



TALENTSCOPE

PLATAFORMA DE TRIAGEM DE CURRÍCULOS



SOBRE NÓS

A TALENTSCOPE É UMA PLATAFORMA DE TRIAGEM DE CURRÍCULOS. COM ELA, O RECRUTADOR TERÁ SEU TEMPO OTIMIZADO, UMA VEZ QUE A PRIMEIRA SELEÇÃO DOS CURRÍCULOS SERÁ FEITA POR NÓS, POR MEIO DE UMA COMPARAÇÃO DOS REQUISITOS PASSADOS PELA EMPRESA COM AS INFORMAÇÕES DO CURRÍCULO DO CANDIDATO. CHEGAMOS PARA OTIMIZAR O SEU TEMPO GARANTINDO QUALIDADE.

MERCADO ATUAL

1

TRIAGEM DE CURRÍCULO

Atualmente recrutadores investem uma média de 16 horas mensais na triagem de currículos, sendo essa uma análise superficial e com grandes riscos de falha.

2

FEEDBACK DEVOLUTIVO

Grande parte das empresas não fornecem feedback alegando que este não é obrigatório, e que a volumetria é muito grande, além de não trazer nenhum retorno para a empresa.

PÚBLICO ALVO

O NOSSO PÚBLICO ALVO INCLUI TODOS OS PROFISSIONAIS DE RECURSOS HUMANOS, COM FOCO NA ÁREA DE ATRAÇÃO DE TALENTOS. PODEM USAR OS NOSSOS SERVIÇOS EMPRESAS DE PEQUENO A GRANDE PORTE, DESDE MICRONEGÓCIOS A MULTINACIONAIS, ALÉM DE AUTÔNOMOS. CONSIDERANDO O MERCADO BRASILEIRO, TEMOS HOJE POTENCIAL DE ATUAÇÃO EM MAIS DE 20 MILHÕES DE EMPRESAS, POIS GRANDE PARTE DELAS POSSUEM O SETOR DE RH.

SOLUÇÃO

TRIAGEM

COM O APOIO DO CHATGPT, O NOSSO APLICATIVO, ASSIM QUE RECEBER O CURRÍCULO DO CANDIDATO, FARÁ UMA ANÁLISE DO CURRÍCULO E POR COMPARAÇÃO COM OS REQUISITOS DEFINIDOS PELA EMPRESA PARA AQUELA VAGA, DEFINIRÁ SE AQUELE CANDIDATO ESTÁ ACIMA OU ABAIXO DA NOTA DE CORTE.

FEEDBACK

UMA VEZ REPROVADO NA VAGA, BASEADO NOS REQUISITOS PASSADOS PELO RECRUTADOR, O APLICATIVO FORMULARÁ UM FEEDBACK HUMANIZADO, PARA QUE O CANDIDATO SAIBA EXATAMENTE POR QUAL MOTIVO HOVE A REPROVAÇÃO, E PODENDO ASSIM TRABALHAR NESTES PONTOS.

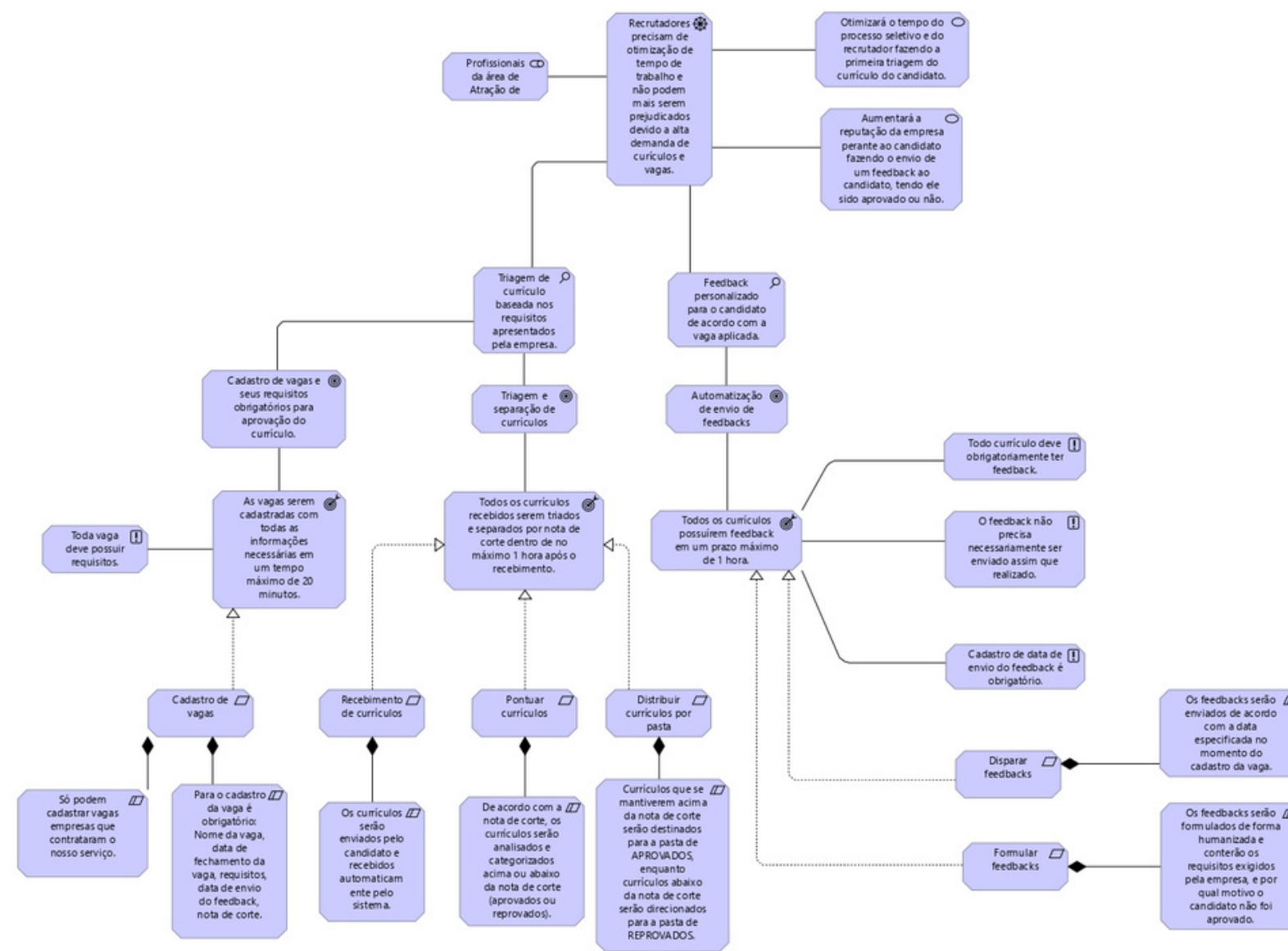
CONCORRENTES

NÃO FORAM ENCONTRADAS PLATAFORMAS NO MERCADO QUE OFERECEM UMA SOLUÇÃO IDÊNTICA A TALENTSCOPE. PORÉM ALGUMAS POSSUEM SIMILARIDADES:

- Zoho Recruiter
- Trackstar Hire
- HireBeat
- iCIMS
- Taleo
- Freshteam
- Vervoe

