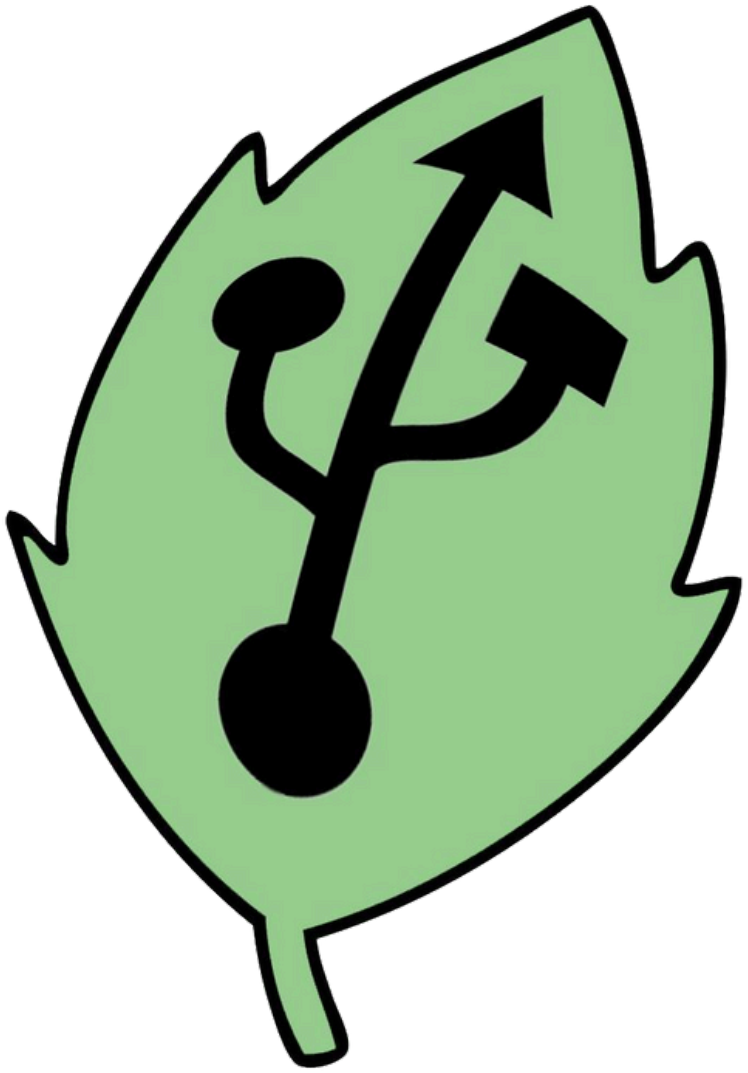


SMART LEAF

SOBRE O PROJETO

Desenvolver um software de padronização e interpretação de prontuários médicos entre diferentes especialidades

- Criação e armazenamento de prontuários médicos digitalizados
- Interpretação automatizada de dados clínicos via API de IA (StackpotIA)
- Padronização de registros médicos
- Geração de relatórios e exportação em PDF
- Interface intuitiva e segura, desenvolvida com Java Spring Boot, MongoDB e HTML/CSS/JS

ODS


3
SAÚDE E
BEM-ESTAR


9
INDÚSTRIA,
INOVAÇÃO E
INFRAESTRUTURA


10
REDUÇÃO DAS
DESIGUALDADES

PROBLEMÁTICA

- Prontuários médicos sem padronização, com registros inconsistentes entre profissionais e especialidades.
- Dificuldade na interpretação e análise dos dados clínicos, gerando retrabalho e risco de erro.
- Tempo excessivo gasto em documentação manual, reduzindo a produtividade dos profissionais de saúde.
- Ausência de ferramentas inteligentes para transformar dados clínicos em informações úteis para médicos.

