no app para recompensar ações sustentáveis.
ETIQUETA DIGITAL	Interface visual contendo informações de sustentabilidade vinculadas a um produto.
B2B	Business to business: relação entre empresas, como a GreenLeens e marcas parceiras.
PEGADA DE	Medida das emissões de gases de efeito estufa associadas à

CARBONO	produção ou uso de um produto.
UX/UI	User experience/user interface: experiência e interface do usuário.
PRODUTO SUSTENTÁVEL	Produto com menor impacto ambiental, socialmente responsável e transparente.
CI/CD	Continuous integration/continuous delivery – integração e entrega contínua de software.

1.4 Visão Geral

São apresentados ainda neste documento diferentes visões arquiteturais de como o sistema deve se comportar em diferentes processos, como deve ser implantado é implementado e restrições de desempenho e qualidade.

2. Representação Arquitetural

Utilizando o modelo "4+1" de vistas arquiteturais de Philippe Kruchten, o GreenLeens será descrito através das seguintes vistas, cada uma abordando um conjunto específico de preocupações:

1. **Visão de cenários (*scenarios view*):** Apresenta as funcionalidades arquiteturais importantes e os usuários do sistema. Ilustra como as outras quatro vistas se interligam e como o sistema interage com seus usuários em situações reais. Esta visão é redundante com as outras (por isso o "+1"), mas serve como um driver para descobrir elementos arquiteturais e para validação do projeto.

2. **Visão lógica (*logical view*):** Descreve as classes e sua organização e apresenta o padrão de arquitetura que deverá ser utilizado para o desenvolvimento do sistema. Foca na estrutura funcional e modular, suportando os requisitos funcionais.
3. **Visão de processos (*process view*):** Mostra o padrão de comportamento do sistema diante de diferentes ações do usuário. Considera requisitos não-funcionais como desempenho e disponibilidade, e questões de concorrência e distribuição.
4. **Visão de desenvolvimento (*development view*):** Descreve a organização estática do software em seu ambiente de desenvolvimento. Foca na estrutura de implementação e manutenção do sistema.
5. **Visão Física (*physical view*):** Descreve a estrutura do ambiente onde o software será instalado. Leva em conta principalmente os requisitos não-funcionais como disponibilidade, confiabilidade, desempenho e escalabilidade.

3. 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 o desenvolvimento;
- Estrutura MVC;
- Linguagem de programação Java;
- O sistema em questão deverá ser multiplataforma;
- Frameworks: Spring Boot (backend), Flutter (aplicativo móvel) e React (aplicação web);
- O sistema em questão deverá ser multiplataforma (Web para administradores/marcas parceiras e Mobile para usuários finais);
- Bancos de dados: PostgreSQL (relacional para dados estruturados) e/ou MongoDB (NoSQL para dados flexíveis ou de grande volume);
- Implantação em ambiente de nuvem (Cloud-based) para escalabilidade e alta disponibilidade.

4. Visão de Casos de Uso

Os casos de uso do sistema GreenLeens serão listados abaixo e seus respectivos atores:

Atores do sistema:

- **Usuário consumidor**: pessoa usuária do aplicativo que busca consumir moda de forma mais consciente.
- **Marca parceira**: empresa do setor de moda que deseja cadastrar seus produtos e fornecer dados ambientais.
- **Administrador**: responsável por validar dados, moderar cadastros e acompanhar indicadores de uso.

ID	NOME DO CASO DE USO	ATOR PRINCIPAL	DESCRIÇÃO
UC01	Visualizar pegada de carbono	Usuário	Exibe os impactos ambientais de uma peça escaneada ou buscada.
UC02	Comparar produtos	Usuário	Permite comparar peças com base em critérios como CO_2, água, energia e etc.
UC03	Ganhar GreenCoins	Usuário	Ao fazer boas escolhas, o usuário é recompensado com moedas verdes.
UC04	Resgatar recompensas	Usuário	Troca GreenCoins por cupons, brindes ou produtos sustentáveis.
UC05	Cadastrar produto sustentável	Marca parceira	Insere dados técnicos e ambientais de um novo item no app.
UC06	Gerar QR code da etiqueta digital	Marca parceira	Gera o QR code para impressão e fixação na roupa física.
UC07	Validar produto e dados	Administrador	Aprova ou rejeita cadastros de produtos e informações sensíveis.

UC08	Reportar produto não encontrado	Usuário	Permite registrar um item que não tem etiqueta ou cadastro no sistema.

5. Visão Lógica

5.1 Visão Geral

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

A camada Model ontém as classes de domínio, como Produto, Usuário, EtiquetaDigital, GreenCoins e seus respectivos relacionamentos. Essa camada também abriga a lógica de negócio, como o cálculo da pegada de carbono, as regras de pontuação com GreenCoins e a geração de relatórios.