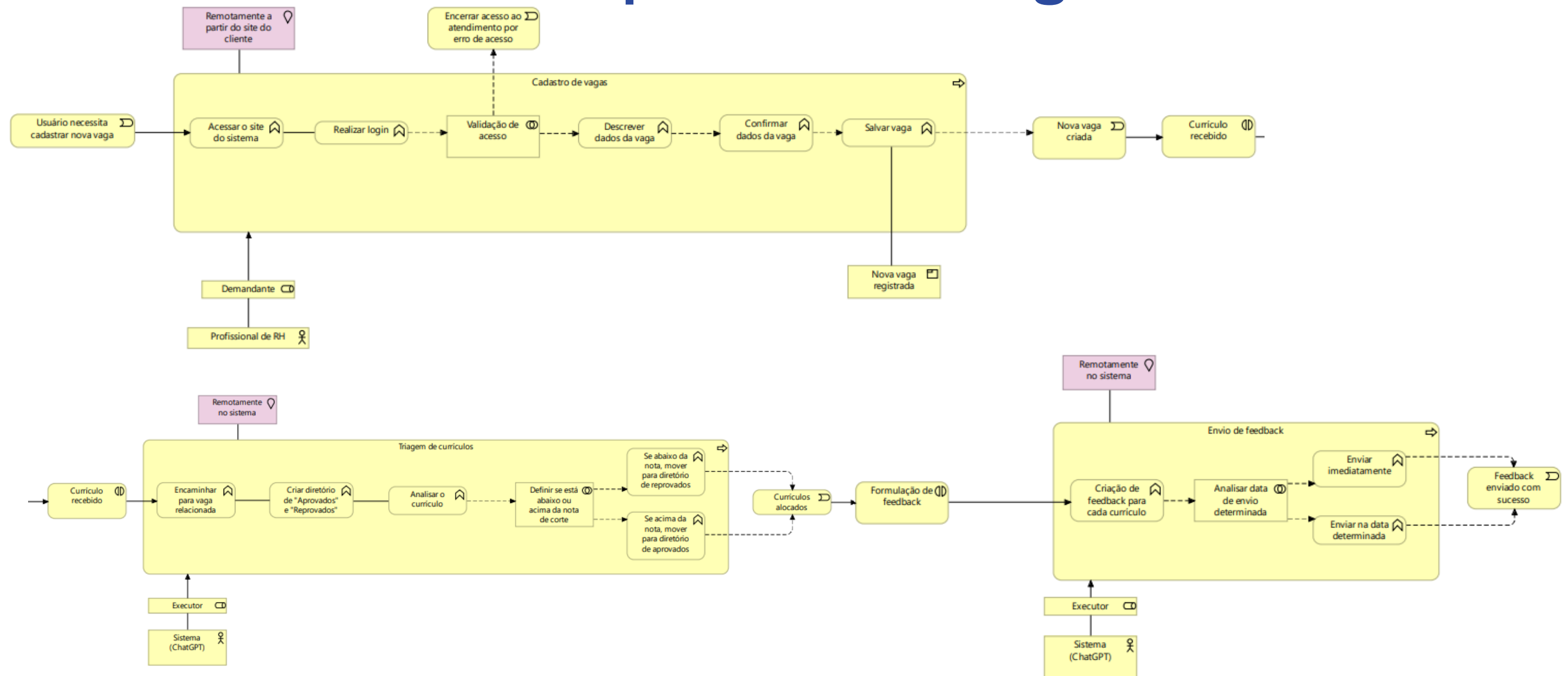
ARQUITETURA DA SOLUÇÃO

Visão da arquitetura



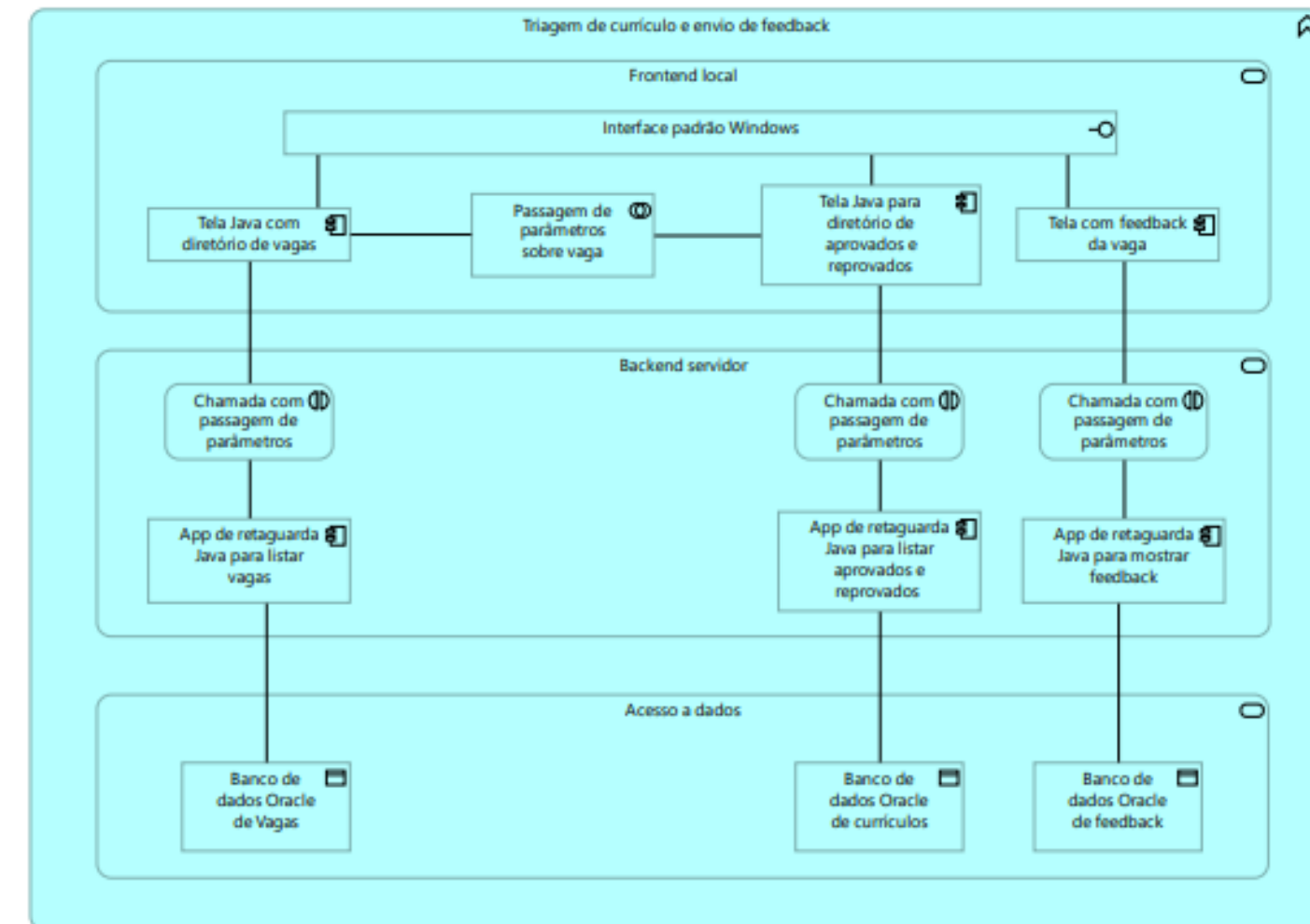
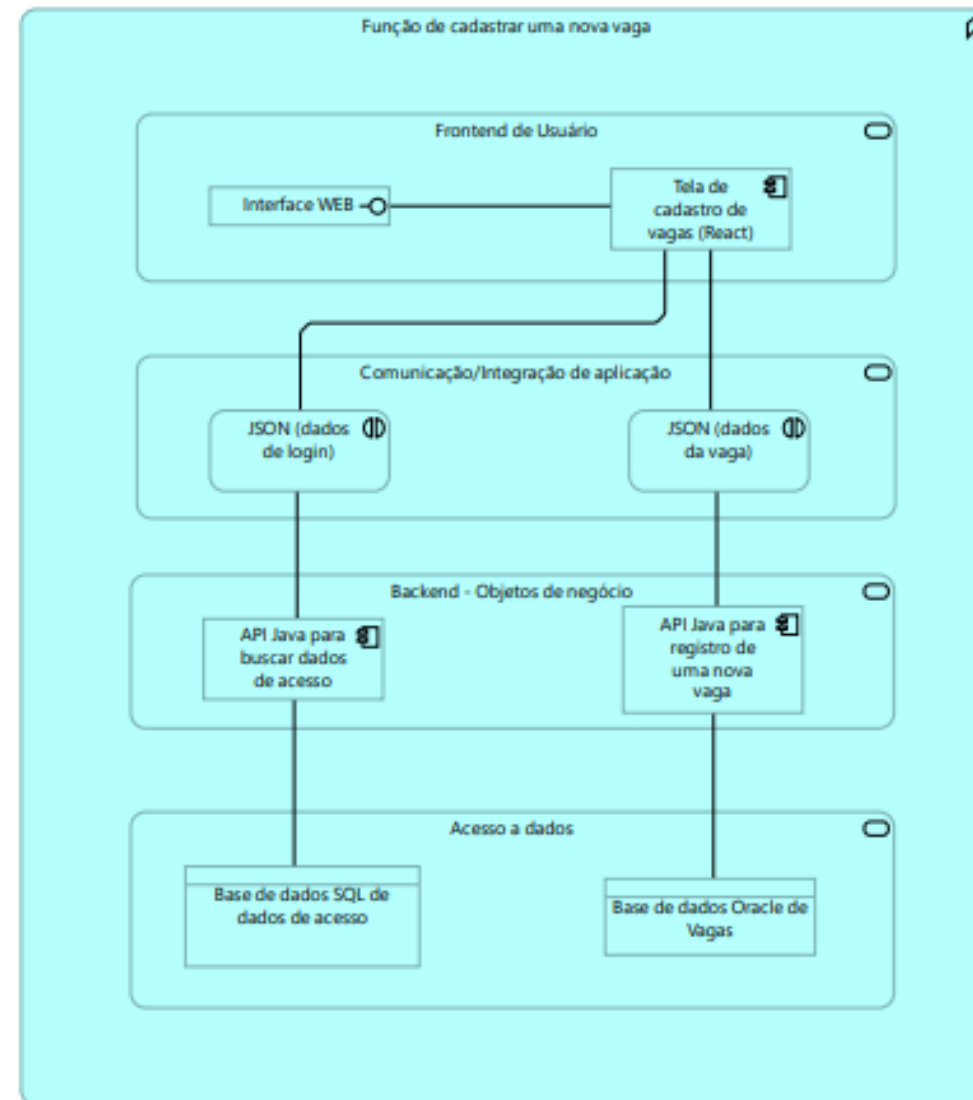
ARQUITETURA DA SOLUÇÃO