CANVAS


Lean Canvas
01 Problema
Prontuários médicos escritos de forma diferente em cada hospital/médico, dificultando padronização.
Dificuldade de outros médicos compreenderem prontuários fora de sua especialidade.
Falta de integração entre sistema hospitalar e apoio inteligente para interpretação de dados.
02 Solução
Plataforma web e app integrados a banco hospitalar.
IA para processar prontuários e adaptá-los conforme especialidade do médico.
Geração de relatórios em PDF padronizados.
Histórico de alterações acessível a diferentes médicos.
03 Métricas-chave
Número de prontuários processados pela IA.
Tempo médio de acesso à informação pelo médico.
Taxa de adoção do sistema por médicos/hospitais.
Redução de erros de interpretação médica.
04 Proposta de Valor Única
"Um sistema inteligente que gera e adapta prontuários médicos de forma padronizada, auxiliando médicos a compreenderem informações com base em sua especialidade e melhorando a comunicação no atendimento."
05 Vantagem Injusta
IA treinada especificamente para linguagem médica em português.
Integração direta com bancos hospitalares (evita retrabalho de cadastro).
Padronização dos prontuários independente do hospital.
06 Canais
Integração direta com sistemas hospitalares (via API/BD).
Aplicativo mobile para médicos.
Web app para hospitais e administradores.
07 Segmento de Clientes
Hospitais e clínicas.
Médicos de diferentes especialidades.
Gestores de saúde que precisam de relatórios padronizados.
08 Estrutura de Custo
Desenvolvimento de software (Java backend, web frontend, app mobile).
Infraestrutura em nuvem (banco de dados, servidores de IA).
Treinamento e atualização do modelo de IA.
Suporte e manutenção.
09 Fonte de Receita
Licenciamento para hospitais.
Assinatura SaaS (Software como Serviço).
Serviços de integração e personalização para cada hospital.

REQUISITOS FUNCIONAIS

- Anotações SOAP
- ExportaçãoProntuários
- LoginSeguro
- Gestão de Médicos
- EspecialidadeIA
- Histórico de Versões
- Edição de Prontuários
- Comparação Diff
- Colaboração Médica
- Integração FHIRHL7
- App Mobile Offline
- Relatórios de Estatísticas
- Recuperação de Senha

REQUISITOS NÃO FUNCIONAIS

- Desempenho de 5s
- Disponibilidade de 99%
- Autenticação Segura
- Escalabilidade Container
- ArquiteturaMVC
- Interface Responsiva
- Backup Restore
- SuporteMultiplataforma
- Conformidade LGPD