A View é responsável pelas interfaces gráficas em Flutter (mobile) e React (web), exibindo os dados para o usuário e recebendo suas interações.

A camada Controller intermedia a comunicação entre a interface e os modelos, processando as ações dos usuários e coordenando as respostas do sistema.

5.2 Principais módulos lógicos

com.greenleens.user	Gerencia dados do usuário, preferências, conquistas e perfil de consumo.
com.greenleens.product	Responsável pelas informações dos produtos, comparação de impacto e exibição de detalhes.
com.greenleens.rewards	Controla o acúmulo e o resgate de GreenCoins.
com.greenleens.qrcode	Gera os QR Codes para etiquetas digitais e os vincula aos produtos.

com.greenleens.recommendation	Motor de recomendação com base em IA, sugerindo produtos sustentáveis.
com.greenleens.admin	Interface de moderação e validação de informações sensíveis.

6. Visão de Implantação

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

6.3 Diagrama de Implantação

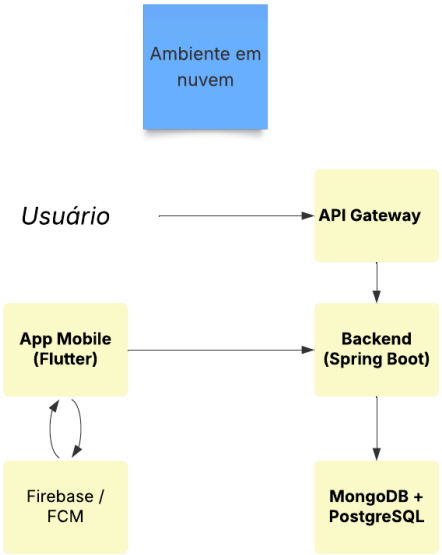

Figura 3 – Mapa conceitual - Criado com o Lucidchart

7. Visão da Implementação

O sistema GreenLeens será implementado com base em princípios de Programação Orientada a Objetos e com adoção do padrão de arquitetura MVC (Model-View-Controller), utilizando as seguintes tecnologias:

- **Backend**: Java 17 com framework Spring Boot, estruturado em camadas (controller, service, repository e domain).

- **Frontend Web**: React.js, utilizando componentes reutilizáveis e comunicação via REST APIs.

- **Aplicativo Mobile**: Flutter, com foco em multiplataforma (iOS e Android), consumindo as mesmas APIs do backend.

- **Bancos de Dados**:

    - **PostgreSQL** para dados estruturados (usuários, produtos, marcas).

    - **MongoDB** para dados flexíveis e relatórios personalizados.

A estrutura de diretórios do projeto será organizada da seguinte forma:

```
/backend
└── src
    ├── controller
    ├── service
    ├── repository
    ├── domain
```

```
    └── config

/web
  └── src
    ├── components
    ├── pages
    └── services

/mobile
  └── lib
    ├── screens
    ├── widgets
    └── services
```

Ambientes definidos:

- **Desenvolvimento (DEV)**: usado para testes locais e homologações internas.
- **Homologação (STAGING)**: ambiente intermediário para QA e validação com stakeholders.
- **Produção (PROD)**: ambiente final em nuvem com autoescalabilidade.

CI/CD:

- O projeto será integrado com GitHub Actions e pipelines de CI/CD usando GitHub + Docker + AWS.
- Cada push para o branch main aciona testes automatizados e, se aprovados, inicia deploy para staging.
- Releases para produção serão feitas via pull request e revisão manual.

Containers:

- Todos os serviços serão empacotados com Docker.
- O orquestrador principal será Kubernetes (AWS EKS) para garantir resiliência, balanceamento e escalabilidade.

8. Tamanho e Desempenho

A aplicação deve rodar através de um site próprio na versão web para os administradores do sistema e das marcas parceiras e por meio de dispositivos móveis na versão mobile que será para o uso do usuário final do GreenLeens.

Nosso compromisso com o desempenho visa garantir que a experiência de uso seja sempre fluida e responsiva, independentemente da demanda:

- **Latência:** Para as ações mais críticas, como consultar os dados de sustentabilidade de um produto ou gerenciar GreenCoins, o objetivo é que a resposta do sistema não ultrapasse 500 milissegundos. Queremos que a informação esteja ali, quase instantaneamente.
- **Vazão (*Throughput*):** O sistema será dimensionado para suportar inicialmente 1.000 requisições por segundo. E, pensando no crescimento, a arquitetura já prevê a capacidade de escalar para picos de até 5.000 requisições por segundo, assegurando estabilidade mesmo nos momentos de maior movimento.
- **Estimativa de volume:** Estamos nos preparando para gerenciar uma vasta coleção de informações, com a expectativa de milhões de produtos cadastrados. Quanto aos usuários, projetamos centenas de milhares de ativos mensalmente, e nossa arquitetura estará pronta para acompanhar essa expansão contínua.

