

**SMART
LEAF**

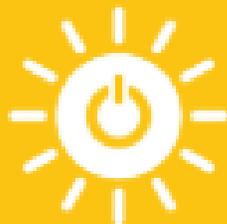
SOBRE O PROJETO

Desenvolver um sistema de gerenciamento de energia solar

- Simulação da geração de energia solar
- Distribuição de forma eficiente para (Locais ou Dispositivos)
- Inicialmente focado para Fatec e seus Arredores

ODS

7 ENERGIA ACESSÍVEL
E LIMPA



9 INDÚSTRIA, INOVAÇÃO
E INFRAESTRUTURA



11 CIDADES E
COMUNIDADES
SUSTENTÁVEIS



PROBLEMÁTICA



- Falta de Iluminação durante períodos noturnos
- Má gestão de energia durante período diurnos
- Auxiliar aos arredores da FATEC (Fornecer Energia extra)

Business Model Canvas

Desenhado para:

Smart Leaf

Desenhado por:

Data:

07/03/24

Versão:

1.0



Versão Original: Strategyzer.com

Este trabalho está licenciado sob a Licença Atribuição-Compartilhada 4.0 Internacional Creative Commons: 

CANVAS

REQUISITOS

Requisitos Funcionais

Registro de Painéis Solares

Registro de Dispositivos

Monitoramento de Geração de Energia

Modo Manual e Automático

Requisitos do Negócio

Promover Sustentabilidade Ambiental

Maximizar Eficiência Energética

Reduzir custos de Energia

Requisitos Não Funcionais

Segurança

Escalabilidade

Interface Intuitiva

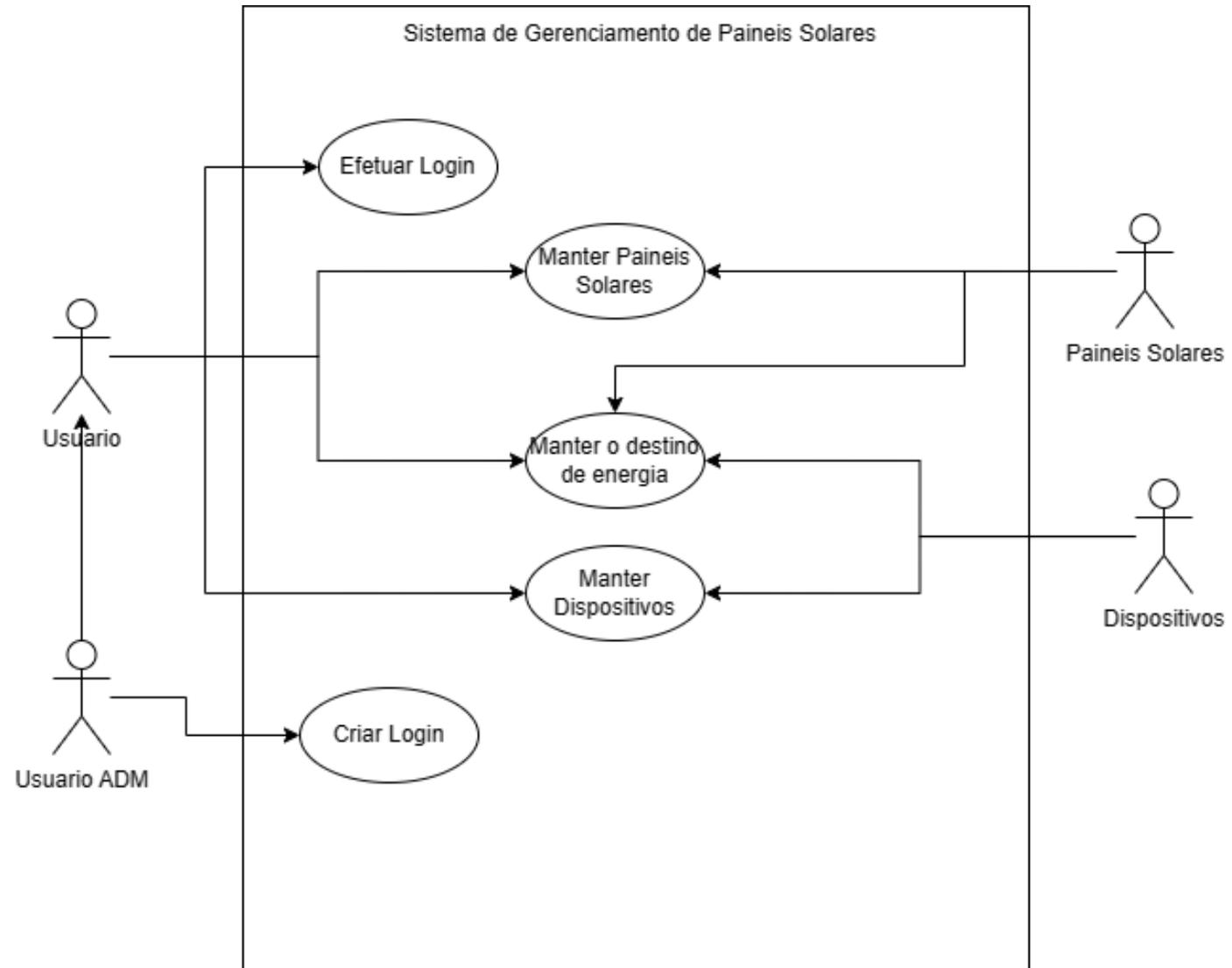
Responsividade

Requisitos do Usuário

Usuários ADM

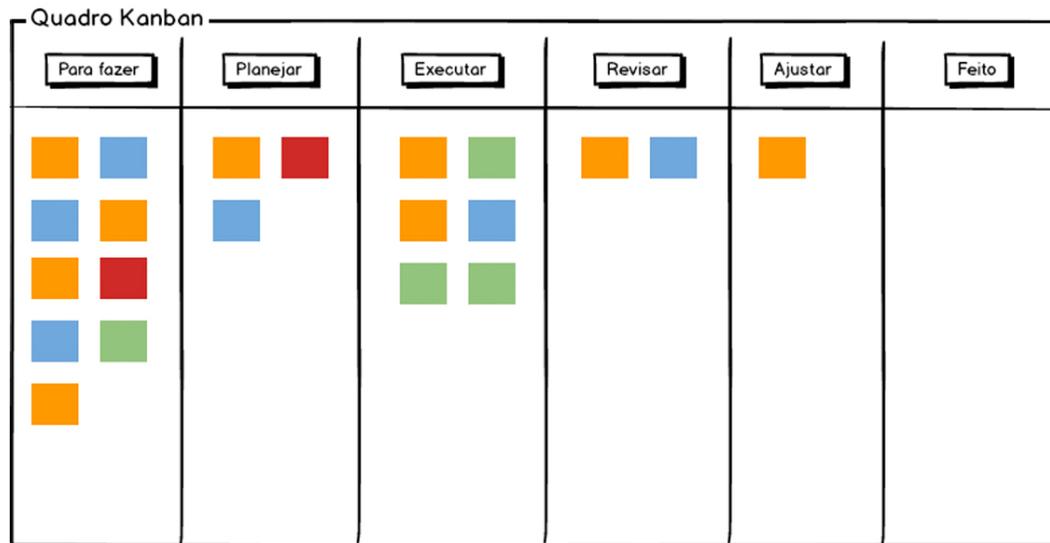
Facilidade de Configuração

DIAGRAMA DE CASO DE USO



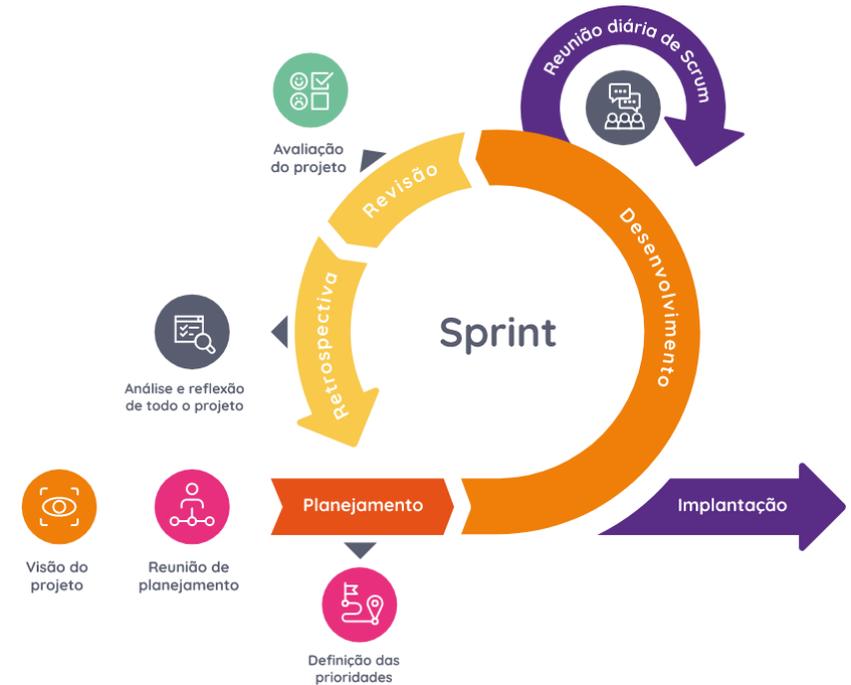
MÉTODO DE DESENVOLVIMENTO

Kanban



@balsamiq

Scrum



PLATAFORMA TRELLO