Arquitetura de negócio



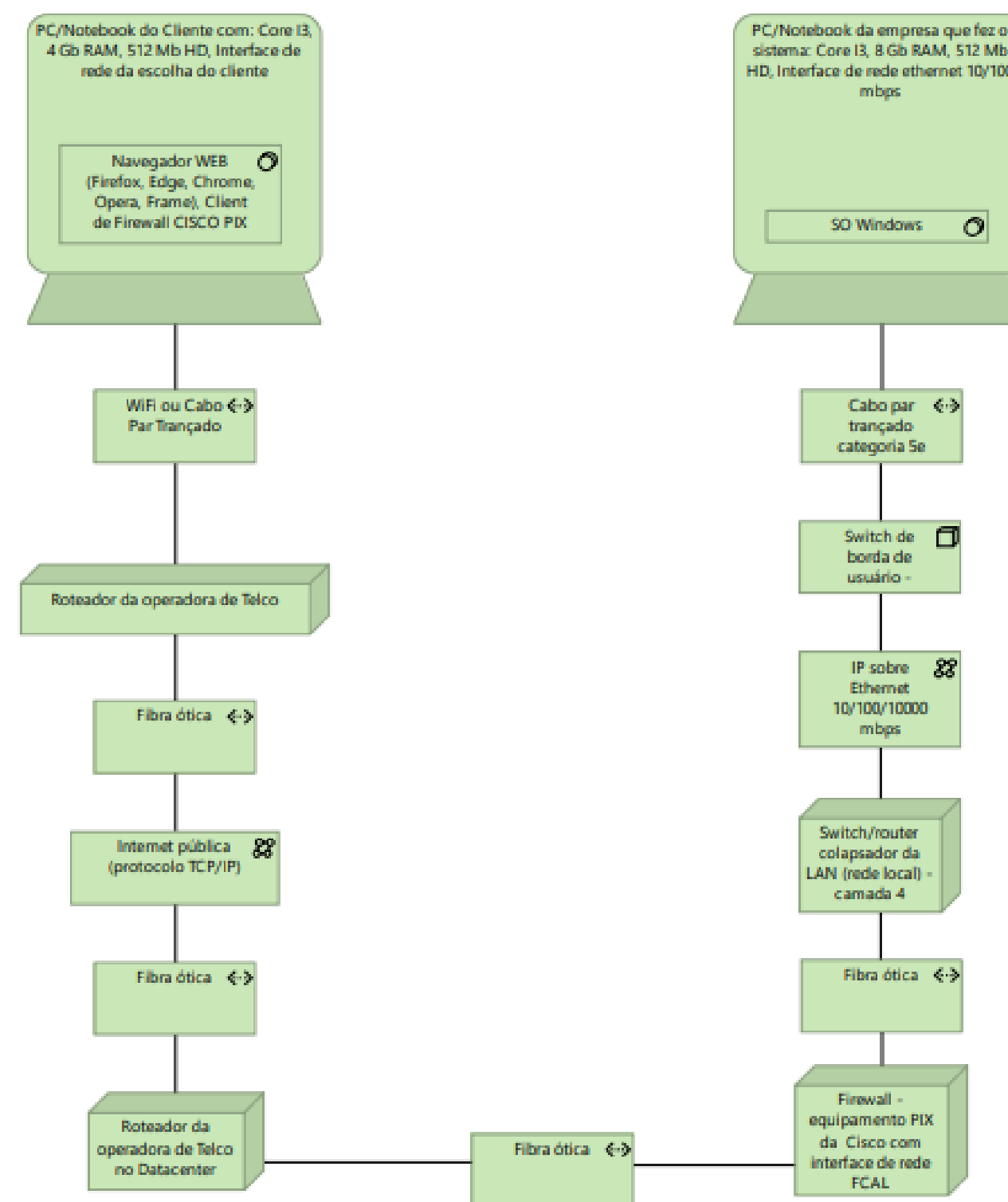
ARQUITETURA DA SOLUÇÃO

Arquitetura de Sistemas



ARQUITETURA DA SOLUÇÃO

Arquitetura de Tecnologia



TECNOLOGIAS UTILIZADAS

BACK END

Para o desenvolvimento do Backend, utilizaremos as linguagens de programação Python e Java, Frameworks Spring, Hibernate e JSON Web Token, e Ferramentas de desenvolvimento Eclipse e VSCode. Utilizaremos também a API do ChatGPT.

FRONT END

Para o desenvolvimento do Frontend, utilizaremos a linguagem de programação React-native nas ferramentas de desenvolvimento VSCode e Android Emulator, e usaremos as extensões ESLint, Git Lensa, Import Cost, ES7, e React Native Tools.

TECNOLOGIAS UTILIZADAS

BANCO DE DADOS

Para o banco de dados utilizaremos o Oracle SQL, com as ferramentas Oracle SQL Developer e Oracle Datamodeler, e para testes utilizaremos a plataforma Banco de Dados H2.

