Fatec Diadema Luigi Papaiz

Centro Paula Souza

SÃO PAULO GOVERNO DO ESTADO
Secretaria de Desenvolvimento Econômico

SL – Smart Leaf – Parte 2

Documento de Arquitetura de Software

Versão 6.0

Histórico de Revisões			
Versão	**Data**	**Descrição**	**Autor**
1.0	06/05/2024	Início do Documento	Kaiki Kenji Fukumoto Aoto
2.0	07/05/2024	Implementação dos diagramas prontos	Kaiki Kenji Fukumoto Aoto
2.1	11/05/2024	Diagrama de Pacote	Kaiki Kenji Fukumoto Aoto
3.0	13/05/2024	Correções nas estruturas do documento	Kaiki Kenji Fukumoto Aoto
4.0	14/05/2024	Revisão do documento	Kaiki Kenji Fukumoto Aoto
5.0	15/05/2024	Correções de escrita	Kaiki Kenji Fukumoto Aoto
6.0	16/06/2024	Finalização do Documento	Kaiki Kenji Fukumoto Aoto

Lista de Figuras

Lista de Tabelas

Sumário

Documento de Arquitetura de Software

1. Introdução

Este documento oferece uma visão geral arquitetural de todo o sistema da Smart Leaf, uma plataforma de gestão de energia sustentável. Utilizando diversas visões arquiteturais, o documento busca representar diferentes aspectos do sistema e capturar as decisões arquiteturais significativas, que foram utilizadas durante o processo de desenvolvimento. Este documento desempenha um papel importante na documentação do projeto, no qual fornece uma compreensão detalhada da arquitetura do sistema, como seções, componentes e aspectos arquiteturais de forma aprofundada. Sendo uma referência para a implementação e manutenção da plataforma Smart Leaf, onde espera-se que este documento atue como uma fonte de informação fundamental para compreender a arquitetura do sistema e orientar suas atividades relacionadas ao projeto.

2. Identificação do Projeto

Esse tópico fornece as informações básicas sobre o projeto. Fornecendo informações cruciais como o nome do projeto, o requisitante e responsável pelo projeto. Dessa forma essa seção além de definir o nome do projeto, esclarece os papeis e responsabilidades de cada parte envolvida no projeto.

Tabela 1 Indentificação do Projeto

Projeto	SL – Smart Leaf
Requisitante	Comunidade Local (FATEC e Arredores)
Responsável do Projeto	Kaiki Kenji Fukumoto Aoto

Fonte: Autores

3. Representação Arquitetural

Neste tópico, é resumido a arquitetura de software do sistema da Smart Leaf, utilizando várias visões arquiteturais para abordar diferentes aspectos do sistema.

- A Visão de Caso de Uso descreve os principais casos de uso, identificando funcionalidades e interações com os usuários, com o Diagrama de Caso de Uso foi ilustrado esses casos e suas relações.
- A Visão de Processos detalha os fluxos de trabalho, mostrando como as informações são processadas, com o diagrama de Atividades (BPMN) foi representado o fluxo de atividades.
- A Visão Lógica apresenta a estrutura do sistema, incluindo componentes principais e suas interações, com o Diagrama de Classes foi mostrado a estrutura de dados e relacionamentos entre componentes.
- Na Visão de Implementação, descrevemos como a arquitetura é implementada fisicamente, incluindo detalhes sobre linguagens de programação, frameworks e tecnologias utilizadas, com o Diagrama de Pacotes foi representado a organização do código fonte e suas dependências.

4. Metas e Restrições da Arquitetura

Neste tópico, é descrito os requisitos e objetivos do software que auxiliaram diretamente a arquitetura do sistema, como também as restrições especiais que precisam ser consideradas durante o processo de design e implementação.

4.1. Metas da Arquitetura

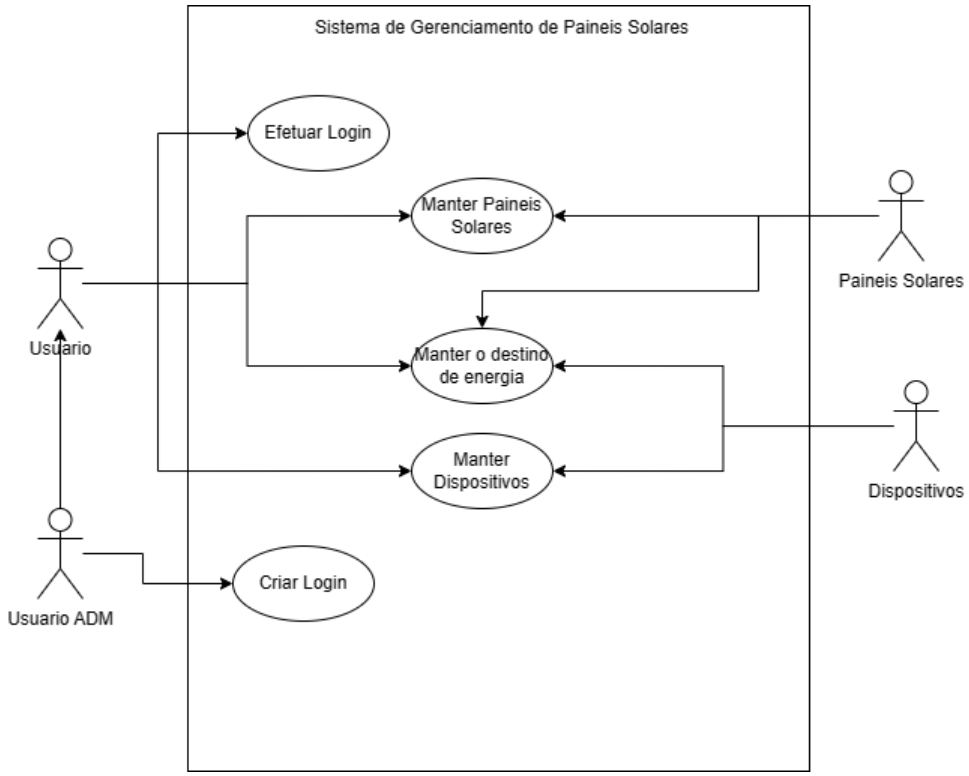
- Segurança: Garantir que o sistema seja seguro contra ameaças de segurança, como acesso não autorizado e vazamento de informações pessoais.
- Desempenho: Assegurar que o sistema seja capaz de lidar com a carga de trabalho esperada, mantendo tempos de respostas aceitáveis e alta disponibilidade.
- Escalabilidade: Projetar o sistema de forma que possa ser escalado facilmente para ligar com um aumento no número de usuários e volume de dados.