The image displays the Trello web interface for a project named "Smart-Leaf-2024". The main workspace is a Kanban board titled "PI 3" with a "Visível à Área de trabalho" (Visible to workspace) permission. The board is organized into five vertical lanes representing sprints:

- Sprint 1 (26/02 - 10/03):** Includes cards for "Reunião inicial", "Mapa da Fatec e Arredores", and "BPMN".
- Sprint 2 (11/03 - 24/03):** Includes cards for "Reunião Inicial", a presentation image titled "Apresentação dia 2024-03-18 at 11.43.44.jpeg", "Requisitos", "Pesquisa sobre os painéis solares", "Tags de divulgação", and "Apresentação -1".
- Sprint 3 (25/03 - 07/04):** Includes cards for "Reunião Inicial", "Stories", "Reunião para Alinhamento do projeto", "Desenvolvimento Código API geração de energia", and "Reunião Final".
- Sprint 4 (08/04 - 28/04):** Includes cards for "Reunião Inicial", a flowchart diagram, "Diagrama de Navegação", "Slides para apresentação final", "Desenvolvimento básico das paginas web", and "Documento de entrega parcial".
- Sprint 5 (29/04 - 27/05):** Includes cards for "Reunião Inicial", "Documento Parte 1", "Diagrama de Pacote", "Reunião de Alinhamento do Projeto", and "Documento Parte 2".

Each card shows a due date, assignees (represented by colored avatars), and a "Adicionar um cartão" (Add a card) button at the bottom. The interface also features a left sidebar with navigation options like "Quadros", "Membros", and "Configurações da Área de trabalho", and a top navigation bar with search and utility icons.

TECNOLOGIAS UTILIZADAS



SOLARPANELCONTROLLER (API)

```
25 @RequestMapping("/api/panels")