ANÁLISE DE DADOS

Para realizar análise de dados utilizaremos a linguagem de programação Python, frameworks Pandas e NumPy, com as ferramentas de desenvolvimento Google Colab e Pycharm.

CRUDS

Visual Studio Code interface showing the `VagaController.java` file in a project named `TalentScope-backend`. The code implements CRUD operations for a `Vaga` entity using Spring Data JPA and Spring MVC.

```
31 VagaRepository repository;
32
33 @GetMapping
34 public List<Vaga> getAll(){
35     return repository.findAll();
36 }
37
38 @PostMapping
39 public ResponseEntity<Vaga> create(@RequestBody @Valid Vaga vaga){
40     log.info("Cadastrando vaga: " + vaga);
41
42     repository.save(vaga);
43     return ResponseEntity.status(HttpStatus.CREATED).body(vaga);
44 }
45
46 @GetMapping("/{id}")
47 public ResponseEntity<Vaga> findById(@PathVariable Long id){
48     log.info("Buscando vaga com id " + id);
49     var vaga = getVaga(id);
50     return ResponseEntity.ok(vaga);
51 }
52
53 @DeleteMapping("/{id}")
54 public ResponseEntity<Vaga> delete(@PathVariable Long id){
55     log.info("Apagando vaga com o id " + id);
56     var vaga = getVaga(id);
57     repository.delete(vaga);
58     return ResponseEntity.noContent().build();
59 }
60
61 @PutMapping("/{id}")
62 public ResponseEntity<Vaga> update(@PathVariable Long id, @RequestBody @Valid Vaga vaga){
63     log.info("Atualizando vaga com o id " + id);
64     getVaga(id);
65     vaga.setId(id);
66     repository.save(vaga);
67     return ResponseEntity.ok(vaga);
68 }
69
70 private Vaga getVaga(Long id) {
71     return repository.findById(id)
72         .orElseThrow(() -> new RestNotFoundException("vaga não encontrada"));
73 }
74 }
```

The Explorer sidebar shows the project structure, and the Outline sidebar shows the class structure.

CRUDS

Atividades Insomnia dom, 21 de mai, 20:18

Insomnia

Application Edit View Window Tools Help

ISA MARY's Personal Project / TaletScope master

ISA MARY KUSUKI YABIKU

No Environment Cookies

Filter

DEL DELETE

GET GET-BY-ID

POST CREATE

GET GET-ALL

Vaga

PUT UPDATE

DEL DELETE

POST CREATE

GET GET-BY-ID

GET GET-ALL

Nivel Permissao

Feedback

Curriculo

POST New Request

PUT UPDATE

DEL DELETE

POST localhost:8080/taletScope/vagas Send 201 Created 195 ms 411 B 11 Minutes Ago

JSON Auth Query Headers 1 Docs

```
1 {
2   "nome": "Analista de Sistemas I",
3   "descricaoCargo": "Desenvolver sistemas em Java",
4   "salario": 5000.00,
5   "dtAbertura": "2023-05-15 14:00:00",
6   "dtEncerramento": "2023-07-22 17:00:00",
7   "dtProgEnvioFeedback": "2023-05-23 16:00:00",
8   "usuario": {
9     "id": 2
10  },
11  "habilidades": [
12    {
13      "id": 1
14    },
15    {
16      "id": 2
17    }
18  ]
19 }
```

Preview Headers 3 Cookies Timeline

```
1 {
2   "id": 1,
3   "nome": "Analista de Sistemas I",
4   "descricaoCargo": "Desenvolver sistemas em Java",
5   "salario": 5000.00,
6   "dtAbertura": "2023-05-15 14:00:00",
7   "dtEncerramento": "2023-07-22 17:00:00",
8   "dtProgEnvioFeedback": "2023-05-23 16:00:00",
9   "usuario": {
10     "id": 2,
11     "nome": null,
12     "email": null,
13     "senha": null,
14     "status": null,
15     "dtCriacao": null,
16     "dtAlteracao": null,
17     "nivelPermissao": null
18   },
19   "habilidades": [
20     {
21       "id": 1,
22       "nome": null
23     },
24     {
25       "id": 2,
26       "nome": null
27     }
28   ]
29 }
```

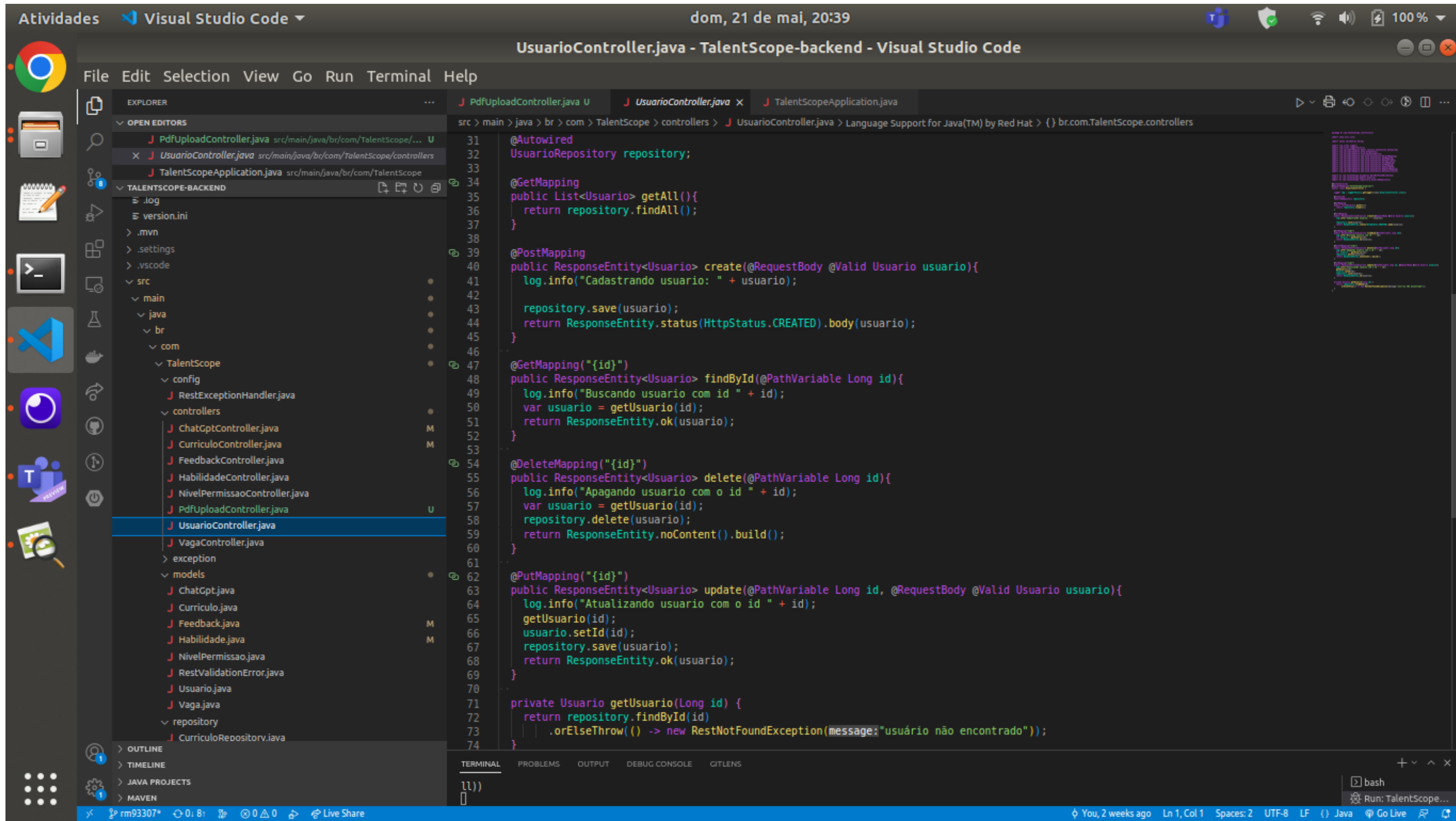
Beautify JSON

\$.store.books[*].author

Preferences

Made with ❤ by Kong

CRUDS



Atividades Visual Studio Code dom, 21 de mai, 20:39

UsuarioController.java - TalentScope-backend - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