4.2. Restrições da Arquitetura

- Restrição ao uso do framework Spring Boot para desenvolvimento do backend do sistema, devido aos benefícios de produtividade, facilidade de configuração e integração que ele proporciona.
- Restrição ao uso do banco de dados MySQL como o sistema de gerenciamento de banco de dados relacional (SGBDR) para armazenamento e recuperação de dados, devido à sua popularidade, desempenho e confiabilidade.
- Restrição ao uso de JavaScript, HTML e CSS para o desenvolvimento do frontend do sistema Smart Leaf. Essas tecnologias são amplamente adotadas para a criação de interfaces de usuário dinâmicas e responsivas, permitindo uma experiência de usuário moderna e eficaz.
- Restrição quanto ao uso de IDEs como IntelliJ IDEA ou Visual Studio Code para o desenvolvimento do projeto com Spring Boot, MySQL, Javascript, HTML e CSS

5. Visão de Casos de Uso

Figura 1 Diagrama de Caso de Uso



Fonte: Autores

5.1. Descrição e Realizações dos Caso de Uso

Nesse tópico é brevemente descrito a forma que os casos de uso ocorrem, descrevendo como o sistema e o usuário se relacionam. Além disso é mostrando as realizações dos casos de uso, exibindo como cada componente do sistema do caso de uso atua dentro do sistema.

Tabela 2 Caso de Uso 01

Nome do Caso de Uso	**UC01 - Efetuar Login**
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário possa efetuar o login
Pré-Condições	É necessário o usuário possuir uma conta
Pós-Condições	O usuário deve conseguir logar no site
Cenário Principal	
Ações do Ator	**Ações do Sistema**
1. Usuário insere email e senha e clica no botão efetuar login	
	2. Valida se o email e senha inseridos existem no sistema
	3. Se existir, o sistema loga do usuário no site
Restrições/Validações:	1. Os campos de email e senha não podem estar vazios
Cenário Exceção - Email ou Senha Incorretos	
Ações do Ator	**Ações do Sistema**
	1. Emitir um alerta informando ao usuário que a senha ou email inseridos, estão incorretos
	2. Não efetuar login do usuário
3. Usuário corrige os campos incorretos e retorna ao passo 2 do cenário principal	
Cenário Exceção - Não possui login cadastrado	
Ações do Ator	**Ações do Sistema**

	1. Emitir um alerta informando ao usuário que o usuário não possui login
	2. Não efetuar login do usuário
Contribuição dos Elementos de Design	
1. A classe AuthUserService é responsável por validar as credenciais do usuário.	
2. A interface de usuário LoginPage permite que o usuário insira a suas credenciais.	
3. O controlador AuthUserController trata as requisições de autenticação de usuário.	

Fonte: Autores

Tabela 3 Caso de Uso 02

Nome do Caso de Uso	**UC02 - Criar Login**
Ator Principal	Usuário ADM
Atores Secundários	
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário ADM possa criar um logins para outros usuários
Pré-Condições	Usuario ADM, deve estar possuir login no sistema
Pós-Condições	O usuário ADM deve conseguir criar um logins para outros usuários no site
Cenário Principal	
Ações do Ator	**Ações do Sistema**
1. Usuário ADM insere suas informações pessoais dos outros usuários	
	2. Valida se as informações inseridas
	3. Verifica se o email inserido já existe no sistema
	4. Se não existir, cria um login para o usuário
Restrições/Validações:	1. Os campos das informações pessoais não podem estar vazios
	2. O email inserido não pode já estar cadastrado no sistema

Cenário Exceção - Informações pessoais incorretas	
Ações do Ator	**Ações do Sistema**
	1. Emitir um alerta informando ao usuário ADM que as informações pessoais inseridas, estão incorretas
	2. Não efetuar cadastro de login
3. Usuário ADM corrige os campos incorretos e retorna ao passo 2 do cenário principal	
Cenário Exceção - Email já cadastrado no sistema	
Ações do Ator	**Ações do Sistema**
	1. Emitir um alerta informando ao usuário ADM que o email inserido já está cadastrado no sistema
	2. Não efetuar cadastro de login
3. Usuário ADM insere outro email e retorna ao passo 2 do cenário principal	
Contribuição dos Elementos de Design	
1. A classe CreateUserService gerencia as operações de adição, remoção e modificação de conta de usuário.	
2. A interface de usuário CreateUserPage permite que o usuário insira os seus dados pessoais.	
3. O controlador CreateUserController trata as requisições de criação de conta do usuário.	

Fonte: Autores

Tabela 4 Caso de Uso 03

Nome do Caso de Uso	UC03 - Manter Painéis Solares
Ator Principal	Usuário
Atores Secundários	Painéis Solares
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário, possa realizar o cadastro, edição e alteração dos painéis solares
Pré-Condições	É necessário que o usuário esteja logado no sistema
Pós-Condições	O usuário deve conseguir manipular os painéis solares
Cenário Principal	

Ações do Ator	Ações do Sistema
1. O usuário escolhe um painel solar para modificar	
2. Realiza as alterações do painel solar e clica no botão de executar	
	3. Recebe as informações da modificação
	4. Executa as alterações
Restrições/Validações:	1. Usuário deve estar logado no sistema para realizar qualquer processo de modificação
Cenário Exceção - Erro ao realizar modificação no painel solar	
Ações do Ator	**Ações do Sistema**
	1. Emitir um alerta informando ao usuário que a modificação não foi realizada
	2. Não realiza a modificação do painel solar
Contribuição dos Elementos de Design	
1. A classe SolarPanelService gerencia as operações de adição, remoção e modificação de painéis solares.	
2. A interface de usuário PanelPage permite que o usuário visualize e interaja com os painéis solares. Além de inserir as informações necessárias para criação do panel solar.	
3. O controlador SolarPanelController trata as requisições de adição, remoção e modificação de painéis solares.	