DIAGRAMA DE CASO DE USO

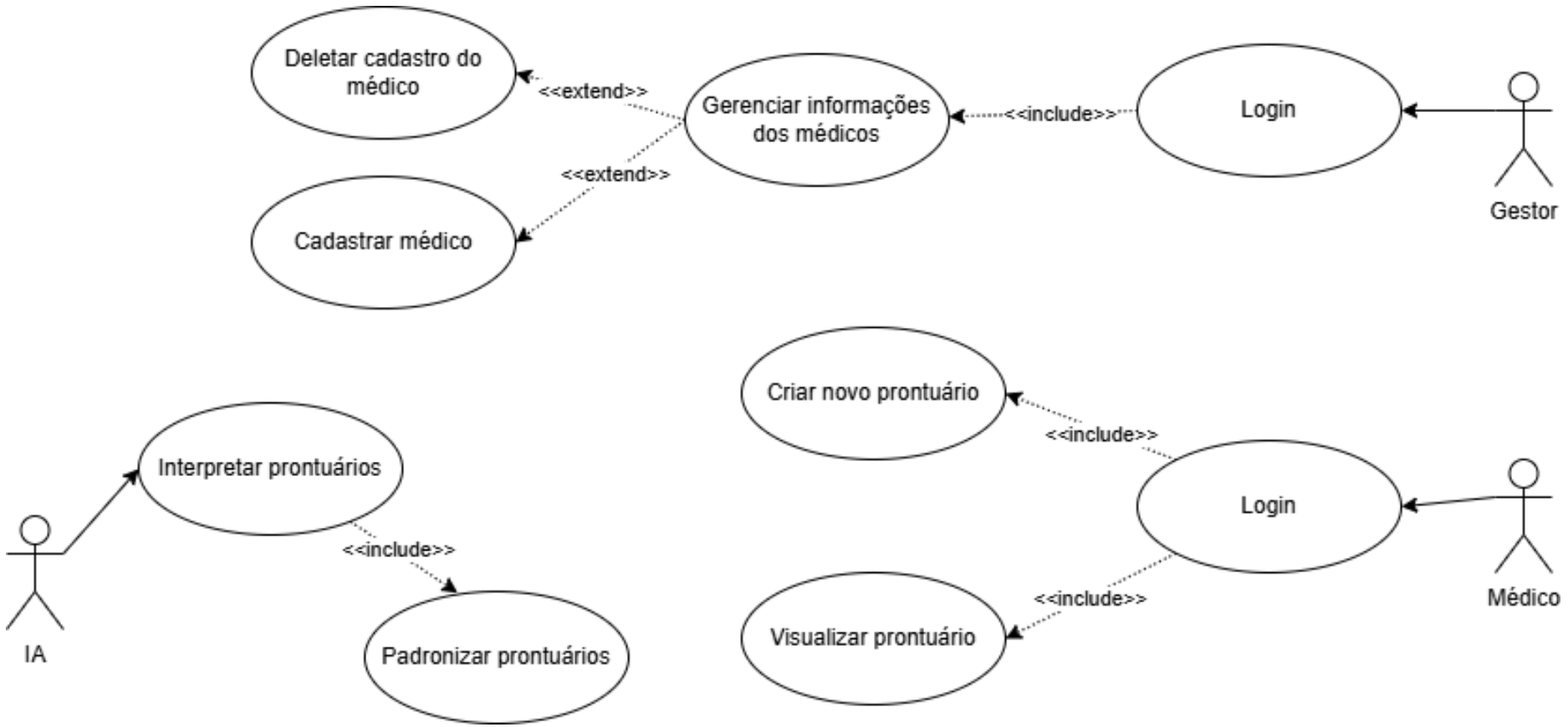