OPEN EDITORS

TalentScope-backend

UserController.java

```
@Autowired
UsuarioRepository repository;

@GetMapping
public List<Usuario> getAll(){
    return repository.findAll();
}

@PostMapping
public ResponseEntity<Usuario> create(@RequestBody @Valid Usuario usuario){
    log.info("Cadastrando usuario: " + usuario);
    repository.save(usuario);
    return ResponseEntity.status(HttpStatus.CREATED).body(usuario);
}

@GetMapping("/{id}")
public ResponseEntity<Usuario> findById(@PathVariable Long id){
    log.info("Buscando usuario com id " + id);
    var usuario = getUsuario(id);
    return ResponseEntity.ok(usuario);
}

@DeleteMapping("/{id}")
public ResponseEntity<Usuario> delete(@PathVariable Long id){
    log.info("Apagando usuario com o id " + id);
    var usuario = getUsuario(id);
    repository.delete(usuario);
    return ResponseEntity.noContent().build();
}

@PutMapping("/{id}")
public ResponseEntity<Usuario> update(@PathVariable Long id, @RequestBody @Valid Usuario usuario){
    log.info("Atualizando usuario com o id " + id);
    getUsuario(id);
    usuario.setId(id);
    repository.save(usuario);
    return ResponseEntity.ok(usuario);
}

private Usuario getUsuario(Long id) {
    return repository.findById(id)
        .orElseThrow(() -> new RestNotFoundException(message:"usuário não encontrado"));
}
```

CRUDS

The screenshot displays the Insomnia application window. The top bar shows the system clock as 'dom, 21 de mai, 20:34' and the application name 'Insomnia'. The main interface is divided into several sections:

- Left Sidebar:** Contains a list of request collections. The 'Usuario' collection is expanded, showing a list of HTTP methods: GET, GET-BY-ID, POST, CREATE, GET, GET-ALL, PUT, UPDATE, DEL, DELETE, GET, GET-BY-ID, POST, CREATE, and GET, GET-ALL. The 'GET-BY-ID' method is selected.
- Top Bar:** Shows the current project 'ISA MARY's Personal Project / TaletScope' and the selected environment 'master'.
- Request Bar:** Displays the selected method 'GET' and the URL 'localhost:8080/talentScope/usuarios/2'. The 'Send' button is highlighted.
- Response Bar:** Shows the status '200 OK', response time '94.7 ms', and size '247 B'.
- Response Body:** Displays the JSON response in a preview format:

```
1 {
2   "id": 2,
3   "nome": "Rafael Ferreira",
4   "email": "rafael@plusofit.com",
5   "senha": "7894562",
6   "status": true,
7   "dtCriacao": "2023-05-15 10:08:02",
8   "dtAlteracao": null,
9   "nivelPermissao": {
10    "id": 2,
11    "nome": "Administrador",
12    "descricao": "Tem permissão total da aplicação"
13  }
14 }
```
- Bottom Bar:** Includes a 'Preferences' button and the text 'Made with ❤ by Kong'.

CRUDS

Visual Studio Code interface showing the implementation of CRUD operations in a Java Spring Boot application. The file `CurriculoController.java` is open, displaying the following code:

```
src > main > java > br > com > TalentScope > controllers > CurriculoController.java > Language Support for Java(TM) by Red Hat > {} br.com.TalentScope.controllers

return repository.findById(id);
}

@PostMapping("/feedback/{feedbackId}/curriculo")
public ResponseEntity<Curriculo> create(@PathVariable(value="feedbackId") Long feedbackId, @RequestBody @Valid Curriculo curriculo) {
    log.info("Cadastrando usuario: " + curriculo);

    Feedback feedback = feedbackRepository.findById(feedbackId)
        .orElseThrow(() -> new RestNotFoundException(message:"Feedback não encontrado"));
    curriculo.setFeedback(feedback);
    repository.save(curriculo);
    return ResponseEntity.status(HttpStatus.CREATED).body(curriculo);
}

@GetMapping("/curriculos/{id}")
public ResponseEntity<Curriculo> findById(@PathVariable Long id){
    log.info("Buscando curriculo com o id " + id);
    var curriculo = getCurriculo(id);
    return ResponseEntity.ok(curriculo);
}

@DeleteMapping("/curriculos/{id}")
public ResponseEntity<Curriculo> delete(@PathVariable Long id){
    log.info("Apagando curriculo com o id " + id);
    var curriculo = getCurriculo(id);
    repository.delete(curriculo);
    return ResponseEntity.noContent().build();
}

@PutMapping("/feedback/{feedbackId}/curriculo/{id}")
public ResponseEntity<Curriculo> update(@PathVariable Long id, @RequestBody @Valid Curriculo curriculo){
    log.info("Atualizando curriculo com o id " + id);
    getCurriculo(id);
    curriculo.setId(id);
    repository.save(curriculo);
    return ResponseEntity.ok(curriculo);
}

private Curriculo getCurriculo(Long id) {
    return repository.findById(id)
        .orElseThrow(() -> new RestNotFoundException(message:"curriculo não encontrado"));
}

}
```

The Explorer sidebar shows the project structure, including the `src/main/java/br/com/TalentScope/controllers` directory. The Terminal at the bottom shows the application running with the command `bash` and the output `1, dtAbertura=null, dtEncerramento=null, dtProgEnvioFeedback=null, usuario=null, habilidades=null, feedback=null`.