Fonte: Autores

Tabela 5 Caso de Uso 04

Nome do Caso de Uso	UC04 - Manter dispositivos
Ator Principal	Usuário
Atores Secundários	Dispositivos
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário, possa realizar o cadastro, edição e alteração dos dispositivos
Pré-Condições	É necessário que o usuário esteja logado no sistema
Pós-Condições	O usuário deve conseguir manipular os dispositivos

Cenário Principal	
Ações do Ator	**Ações do Sistema**
1. O usuário escolhe um dispositivo para modificar	
2. Realiza as alterações do dispositivo e clica no botão de executar	
	3. Recebe as informações da modificação
	4. Executa as alterações
Restrições/Validações:	1. Usuário deve estar logado no sistema para realizar qualquer processo de modificação
Cenário Exceção - Erro ao realizar modificação no dispositivo	
Ações do Ator	**Ações do Sistema**
	1. Emitir um alerta informando ao usuário que a modificação não foi realizada
	2. Não realiza a modificação do dispositivo
Contribuição dos Elementos de Design	
1. A classe DevicesService gerencia as operações de adição, remoção e modificação de dispositivos.	
2. A interface de usuário DevicesPage permite que o usuário visualize e interaja com os dispositivos. Além de inserir as informações necessárias para criação do dispositivo.	
3. O controlador DevicesController trata as requisições de adição, remoção e modificação de dispositivos.	

Fonte: Autores

Tabela 6 Caso de Uso 05

Nome do Caso de Uso	**UC05 - Manter o destino de energia**
Ator Principal	Usuário
Atores Secundários	Dispositivos, Painéis Solares
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário, possa realizar o cadastro, edição e alteração dos destinos de energia de para cada dispositivo
Pré-Condições	É necessário que o usuário esteja logado no sistema

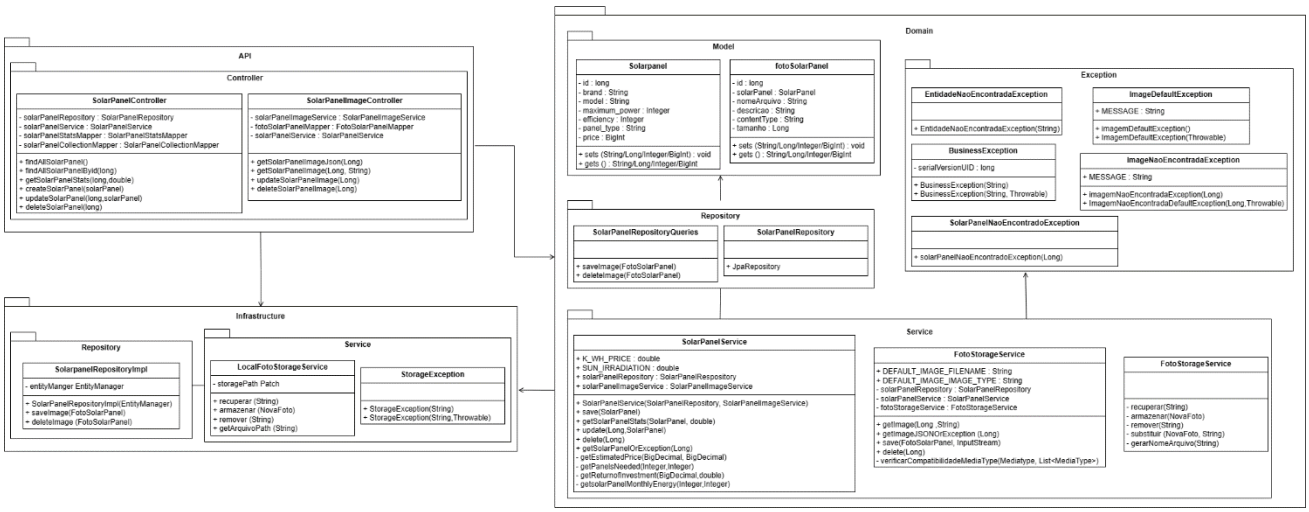
Pós-Condições	O usuário deve conseguir manipular os o destino de energia
Cenário Principal	
Ações do Ator	**Ações do Sistema**
1. O usuário escolhe um destino de energia para modificar	
2. Realiza as alterações do destino de energia e clica no botão de executar	
	3. Recebe as informações da modificação
	4. Executa as alterações
Restrições/Validações:	1. Usuário deve estar logado no sistema para realizar qualquer processo de modificação
Cenário Exceção - Erro ao realizar modificação no destino de energia	
Ações do Ator	**Ações do Sistema**
	1. Emitir um alerta informando ao usuário que a modificação não foi realizada
	2. Não realiza a modificação no destino de energia
Contribuição dos Elementos de Design	
1. A classe SolarPanelDeviceService gerencia as operações de adição, remoção e modificação dos destinos de energia.	
2. A interface de usuário SolarPanelDevicePage permite que o usuário visualize e interaja com os destinos de energia. Além de permitir que o usuário faça a relação entre o painel solar e o dispositivo	
3. O controlador SolarPanelDeviceController trata as requisições de adição, remoção e modificação de destinos de energia.	

Fonte: Autores

6. Visão Lógica

Nesse tópico, é exibido a estrutura interna do sistema da Smart Leaf, exibindo seus principais componentes e suas interações. Sendo importante para compreender como as diferentes partes do sistema se relacionam e como trabalham em conjunto para cumprir seus objetivos. O diagrama de classes abaixo oferece uma representação visual dessa estrutura, identificando as classes principais, seus atributos, métodos e os relacionamentos entre elas.

Figura 2 Diagrama de Classe



Fonte: Autores

6.1. Visão Geral

Dentro do projeto do sistema da Smart Leaf, existem as seguintes camadas que estão organizadas da seguinte forma:

1 – Camada de interface do Usuario (UI) – (Pacote de Front-end), essa camada contém todos os elementos relacionados à apresentação e interação com o usuário, como telas, formulários e elementos visuais.

2 – Camada de Lógica de Negócios (Backend) – (Pacote de Serviços | Pacote de Controladores | Pacote da API), essa camada contém toda a lógica de processamento e manipulação de dados da aplicação, como as regras de negócio e as operações de CRUD.