26 public class SolarPanelController implements SolarPanelControllerOpenApi {
27
28     private final SolarPanelRepository solarPanelRepository;
29     private final SolarPanelService solarPanelService;
30     private final SolarPanelStatsMapper solarPanelStatsMapper;
31     private final SolarPanelCollectionMapper solarPanelCollectionMapper;
32     private final SolarPanelUnmapper solarPanelUnmapper;
33
34     @GetMapping
35     @ResponseStatus(HttpStatus.OK)
36     public List<SolarPanelDTO> findAllSolarPanels() {
37         return solarPanelCollectionMapper.toCollectionModel(solarPanelRepository.findAll());
38     }
39
40
41     @GetMapping("/{id}")
42     @ResponseStatus(HttpStatus.OK)
43     public SolarPanel findSolarPanelById(@PathVariable Long id) {
44         return solarPanelService.getSolarPanelOrException(id);
45     }
46
47
48     @GetMapping("/{id}/stats")
49     @ResponseStatus(HttpStatus.OK)
50     public SolarPanelStatsDTO getSolarPanelStats(@PathVariable Long id, @RequestParam("kwh") double kwh) {
51         SolarPanel panel = solarPanelService.getSolarPanelOrException(id);
52         return solarPanelStatsMapper.toModel(panel, kwh);
53     }
54
55
56     @ApiResponses(value = @ApiResponse(responseCode = "201"))
57     @PostMapping
58     @ResponseStatus(HttpStatus.CREATED)
59     public SolarPanel createSolarPanel(@RequestBody @Valid SolarPanelInput solarPanelInput) {
60         var solarPanel = solarPanelUnmapper.toDomainObject(solarPanelInput);
61         return solarPanelService.save(solarPanel);
62     }
63
64 }
```