CRUDS

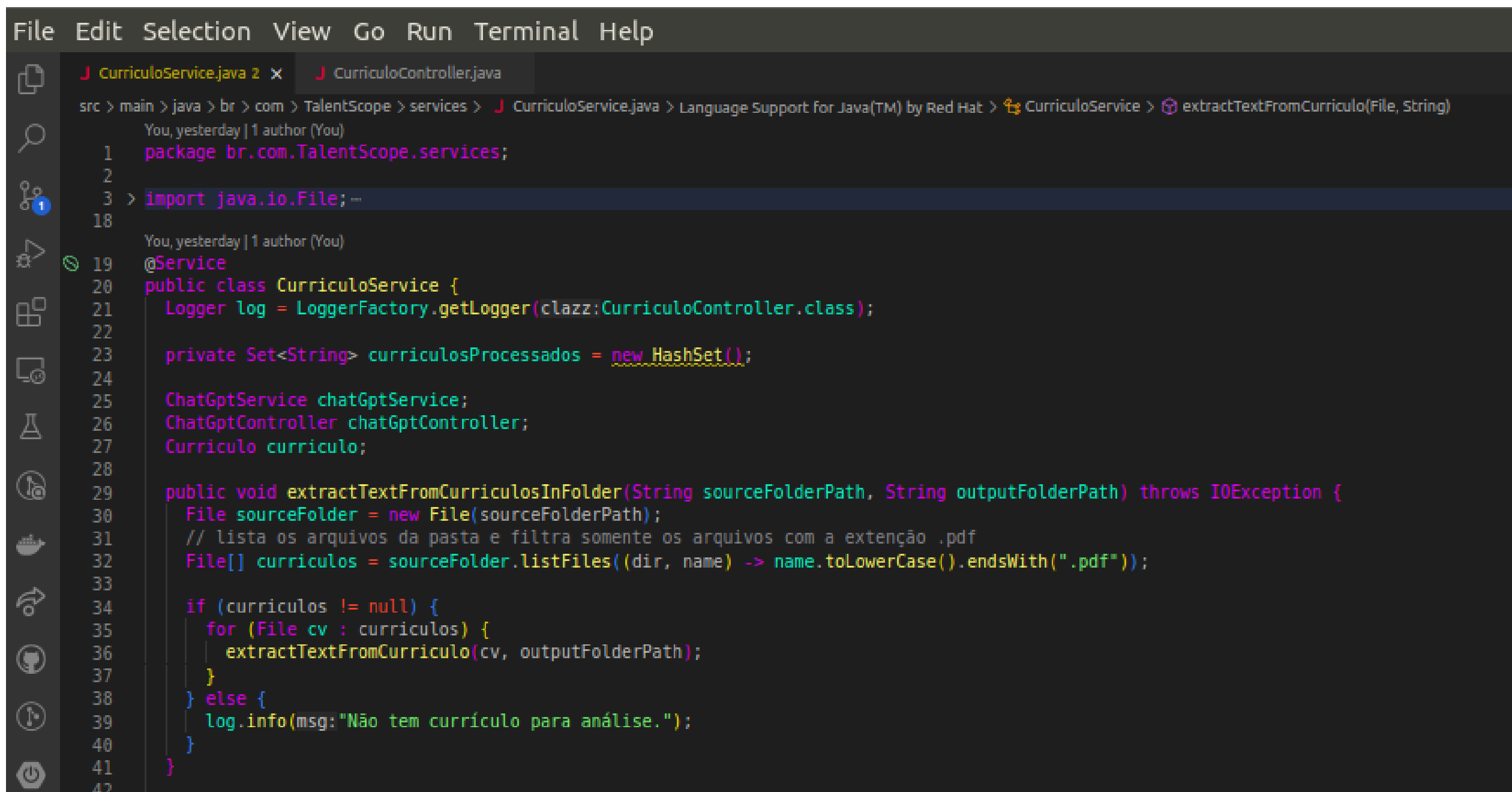
The screenshot displays the Insomnia application window. The top bar shows the system clock as 'dom, 21 de mai, 20:20' and the application title 'Insomnia'. The main interface is divided into several sections:

- Left Sidebar:** Contains a list of environments (No Environment, Cookies) and a collection of requests categorized by folders like 'Nivel Permissao', 'Feedback', 'Curriculo', and 'Usuario'. The 'Curriculo' folder is selected, showing a list of requests including 'New Request', 'UPDATE', 'DELETE', 'GET-BY-ID', and 'CREATE'.
- Top Bar:** Displays the current request details: 'GET localhost:8080/talentScope/curriculos'. The status is '200 OK' with a response time of '14.6 ms' and a body size of '1036 B'.
- Body Tab:** Shows the response body in JSON format, which is a list of two objects representing curriculum entries. The first object has an 'id' of 1, a file name 'rafael-cv.pdf', and a candidate name 'Rafael Ferreira'. The second object has an 'id' of 2 and a name 'Administrador'.
- Preview Tab:** Provides a visual representation of the JSON response, showing the nested structure of the data.

The bottom of the window features a 'Preferences' button and a 'Made with ❤ by Kong' watermark.

CORE DA APLICAÇÃO

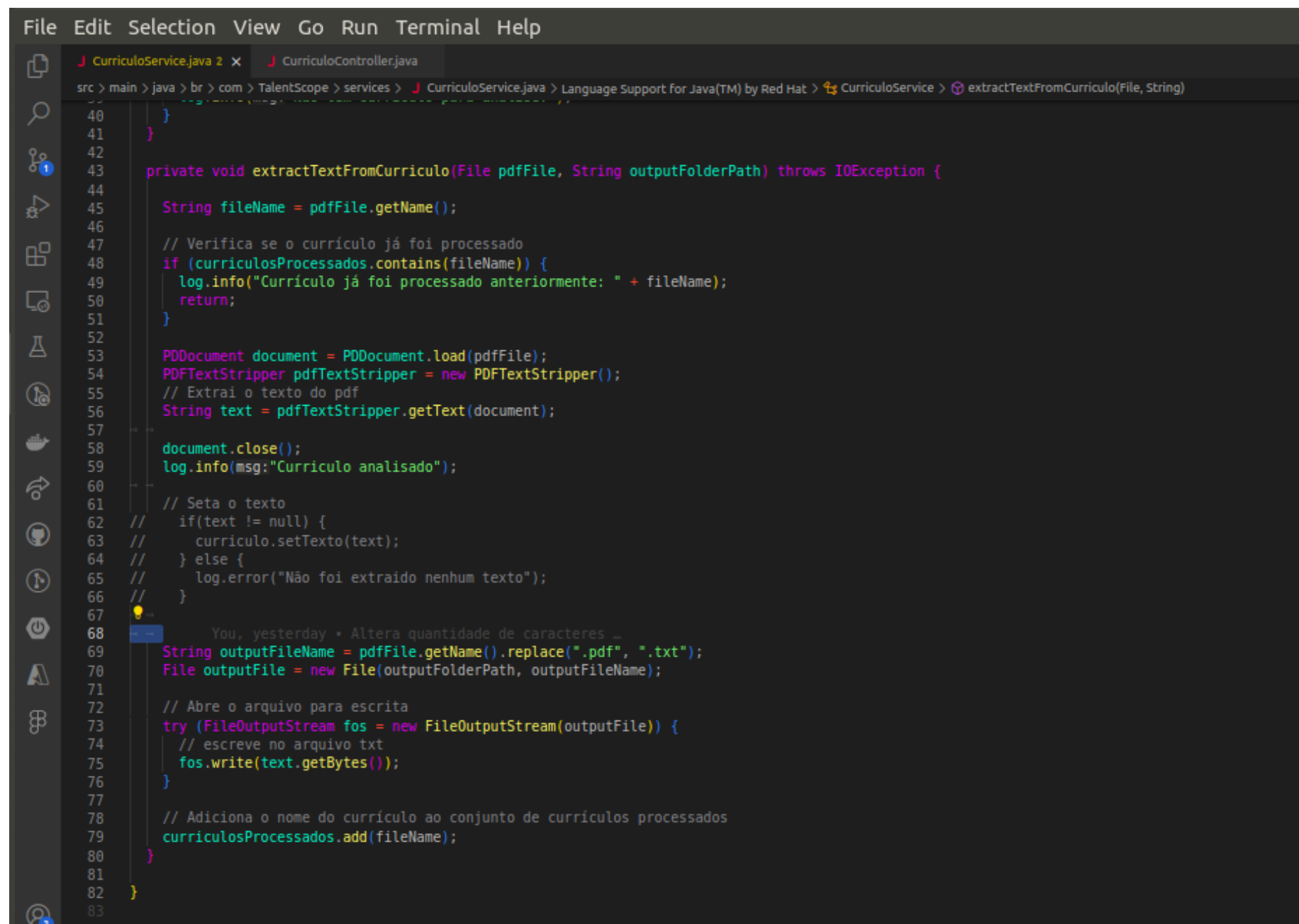
Classe service do currículo que faz o processo do tratamento do currículo. Recebe os currículos em PDF, extrai e guarda as informações do currículo e salva em outra pasta como arquivo txt.



```
File Edit Selection View Go Run Terminal Help
CurriculoService.java 2 x CurriculoController.java
src > main > java > br > com > TalentScope > services > CurriculoService.java > Language Support for Java(TM) by Red Hat > CurriculoService > extractTextFromCurrículo(File, String)
You, yesterday | 1 author (You)
1 package br.com.TalentScope.services;
2
3 > import java.io.File; -
18
You, yesterday | 1 author (You)
19 @Service
20 public class CurriculoService {
21     Logger log = LoggerFactory.getLogger(clazz:CurriculoController.class);
22
23     private Set<String> curriculosProcessados = new HashSet();
24
25     ChatGptService chatGptService;
26     ChatGptController chatGptController;
27     Curriculo curriculo;
28
29     public void extractTextFromCurrículosInFolder(String sourceFolderPath, String outputFolderPath) throws IOException {
30         File sourceFolder = new File(sourceFolderPath);
31         // lista os arquivos da pasta e filtra somente os arquivos com a extensão .pdf
32         File[] curriculos = sourceFolder.listFiles((dir, name) -> name.toLowerCase().endsWith(".pdf"));
33
34         if (curriculos != null) {
35             for (File cv : curriculos) {
36                 extractTextFromCurrículo(cv, outputFolderPath);
37             }
38         } else {
39             log.info(msg:"Não tem currículo para análise.");
40         }
41     }
42 }
```