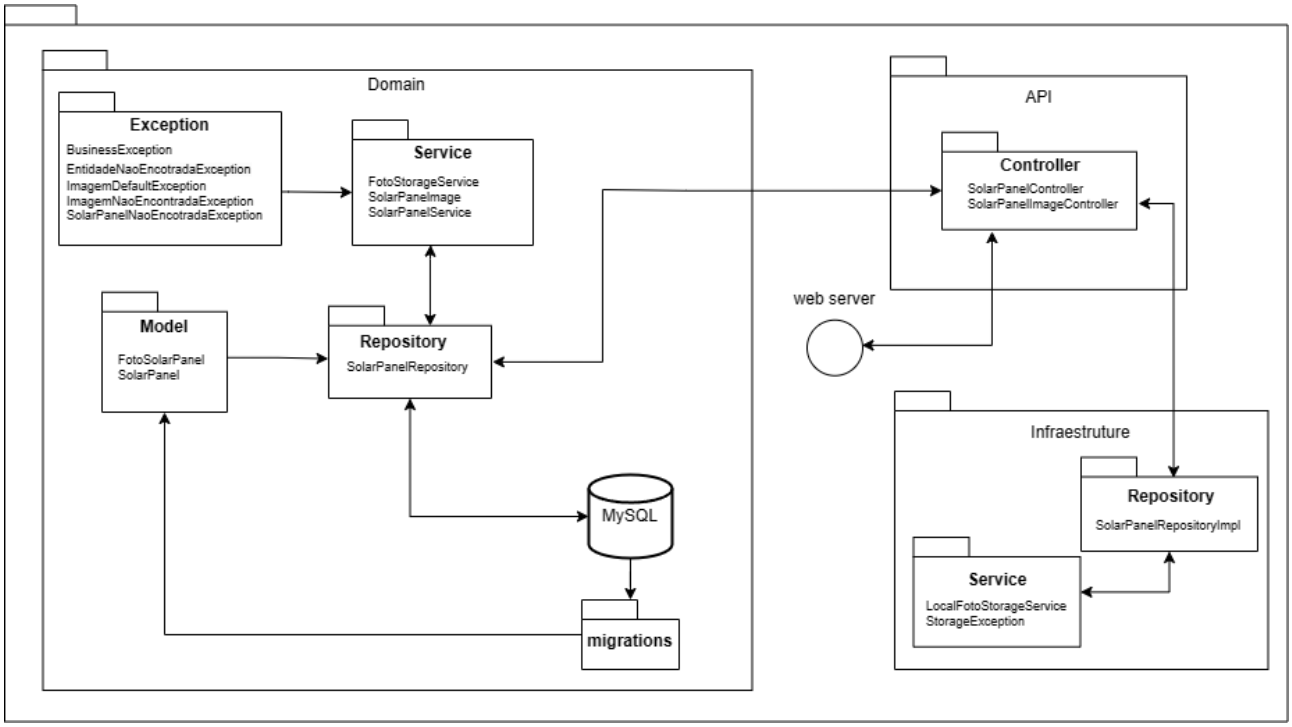
3 – Camada de Persistência de Dados (Backend) – (Pacote de Repositórios), essa camada é responsável pelo acesso e armazenamento de dados no banco de dados, como operações de leitura, gravação e consulta.

Dessa forma o relacionamento entre as camadas, seguem uma sequência definida, a camada de interface solicita as requisições para a camada de lógica de negócios, que processa essa requisição e chama se necessário a camada de Persistência de dados que realiza o acesso com o banco de dados e os resultados então são repassados da camada de lógica de negócios para a interface, que os apresenta ao usuário.

6.2. Pacotes de Design Significativos do Ponto de Vista da Arquitetura

Neste tópico, é apresentado os pacotes de design que desempenham um papel importante na estrutura e organização da arquitetura do sistema Smart Leaf, exibindo uma visão clara sobre o código-fonte. Cada pacote aborda uma área específica do sistema, que desempenha uma funcionalidade ou responsabilidade. Na figura a seguir é exibido a estrutura e as relações entre os pacotes de design, oferecendo uma visão visual da arquitetura geral do sistema.

Figura 3 Diagrama de Pacote



Fonte: Autores

6.2.1. Descrição dos pacotes de design

Nesse tópico é descrito os pacotes exibidos no diagrama de pacote. Mostrando o seu nome, seu objetivo dentro do sistema e descrito detalhadamente a suas subseções.

Tabela 7 Pacote Service(Domain)

Nome do Pacote	**Service (Domain)**
Descrição	Este pacote contém classes responsáveis por fornecer serviços essenciais para o funcionamento do sistema. Ele encapsula a lógica de negócios e fornece uma interface para interagir com outras camadas.
Descrição das Subseções	
Nome da Subseção (1)	SolarPanelService
Descrição	Gerencia operações relacionadas aos painels solares, como registro, modificações e exclusão.
Responsabilidades e Operações	getSolarPanelStats(SolarPanel solarPanel, double kwh)
	update(Long id, SolarPanel solarPanel)
	delete(Long id)
	getSolarPanelOrException(Long id)
	getEstimatedPrice(BigDecimal panelsNeeded, BigDecimal price)
	getPanelsNeeded(Integer maximumPower, Integer efficiencyPercentage, double kwh)
	getReturnOfInvestment(BigDecimal investment, double solarPanelMonthlyEnergy)

	getsolarPanelMonthlyEnergy(Integer maximumPower, Integer efficiencyPercentage)
Nome da Subseção (2)	SolarPanelImageService
Descrição	Gerencia operações relacionadas as fotos do painels solares, como validação, registro, modificações e exclusão
Responsabilidades e Operações	getImage(Long id, String acceptHeader)
	getImageJSONOrException(Long id)
	save(FotoSolarPanel foto, InputStream dadosArquivo)
	delete(Long id)
	verificarCompatibilidadeMediaType(MediaType mediaTypeFoto, List<MediaType> providedMediaTypes)
Nome da Subseção (3)	FotoStorageService
Descrição	Serve como interface para serviços relacionados ao armazenamento de fotos.
Responsabilidades e Operações	armazenar(NovaFoto novaFoto)
	recuperar(String nomeArquivo)
	save(FotoSolarPanel foto, InputStream dadosArquivo)
	remover(String nomeArquivo)
	substituir(NovaFoto novaFoto, String nomeArquivoAntigo)
	gerarNomeArquivo(String nomeOriginal)