SOLAR PANEL SERVICE (DOMAIN)

```
EXPLORER
...
SolarPanelService.java 1 X

SMART-LEAF-BACK
> .mvn
> src
  > main
    > java \fatec\sp\gov\...
      > api
      > config
      > domain
        > exception
        > model
        > repository
        > service
          FotoStorageService.java
          SolarPanelImageS... 5
          SolarPanelService... 1
          infrastructure
          local_run
          SmartLeafApplication.java
          resources
          test
          target
          .gitignore
          LICENSE
          mvnw
          mvnw.cmd
          pom.xml
          README.md
    > SOURCE CONTROL
  > OPEN EDITORS
    X SolarPanelService.java s... 1
  > OUTLINE
  > TIMELINE
  > MAVEN
  > JAVA PROJECTS

42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82

public StatsDTO getSolarPanelStats(SolarPanel solarPanel, double kwh) {
    StatsDTO solarPanelStats = new StatsDTO();

    var solarPanelPrice = solarPanel.getPrice();
    var maximumPower = solarPanel.getMaximumPower();
    var efficiencyPercentage = solarPanel.getEfficiency();

    var panelsNeeded = getPanelsNeeded(maximumPower, efficiencyPercentage, kwh);
    var estimatedPrice = getEstimatedPrice(panelsNeeded, solarPanelPrice);
    var returnOfInvestment = getReturnOfInvestment(estimatedPrice, getSolarPanelMonthlyEnergy(
        maximumPower, efficiencyPercentage
    ));

    solarPanelStats.setPanelsNedeed(panelsNeeded);
    solarPanelStats.setEstimatedPrice(estimatedPrice);
    solarPanelStats.setReturnOfInvestment(returnOfInvestment);

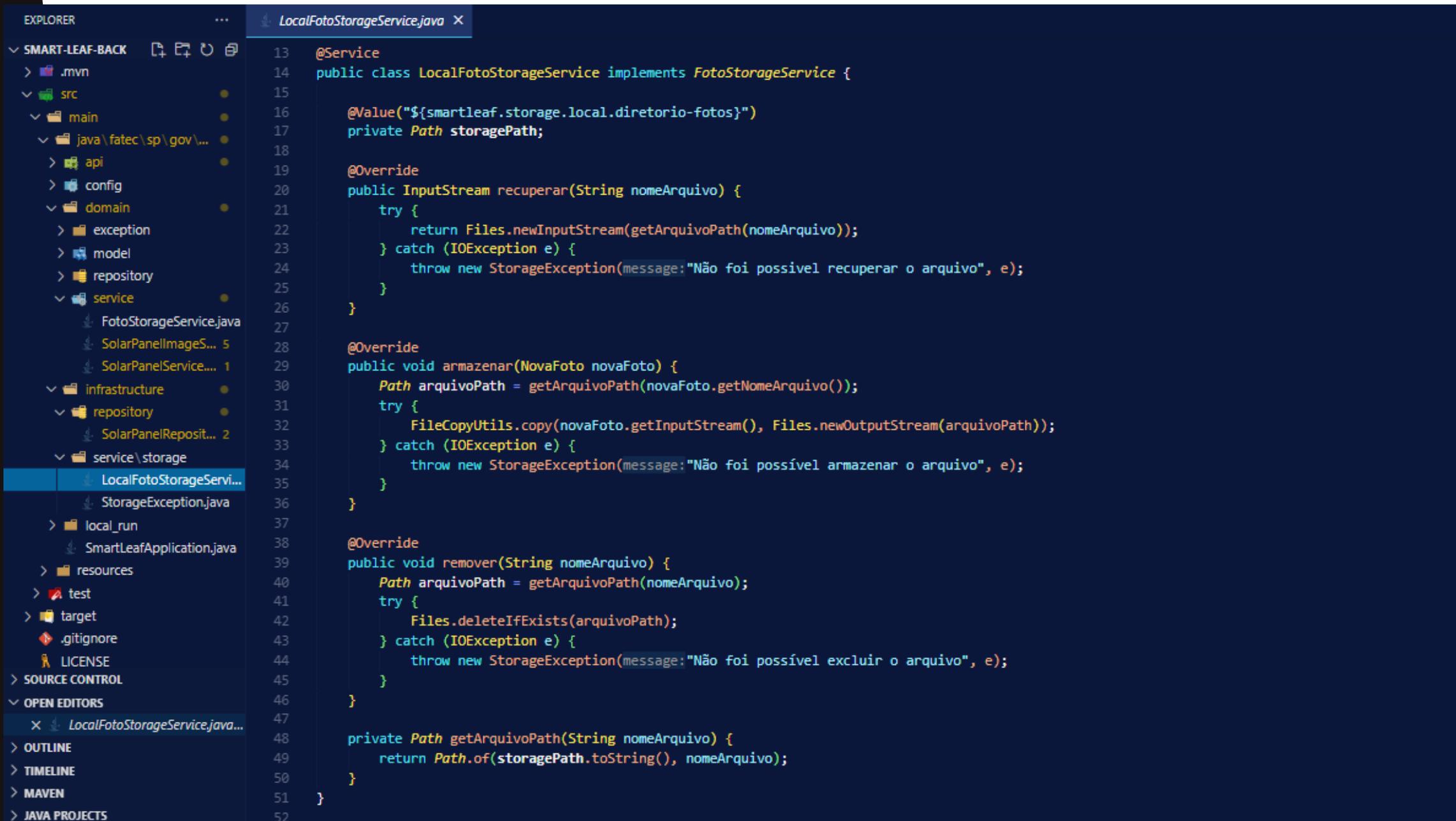
    return solarPanelStats;
}

@Transactional
public SolarPanel update(Long id, SolarPanelInput solarPanelInput) {
    var currentSolarPanel = getSolarPanelOrException(id);
    solarPanelUnmapper.copyToDomainObject(solarPanelInput, currentSolarPanel);
    return solarPanelRepository.save(currentSolarPanel);
}

@Transactional
public void delete(Long id) {
    try {
        solarPanelRepository.deleteById(id);
    } catch (DataIntegrityViolationException e) {
        solarPanelImageService.delete(id);
        solarPanelRepository.flush();
        solarPanelRepository.deleteById(id);
    }
}

public SolarPanel getSolarPanelOrException(Long id) {
    return solarPanelRepository.findById(id)
        .orElseThrow(() -> new RuntimeException("Solar Panel not found"));
}
```