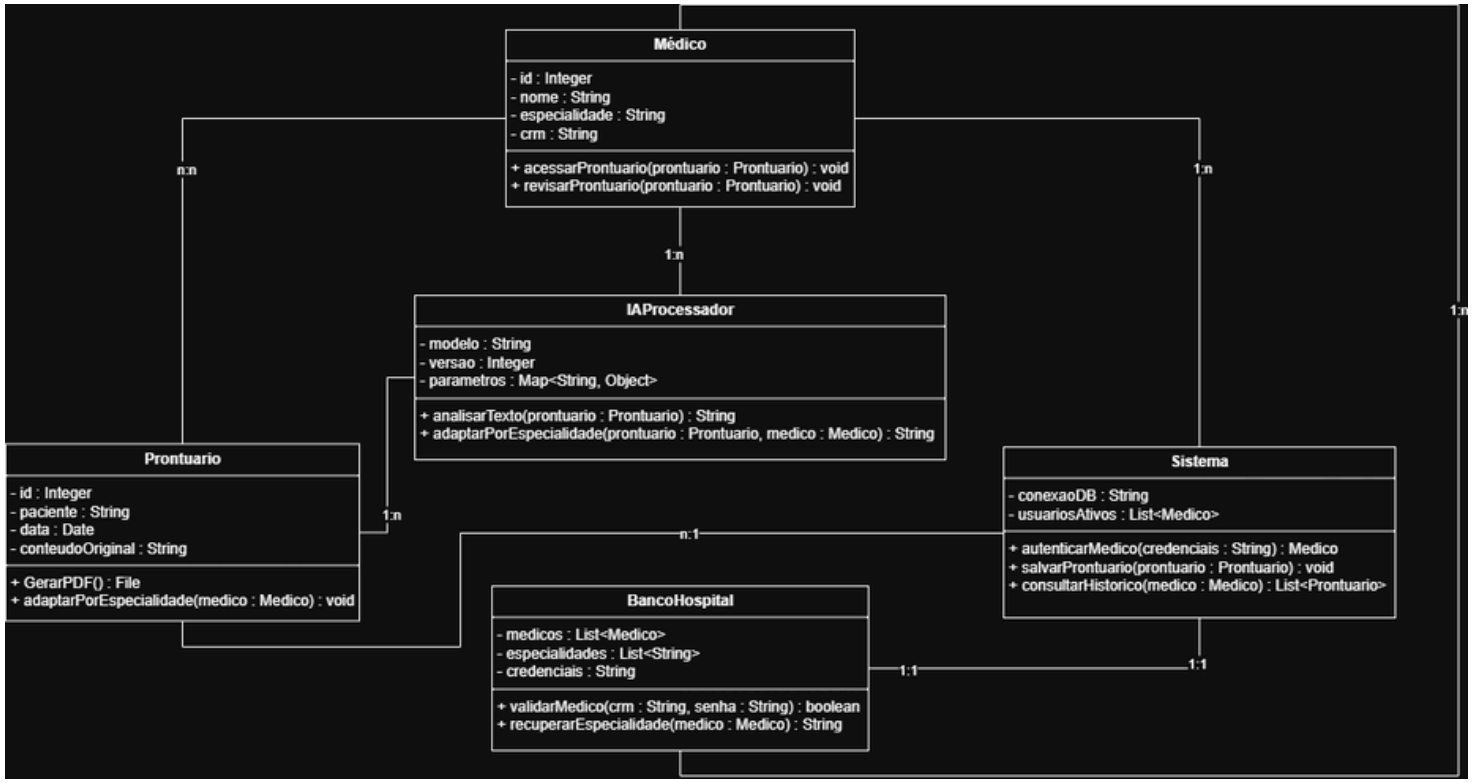
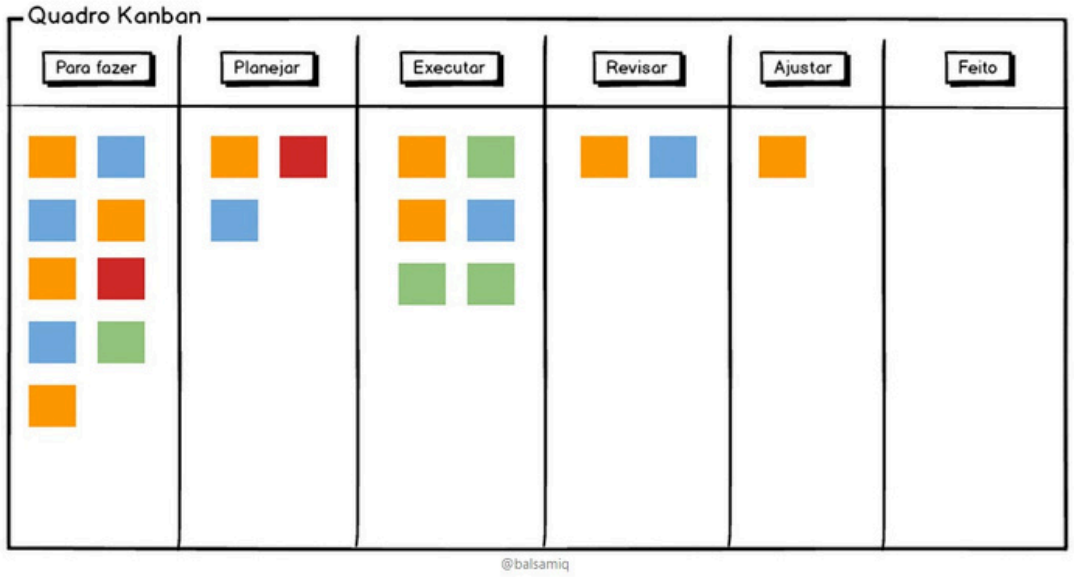