Fonte: Autores

Tabela 8 Pacote Exception (Domain)

Nome do Pacote	**Exception (Domain)**
Descrição	Este pacote contém classes responsáveis por realizar a lógica e o tratamento de erros que podem ocorrer durante o funcionamento do sistema.
Descrição das Subseções	
Nome da Subseção (1)	BusinessException
Descrição	Classe que serve como base geral, para ser usada em outras exeções (Interface)
Responsabilidades e Operações	BusinessException(String message)
	BusinessException(String message, Throwable cause)
Nome da Subseção (2)	EntidadeNãoEcontradaException
Descrição	Classe que é acionada quando uma entidade não é encontrada no sistema

Responsabilidades e Operações	EntidadeNaoEncontradaException(String message)
Nome da Subseção (3)	ImagemDefaultException
Descrição	Classe que é acionada quando uma imagem não pode ser apagada
Responsabilidades e Operações	ImagemDefaultException()
	ImagemDefaultException(Throwable cause)
Nome da Subseção (4)	ImagemNãoEcontradaException
Descrição	Classe que é acionada quando uma imagem não pode ser buscada e recuperada
Responsabilidades e Operações	ImagemNaoEncontradaException(Long id)
	ImagemNaoEncontradaException(Long id, Throwable cause)
Nome da Subseção (5)	SolarPanelNaoEcontradoException
Descrição	Classe que é acionada quando um painel solar não foi encontrado
Responsabilidades e Operações	SolarPanelNaoEncontradoException(Long id)

Fonte: Autores

Tabela 9 Pacote Model (Domain)

Nome do Pacote	**Model (Domain)**
Descrição	Este pacote contém as classes que representam os modelos de dados essenciais para o funcionamento do sistema. Ele encapsula a estrutura e o comportamento dos objetos de dados do sistema.
Descrição das Subseções	
Nome da Subseção (1)	SolarPanel
Descrição	Representa um painel solar armazenado no sistema
Responsabilidades e Atributos	id : Long
	brand: String
	model : String
	maximumPower : Integer
	Efficiency : Integer
	panelType : String
	price : BigDecimal
Nome da Subseção (2)	FotoSolarPanel
Descrição	Representa uma foto do painel solar armazenado no sistema

Responsabilidades e Atributos	id : Long
	nomeArquivo : String
	descricao : String
	contentType : String
	tamanho : Long

Fonte: Autores

Tabela 10 Pacote Repository (Domain)

Nome do Pacote	**Repository (Domain)**
Descrição	Este pacote contém as interfaces e classes responsáveis pela comunicação com o banco de dados para operações de persistência relacionadas aos modelos de dados do sistema.
Descrição das Subseções	
Nome da Subseção (1)	SolarPanelRepository
Descrição	Interface responsável por definir os métodos de acesso aos dados relacionados aos paineis solares no banco de dados.
Responsabilidades e Atributos	N/A (Interface)
Nome da Subseção (2)	SolarPanelRepositoryQueries
Descrição	Interface responsável por definir os métodos de acesso aos dados relacionados às fotos no banco de dados.
Responsabilidades e Atributos	N/A (Interface)

Fonte: Autores

Tabela 11 Pacote Controller (API)

Nome do Pacote	**Controller (API)**
Descrição	Este pacote contém classes responsáveis por fornecer serviços essenciais para o funcionamento da API. Ele encapsula a lógica de negócios e fornece rotas para efetuar as operações.
Descrição das Subseções	
Nome da Subseção (1)	SolarPanelController
Descrição	Gerencia os métodos de rotas da API
Responsabilidades e Atributos	findAllSolarPanels()
	findSolarPanelById(@PathVariable Long id)
	getSolarPanelStats(@PathVariable Long id, @RequestParam("kwh") double kwh)

	createSolarPanel(@Valid @RequestBody SolarPanel solarPanel)
	updateSolarPanel(@PathVariable Long id, @Valid @RequestBody SolarPanel solarPanel)
Nome da Subseção (2)	SolarPanelImageController
Descrição	Valida e Gerencia as operações relacionadas com as imagens por meio das requisições HTTP
Responsabilidades e Atributos	getSolarPanelImageJSON(@PathVariable Long id)
	getSolarPanelImage(@PathVariable Long id, @RequestHeader(name = "accept") String acceptHeader)
	getSolarPanelStats(@PathVariable Long id, @RequestParam("kwh") double kwh)
	updateSolarPanelImage(@PathVariable Long id, @Valid FotoProdutoInput fotoProdutoInput)
	deleteSolarPanelImage(@PathVariable Long id)

Fonte: Autores

Tabela 12 Pacote Repository (Infrastructure)

Nome do Pacote	**Repository (Infrastructure)**
Descrição	Este pacote contém as interfaces e classes responsáveis pela comunicação com o banco de dados para operações de persistência relacionadas aos modelos infraestruturais do sistema.
Descrição das Subseções	
Nome da Subseção (1)	SolarPanelRepositoryImpl
Descrição	Interface implementada para ser responsável por realizar consultas perosnalizadas
Responsabilidades e Atributos	deleteImage(FotoSolarPanel fotoSolarPanel)
	saveImage(FotoSolarPanel foto)
	SolarPanelRepositoryImpl(EntityManager entityManager)

Fonte: Autores

Tabela 13 Pacote Service (Infrastructure)

Nome do Pacote	**Service (Infrastructure)**
Descrição	Este pacote contém classes responsáveis por fornecer serviços que auxiliam a ligação entre a infraestrutura local com o sistema
Descrição das Subseções	
Nome da Subseção (1)	LocalFotoStorageService