- **Inteligência Artificial (IA):** As funcionalidades de IA, como as recomendações personalizadas, envolvem um processamento mais intensivo. Embora essas operações possam ter uma latência um pouco maior (algo em torno de 3 a 5 segundos para cálculos em lote ou análises mais complexas), elas serão executadas de forma assíncrona. Assim, garantimos que a fluidez da interface principal do usuário não seja comprometida enquanto a IA trabalha em segundo plano.

9. 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.

A qualidade do software GreenLeens é um pilar fundamental de sua arquitetura. Adotamos o padrão MVC e princípios de Orientação a Objetos não apenas por boas práticas, mas porque sabemos que isso garante um código-fonte mais organizado, facilitando a manutenibilidade e a portabilidade do sistema. Além desses ganhos intrínsecos, priorizamos os seguintes atributos de qualidade, essenciais para o sucesso do GreenLeens:

Disponibilidade: Nosso objetivo é que o GreenLeens esteja sempre online. Por isso, a meta é alcançar um *uptime* de 99.9% para as funcionalidades críticas, assegurado por estratégias robustas de balanceamento de carga, auto scaling e replicação de banco de dados. Isso significa que os usuários terão acesso contínuo às informações e recursos do aplicativo.

Confiabilidade: Acreditamos na precisão e consistência dos nossos dados. As informações de sustentabilidade dos produtos, os detalhes dos itens e as transações de GreenCoins serão protegidos contra perdas e inconsistências. Garantimos isso através de transações de banco de dados ACID e planos de backup e recuperação de desastres bem definidos.

Segurança: A proteção dos dados dos nossos usuários e parceiros é primordial. Implementaremos mecanismos robustos para gerenciar o acesso, com autenticação segura e controle de acesso baseado em papéis (usuário, marca, administrador).Todos os dados serão

criptografados, tanto em trânsito (usando HTTPS/TLS) quanto em repouso (no banco de dados e armazenamento). Utilizaremos firewalls de aplicação (como o AWS WAF) e conduziremos revisões de código rigorosas para mitigar ataques comuns (como injeção SQL e XSS).

Usabilidade: O GreenLeens foi concebido para ser intuitivo. Nossas interfaces de usuário, tanto no mobile quanto na web, serão responsivas e fáceis de usar, com foco em proporcionar uma experiência fluida que incentive a interação e a adoção contínua do aplicativo.

Escalabilidade: Como um projeto com potencial de crescimento, a arquitetura de nuvem na AWS foi escolhida especificamente para permitir que o GreenLeens escale horizontalmente. Isso nos permite lidar com um volume crescente de usuários e dados de produtos, suportando picos de demanda sem comprometer o desempenho.

Manutenibilidade: Acreditamos que um código limpo é um código sustentável. A estrutura modular do nosso software, aliada à aplicação de padrões de design e às práticas de CI/CD, facilitará significativamente a identificação e correção de bugs, bem como a implementação eficiente de novas funcionalidades.

Portabilidade: Pensando na flexibilidade e no alcance, optamos por tecnologias multiplataforma (Flutter para mobile, React para web e Java para o backend). Essa escolha, combinada com a implantação em nuvem (independente de hardware físico), garante que o sistema possa ser executado em diversos ambientes e dispositivos sem grandes adaptações.

10. REFERÊNCIAS BIBLIOGRÁFICAS

KRUCHTEN, Philippe. *Architectural Blueprints—The "4+1" View Model of Software Architecture*. IEEE Software, v. 12, n. 6, p. 42–50, 1995.

SOMMERVILLE, Ian. *Engenharia de Software*. 10. ed. São Paulo: Pearson, 2019.

PRESSMAN, Roger S.; MAXIM, Bruce R. *Engenharia de Software: Uma Abordagem Profissional*. 8. ed. São Paulo: McGraw-Hill, 2016.

FOWLER, Martin. *UML Essencial*. 3. ed. Porto Alegre: Bookman, 2005.

PEREIRA, Luciana M. de Souza et al. *Arquitetura de Software com ênfase em Microsserviços*. Brasília: Ministério da Economia, 2021. Disponível em: https://www.gov.br/economia/arquitetura-software.pdf. Acesso em: 02 jun. 2025.

GOOGLE CLOUD. *Firebase Documentation*. Disponível em: https://firebase.google.com/docs. Acesso em: 05 jun. 2025.

LUCIDCHART. *What is a Use Case Diagram?*. Disponível em: https://www.lucidchart.com/pages/uml-use-case-diagram. Acesso em: 08 jun. 2025.

GREENPEACE BRASIL. *A Pegada da Moda: O impacto ambiental das nossas roupas*. 2020. Disponível em: https://www.greenpeace.org/brasil/. Acesso em: 05 jun. 2025.