DIAGRAMA DE CLASSE



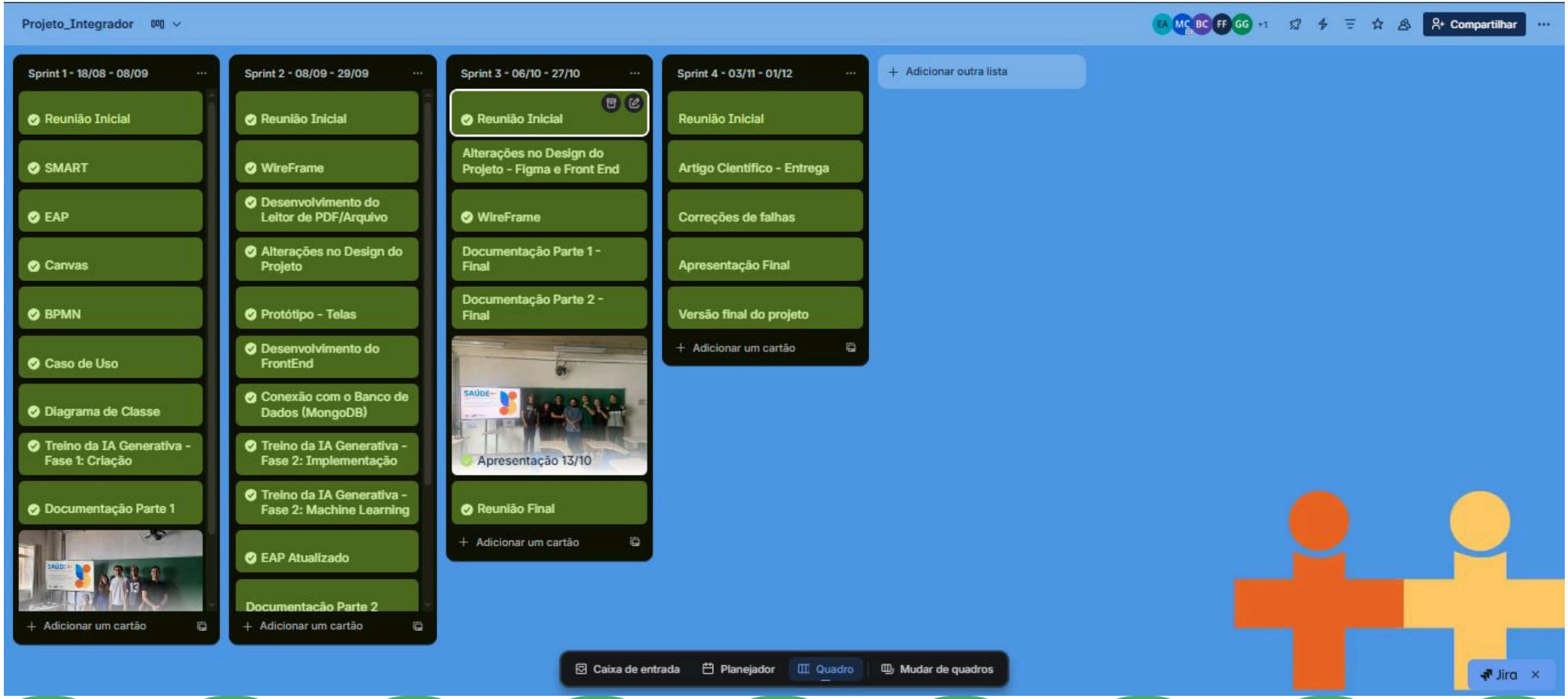
MÉTODO DE DESENVOLVIMENTO

Kanban