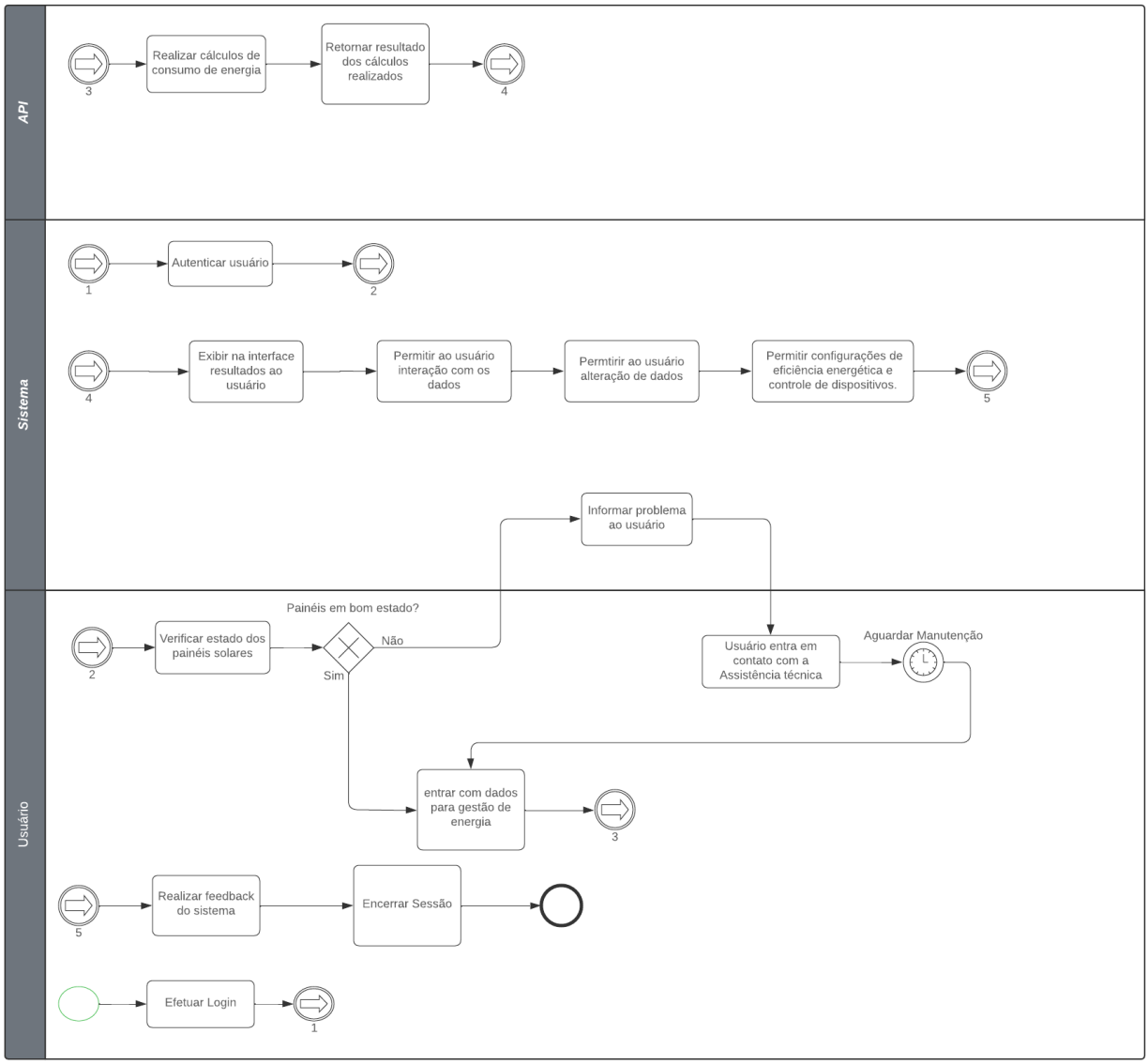
Descrição	Classe Implementação, responsável por armazenar, recuperar e remover fotos do sistema de arquivos local.
Responsabilidades e Atributos	recuperar(String nomeArquivo)
	armazenar(NovaFoto novaFoto)
	remover(String nomeArquivo)
	getArquivoPath(String nomeArquivo)
Nome da Subseção (2)	StorageException
Descrição	Classe Implementação, responsável por armazenar, recuperar e remover fotos do sistema de arquivos local.
Responsabilidades e Atributos	StorageException(String message)
	StorageException(String message, Throwable cause)

Fonte: Autores

7. Visão de Processos

Nesse tópico é exibido como diferentes fluxos de trabalho e processos ocorrem no sistema da Smart Leaf, sendo importante para entender como as informações são processadas e movimentadas ao longo do sistema. O diagrama BPMN abaixo oferece uma representação visual desses processos, mostrando as atividades, eventos e gateways que compõe o funcionamento do sistema.

Figura 4 BPMN



Fonte: Autores

8. Visão de Negócio da Implementação

O sistema é dividido em três camadas principais: Apresentação, Lógica de Negócios e Acesso a Dados.
Divisão em Camadas

- Apresentação: Responsável por receber as requisições dos usuários e apresentar as interfaces gráficas. Utiliza tecnologias web como HTML, CSS e JavaScript.
- Lógica de Negócios: Contém as regras de negócio do sistema, coordenando as operações e processos internos. Implementado em Java usando o Spring FrameWork.
- Acesso a Dados: Responsável por realizar operações de leitura e escrita no banco de dados. Utiliza tecnologias como JPA e Hibernate para a integração com o banco de dados relacional MySQL.

Subsistemas

- Autenticação e Autorização: Gerencia o controle de acesso ao sistema, autenticando usuários e autorizando suas ações.
- Gestão de Painéis Solares: Subsistema responsável por gerenciar os painéis solares do sistema, incluindo cadastro, edição e exclusão.
- Gestão de Dispositivos: Subsistema responsável por gerenciar dispositivos do sistema, incluindo cadastro, edição e exclusão.
- Gestão de Relacionamento entre Painéis solares e Dispositivos: Subsistema responsável por gerenciar o relacionamento entre os dispositivos e os painéis solares do sistema, incluindo o cadastro, a edição e a exclusão do relacionamento.

Componentes Significativos

- API REST: Gerencia e fornece as operações de CRUD para o sistema, permitindo a integração do sistema principal com os outros subsistemas.
- Bootstrap: Conjunto de componentes reutilizáveis para a construção da interface de usuário, além de tecnologias que permitem que o site permaneça responsiva em diferentes tipos de tamanho de tela.