LOCALFOTOSTORAGESERVICE (INFRASTRUCTURE)



```
13 @Service
14 public class LocalFotoStorageService implements FotoStorageService {
15
16     @Value("${smartleaf.storage.local.diretorio-fotos}")
17     private Path storagePath;
18
19     @Override
20     public InputStream recuperar(String nomeArquivo) {
21         try {
22             return Files.newInputStream(getArquivoPath(nomeArquivo));
23         } catch (IOException e) {
24             throw new StorageException(message:"Não foi possível recuperar o arquivo", e);
25         }
26     }
27
28     @Override
29     public void armazenar(NovaFoto novaFoto) {
30         Path arquivoPath = getArquivoPath(novaFoto.getNomeArquivo());
31         try {
32             FileCopyUtils.copy(novaFoto.getInputStream(), Files.newOutputStream(arquivoPath));
33         } catch (IOException e) {
34             throw new StorageException(message:"Não foi possível armazenar o arquivo", e);
35         }
36     }
37
38     @Override
39     public void remover(String nomeArquivo) {
40         Path arquivoPath = getArquivoPath(nomeArquivo);
41         try {
42             Files.deleteIfExists(arquivoPath);
43         } catch (IOException e) {
44             throw new StorageException(message:"Não foi possível excluir o arquivo", e);
45         }
46     }
47
48     private Path getArquivoPath(String nomeArquivo) {
49         return Path.of(storagePath.toString(), nomeArquivo);
50     }
51 }
52
```

OBRIGADO