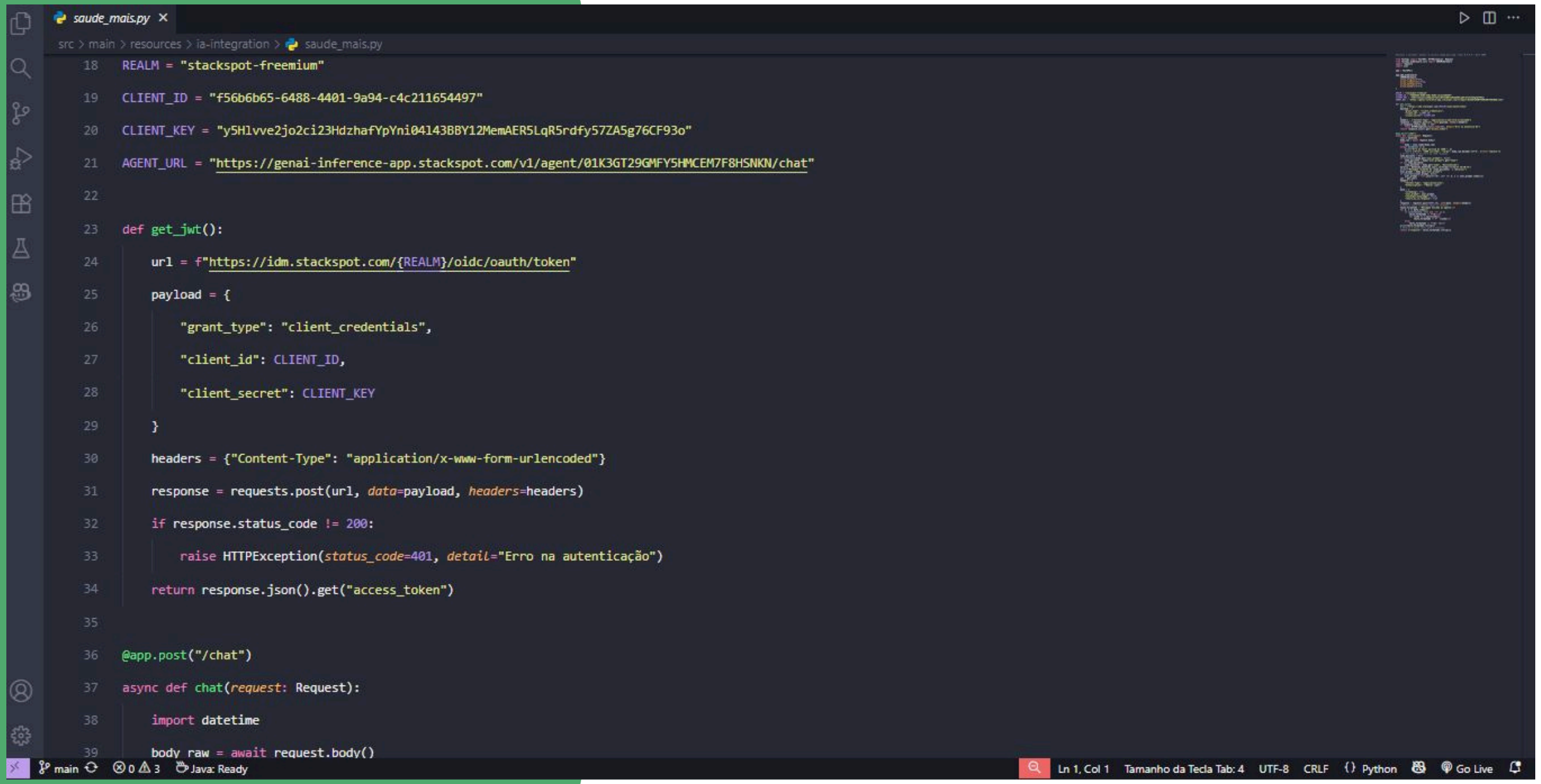
Scrum

PLATAFORMA TRELLO

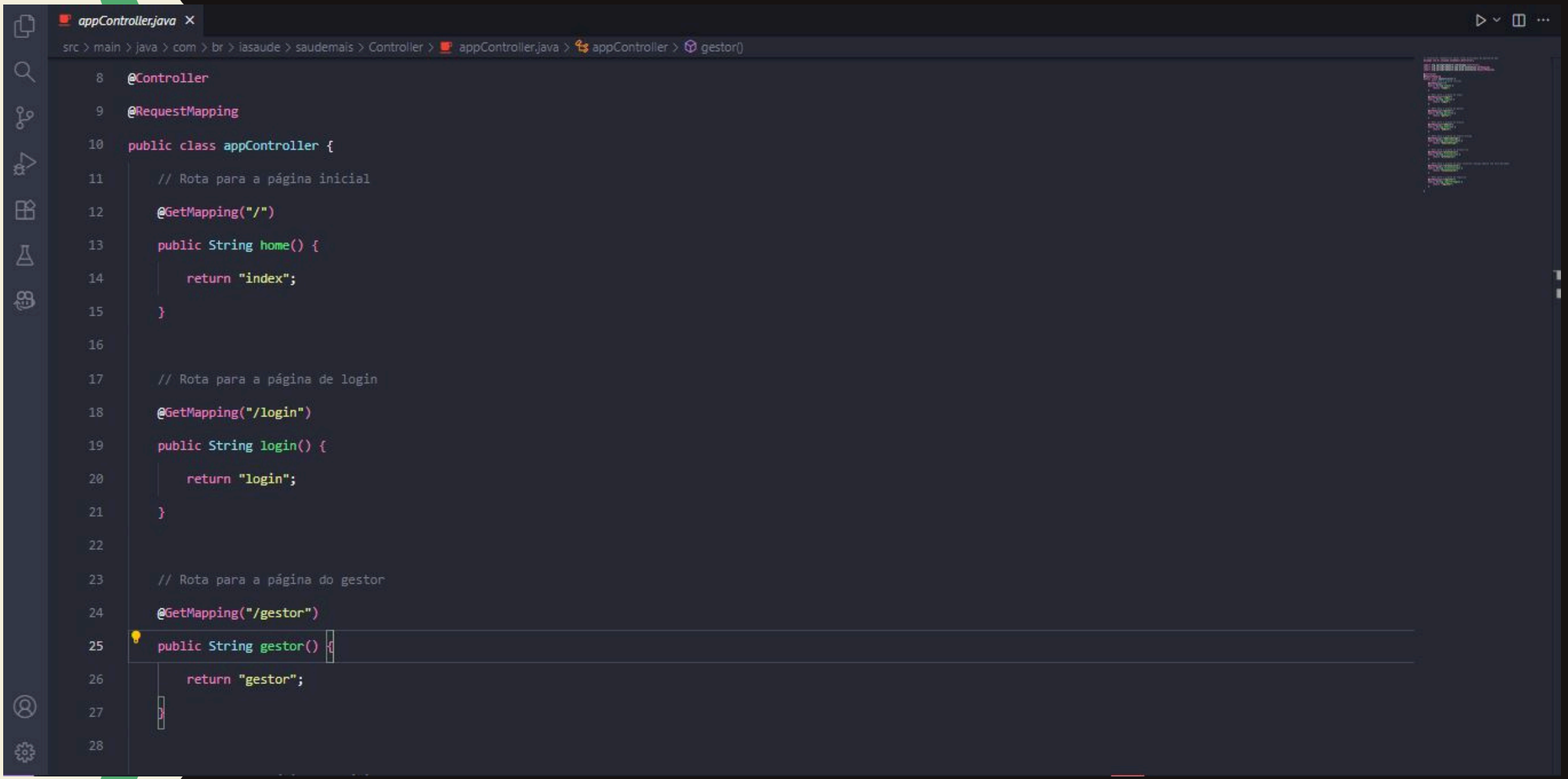