9. Tamanho e Desempenho

Nesse tópico é abordado o tamanho e desempenho que influenciam na arquitetura do sistema

- Escalabilidade: O sistema é capaz de lidar com um aumento na carga de trabalho ou volume de dados sem comprometer o desempenho, dessa forma é possível se adaptar em diferentes tipos de cenário.
- Disponibilidade: O sistema está disponível em grande parte do tempo, tendo um tempo de inatividade mínimo, sendo assim importante para garantir a gestão de energia e não afetar algum possível comprometimento de falta de energia em algum dispositivo.
- Desempenho: O sistema tem um tempo de resposta e capacidade de processamento rápido, devido apresentar estruturas em seu corpo que foram otimizadas para esse objetivo. Permitindo que o usuário não tenha que esperar muito tempo para visualizar as atualizações e modificações.
- Flexibilidade de Plataforma: Devido ao sistema ser alocado na web, todos os dispositivos que possuem uma conexão com internet e que consigam navegar de forma fluida pela web, poderão acessar o sistema da Smart Leaf.
- Restrições de Desempenho: Caso o usuário não tenha acesso a internet não será possível acessar o sistema. E apenas usuário cadastrados no sistema poderão acessar o sistema.
- Testes de Desempenho: Testes foram realizados em diferentes navegadores, com diferentes velocidades de internet e em diferentes tipos de dispositivos.

10. Qualidade

A qualidade do sistema da Smart Leaf foi avaliada por meio de testes, que não apenas verificaram a funcionalidade dos componentes, mas também a sua capacidade de atender a critérios essenciais. Como a sua habilidade de evoluir, operar consistentemente em ambientes diversos e garantir a segurança e privacidade dos dados.

- Testes de Confiabilidade: Testes que lidam com situações de falha foram efetuados e através disso, foram desenvolvidas estratégias de tolerância a falhas para garantir que o sistema seja altamente confiável e resiliente.
- Teste de Unidades: Testes realizados permitiram desenvolver componentes que atuem de forma modular e flexível, permitindo que novos recursos e funcionalidades sejam adicionados sem afetar o

funcionamento. Onde a existe a separação entre os componentes e o uso de padrões adequados facilitam a extensibilidade do sistema ao longo do tempo.

- Teste de Navegadores: O sistema foi acessado em diversos tipos de navegadores como: Google, Edge, Firefox e Opera, no qual todos os navegadores foram capazes de acessar de forma rápida e sem problemas.
- Teste de Dispositivos: A arquitetura foi projetada independentemente da plataforma, permitindo que o software seja executado em diferentes ambientes de hardware e software. Devido a adoção de padrões de desenvolvimento.
- Teste de Seguranças: A segurança dos dados e a proteção da privacidade dos usuários são elementos essenciais dentro do sistema. Dessa forma foi realizado testes com mecanismos de autenticação, autorização, criptografia e controle de acesso, para assim conseguir identificar e eliminar as vulnerabilidades e garantir a privacidade do usuário.

Através desses testes, foi comprovado a qualidade da arquitetura do software, garantindo que o sistema seja robusto, flexível e adaptável às necessidades em constante evolução dos usuários e do ambiente operacional. Isso resulta em uma boa experiência para o usuário, reduzindo custos operacionais e aumentando a competitividade do produto no mercado.

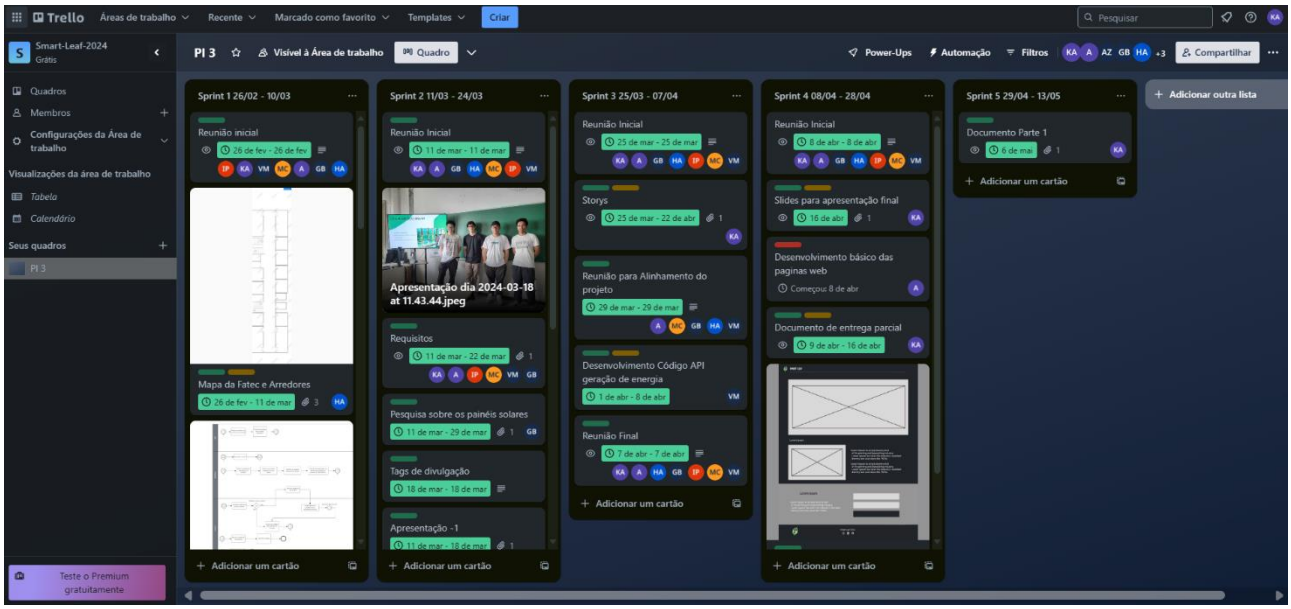
11. Anexos

Nesse tópico, é apresentado o cronograma do projeto, onde inicialmente foi elaborado a Estrutura Analítica do Projeto (EAP), que serviu como base para o planejamento inicial. Posteriormente foi implementado na prática através da plataforma Trello, onde foi possível visualizar as sprints, tarefas e assim consultar e gerenciar o progresso do projeto de forma eficaz.

11.1. Trello

Nesse tópico é exibido uma imagem do trello, exibindo como as sprints foram organizadas, como foram distribuídas as tarefas para os membros do projeto e mostra progresso geral do projeto do início ao fim.