CORE DA APLICAÇÃO

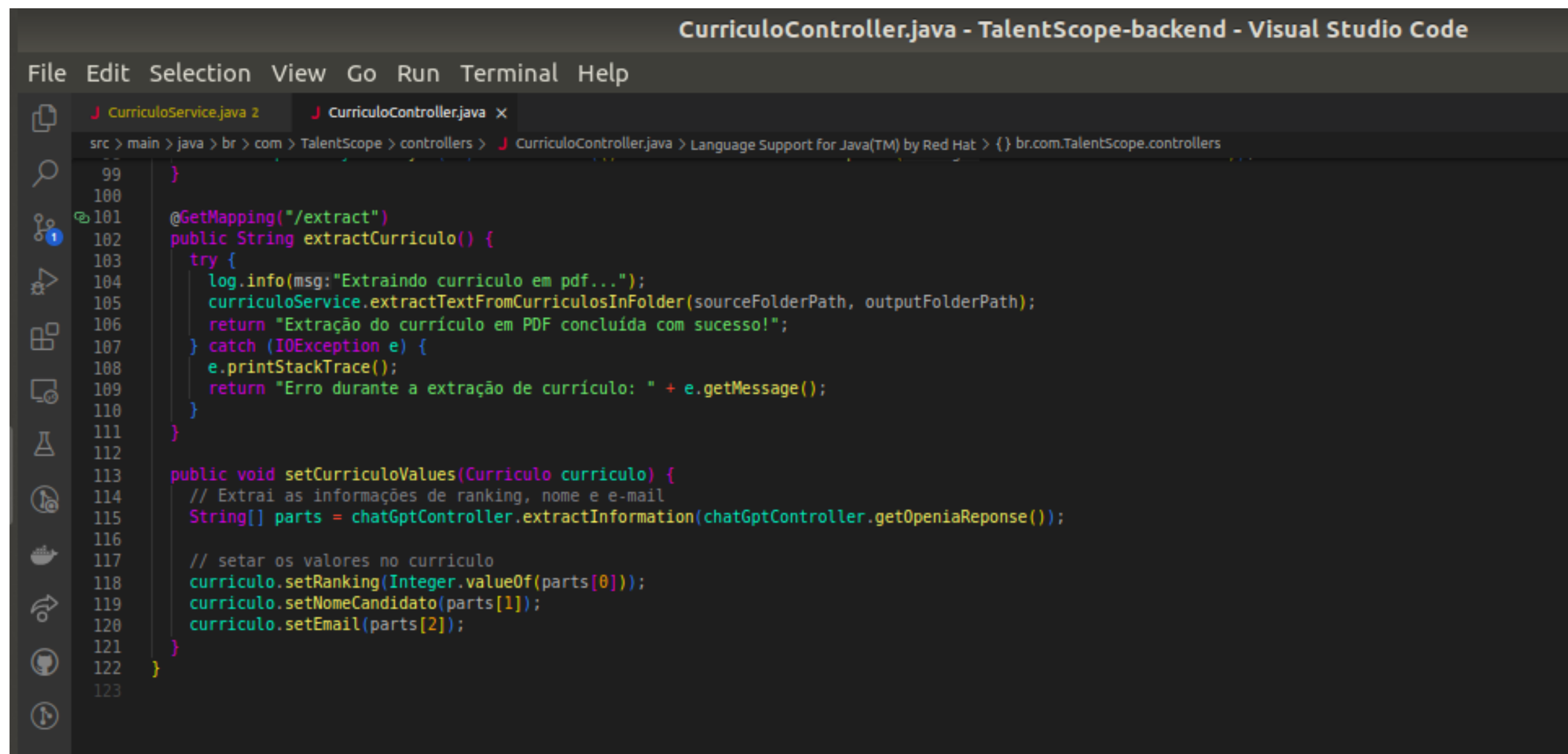
Classe service do currículo que faz o processo do tratamento do currículo. Recebe os currículos em PDF, extrai e guarda as informações do currículo e salva em outra pasta como arquivo txt.



```
File Edit Selection View Go Run Terminal Help
J CurriculoService.java 2 x J CurriculoController.java
src > main > java > br > com > TalentScope > services > J CurriculoService.java > Language Support for Java(TM) by Red Hat > CurriculoService > extractTextFromCurriculo(File, String)
40 }
41 }
42 }
43 private void extractTextFromCurriculo(File pdfFile, String outputFolderPath) throws IOException {
44     String fileName = pdfFile.getName();
45     // Verifica se o currículo já foi processado
46     if (currículosProcessados.contains(fileName)) {
47         log.info("Currículo já foi processado anteriormente: " + fileName);
48         return;
49     }
50     PDDocument document = PDDocument.load(pdfFile);
51     PDFTextStripper pdfTextStripper = new PDFTextStripper();
52     // Extrai o texto do pdf
53     String text = pdfTextStripper.getText(document);
54     document.close();
55     log.info(msg:"Currículo analisado");
56     // Seta o texto
57     // if(text != null) {
58     //     currículo.setTexto(text);
59     // } else {
60     //     log.error("Não foi extraído nenhum texto");
61     // }
62     You, yesterday • Altera quantidade de caracteres -
63     String outputFileName = pdfFile.getName().replace(".pdf", ".txt");
64     File outputFile = new File(outputFolderPath, outputFileName);
65     // Abre o arquivo para escrita
66     try (FileOutputStream fos = new FileOutputStream(outputFile)) {
67         // escreve no arquivo txt
68         fos.write(text.getBytes());
69     }
70     // Adiciona o nome do currículo ao conjunto de currículos processados
71     currículosProcessados.add(fileName);
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
```