TECNOLOGIAS UTILIZADAS

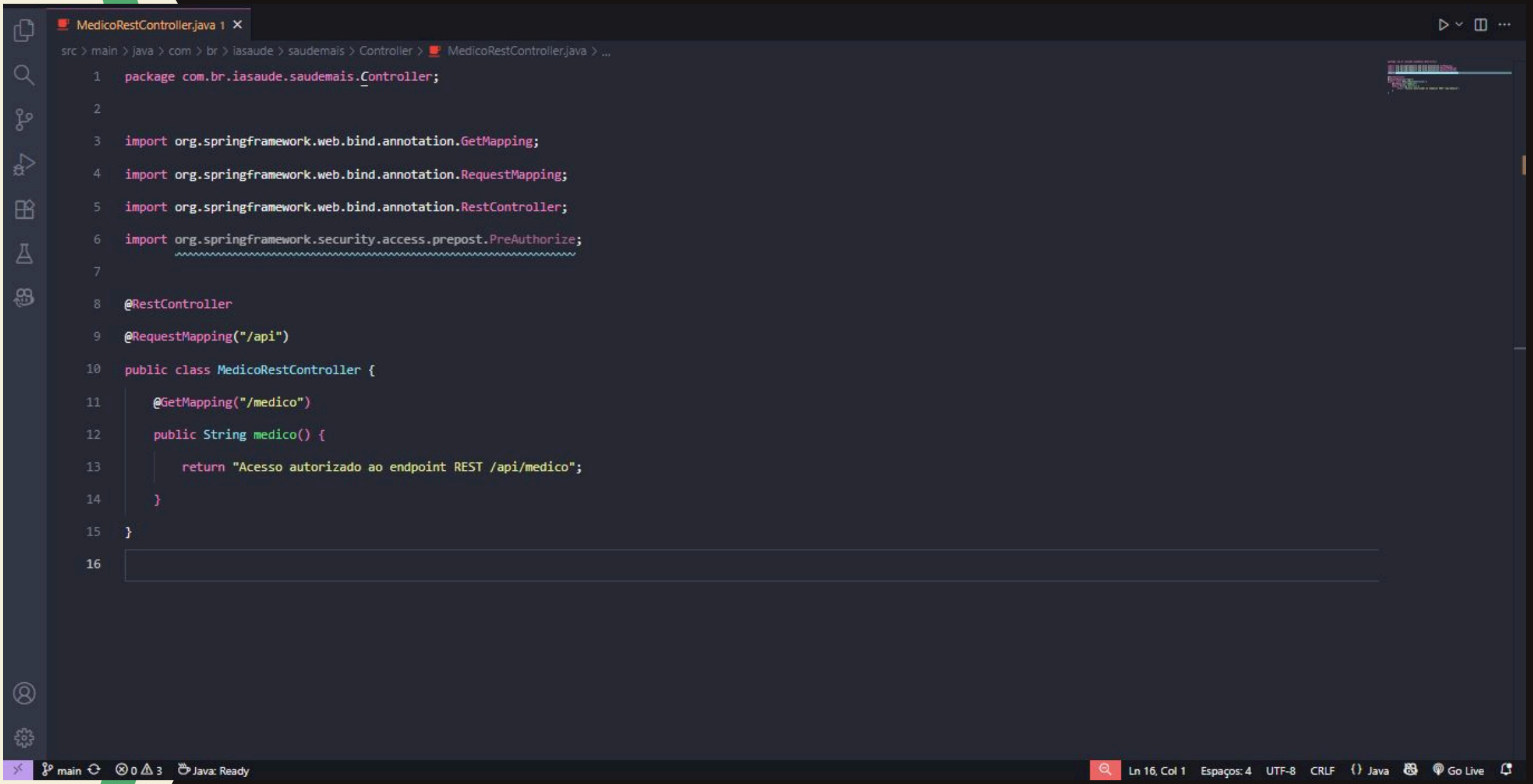
saude_mais.py (AI API)