Figura 5 Trello

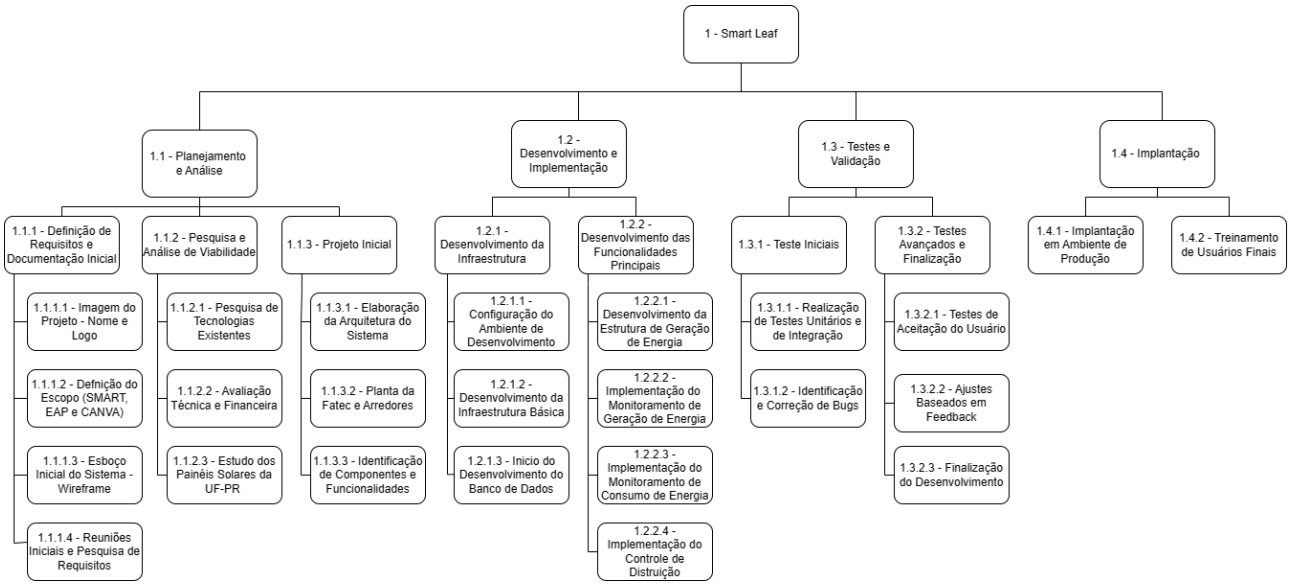


Fonte: Autores

11.2. EAP

Nesse tópico é exibido a estrutura analítica do projeto (EAP), mostrando o escopo do projeto, organizado de forma hierárquica, com a separação em fases.

Figura 6 EAP



Fonte: Autores

11.2.1. EAP nas Sprints

Nesse tópico é exibido a aplicação da estrutura da EAP nas sprints do projeto, mostrando em qual sprint foi realizada as tarefas determinadas dentro da EAP.

Tabela 14 Sprint 01

Sprint – 1 (26/02/2024 – 10/03/2024)	
Item	**Descrição**
1.1	Planejamento e Análise
1.1.1	Definição de Requisitos e Documentação Inicial
1.1.1.1	Imagem do Projeto - Nome e Logo
1.1.1.2	Defnição do Escopo (SMART, EAP e CANVA)
1.1.1.3	Esboço Inicial do Sistema – Wireframe

Fonte: Autores

Tabela 15 Sprint 02

Sprint – 2 (11/03/2024 – 24/03/2024)	
Item	**Descrição**
1.1.1.4	Reuniões Iniciais e Pesquisa de Requisitos
1.1.2	Pesquisa e Análise de Viabilidade
1.1.2.1	Pesquisa de Tecnologias Existentes

1.1.2.2	Avaliação Técnica e Financeira
1.1.2.3	Estudo dos Painéis Solares da UF-PR
1.1.3	Projeto Inicial
1.1.3.1	Elaboração da Arquitetura do Sistema

Fonte: Autores

Tabela 16 Sprint 03

Sprint – 3 (25/03/2024 – 07/04/2024)	
Item	**Descrição**
1.1.3.2	Planta da Fatec e Arredores
1.1.3.3	Identificação de Componentes e Funcionalidades
1.2	Desenvolvimento e Implementação
1.2.1	Desenvolvimento da Infraestrutura
1.2.1.1	Configuração do Ambiente de Desenvolvimento
1.2.1.2	Desenvolvimento da Infraestrutura Básica
1.2.1.3	Início do Desenvolvimento do Banco de Dados
1.2.2	Desenvolvimento das Funcionalidades Principais
1.2.2.1	Desenvolvimento da Estrutura de Geração de Energia

Fonte: Autores

Tabela 17 Sprint 04

Sprint 4 – (08/04/2024 – 29/04/2024)	
Item	**Descrição**
1.2.2.2	Implementação do Monitoramento de Geração de Energia
1.2.2.3	Implementação do Monitoramento de Consumo de Energia
1.2.2.4	Implementação do Controle de Distribuição
1.3	Testes e Validação
1.3.1	Teste Iniciais
1.3.1.1	Realização de Testes Unitários e de Integração
1.3.1.2	Identificação e Correção de Bugs

Fonte: Autores

Tabela 18 Sprint 05

Sprint 5 – (29/04/2024 – 13/05/2024)	
Item	**Descrição**
1.3.2	Testes Avançados e Finalização

1.3.2.1	Testes de Aceitação do Usuário
1.3.2.2	Ajustes Baseados em Feedback
1.3.2.3	Finalização do Desenvolvimento
1.4	Implantação
1.4.1	Implantação em Ambiente de Produção
1.4.2	Treinamento de Usuários Finais

Fonte: Autores

12. Referências

- LucidChart : O que é um diagrama de classe UML? | Lucidchart
- LucidChart : Diagrama de caso de uso UML: O que é, como fazer e exemplos | Lucidchart
- LucidChart : Tutorial sobre diagramas de pacotes | Lucidchart

13. Aprovações

Aprovações		
Participante	**Assinatura**	**Data**
Kaiki Kenji Fukumoto Aoto	Kaiki Kenji Fukumoto Aoto	19/05/2024