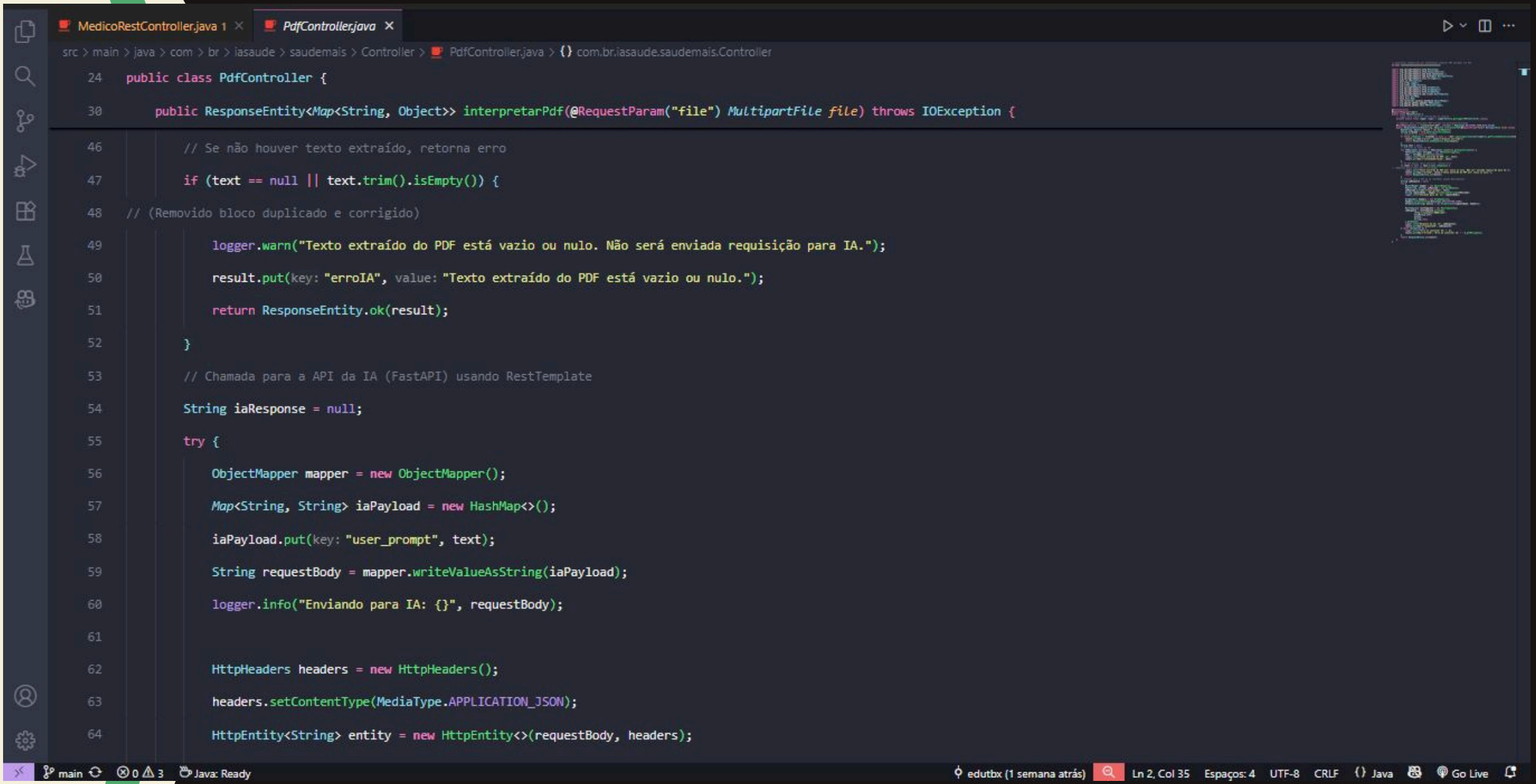
Appcontroller (model)



Medicorestcontroller



Pdfcontroller



Profilecontroller

MedicoRestController.java 1 | ProfileController.java

src > main > java > com > br > iasaude > saudemais > Controller > ProfileController.java > {} com.br.iasaude.saudemais.Controller

```java
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.Arrays;
import java.util.Map;

@RestController
public class ProfileController {
    private final Environment env;

    public ProfileController(Environment env) {
        this.env = env;
    }

    @GetMapping("/api/profile")
    public Map<String, Object> getProfile() {
        String[] profiles = env.getActiveProfiles();
        return Map.of(
            k1: "activeProfiles", profiles,
            k2: "dev", Arrays.asList(profiles).contains(o: "dev")
        );
    }
}
```

main | 0 ⚠ 3 | Java: Ready | edutbx (1 semana atrás) | Ln 1, Col 35 | Espaços: 4 | UTF-8 | CRLF | Java | Go Live

OBRIGADO