CORE DA APLICAÇÃO

Classe controller do currículo com o endpoint para o tratamento do currículo



```
CurriculoController.java - TalentScope-backend - Visual Studio Code

File Edit Selection View Go Run Terminal Help

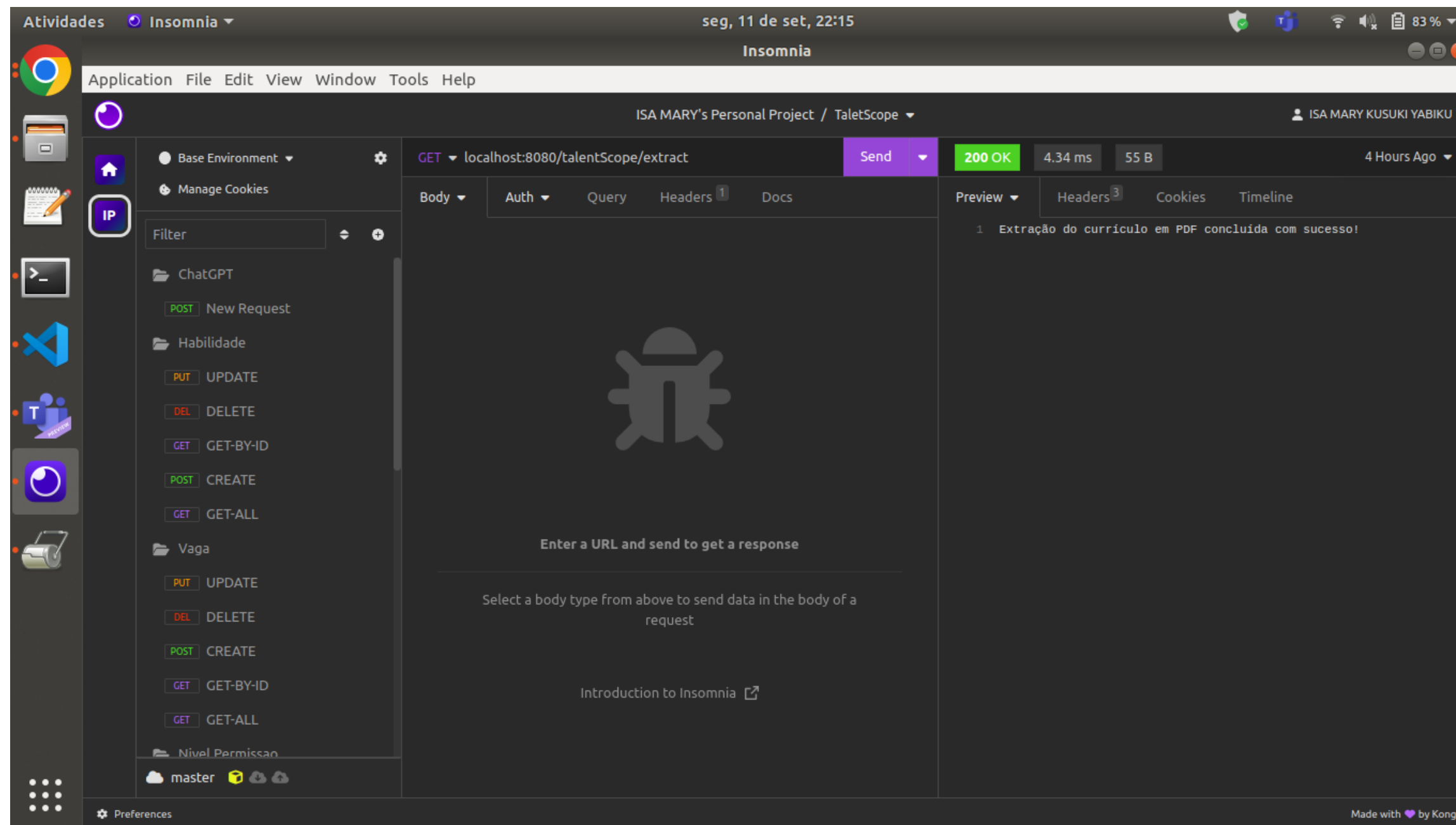
J CurriculoService.java 2 J CurriculoController.java x

src > main > java > br > com > TalentScope > controllers > J CurriculoController.java > Language Support for Java(TM) by Red Hat > {} br.com.TalentScope.controllers

99     }
100
101     @GetMapping("/extract")
102     public String extractCurriculo() {
103         try {
104             log.info(msg:"Extraindo curriculo em pdf...");
105             curriculoService.extractTextFromCurriculosInFolder(sourceFolderPath, outputFolderPath);
106             return "Extração do currículo em PDF concluída com sucesso!";
107         } catch (IOException e) {
108             e.printStackTrace();
109             return "Erro durante a extração de currículo: " + e.getMessage();
110         }
111     }
112
113     public void setCurriculoValues(Curriculo curriculo) {
114         // Extraí as informações de ranking, nome e e-mail
115         String[] parts = chatGptController.extractInformation(chatGptController.getOpeniaReponse());
116
117         // setar os valores no curriculo
118         curriculo.setRanking(Integer.valueOf(parts[0]));
119         curriculo.setNomeCandidato(parts[1]);
120         curriculo.setEmail(parts[2]);
121     }
122 }
123
```

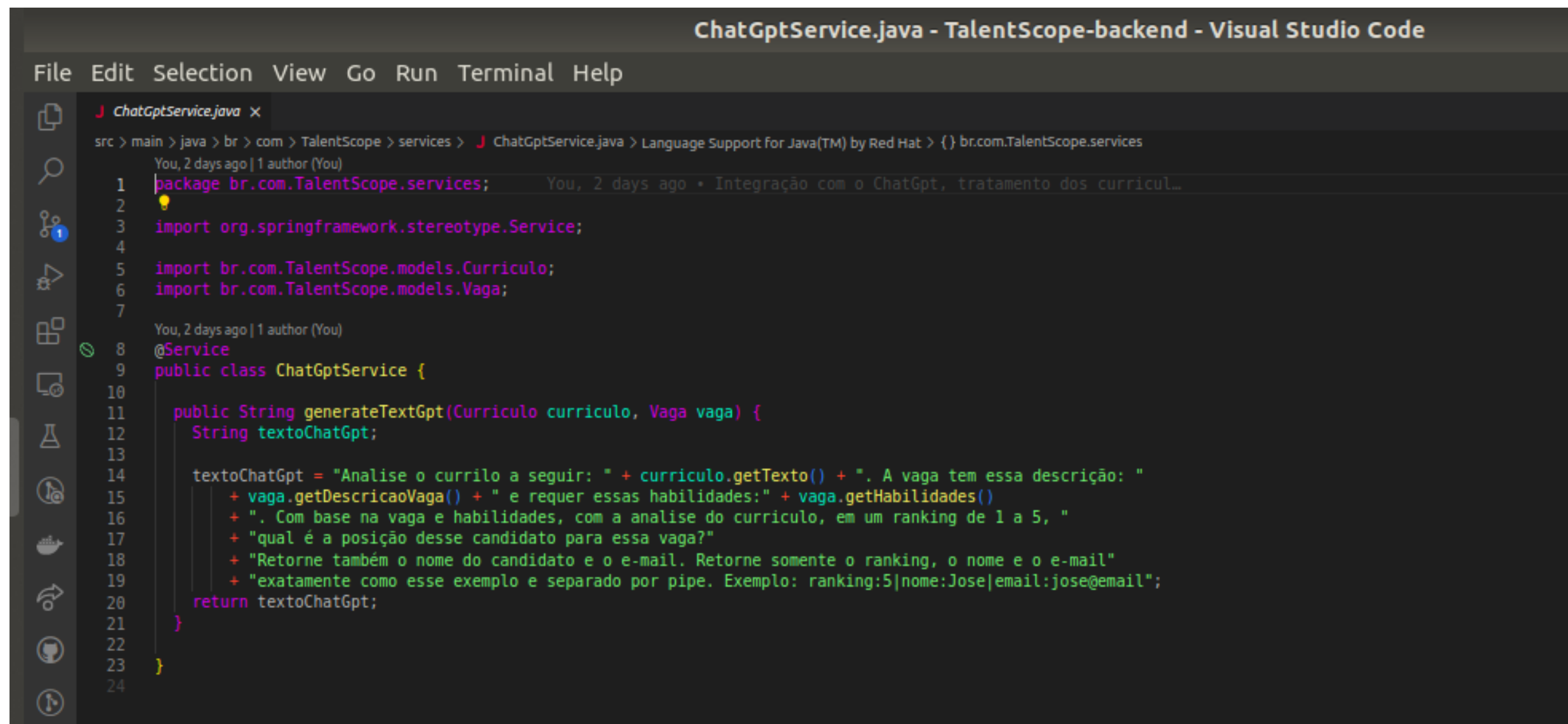
CORE DA APLICAÇÃO

Teste com insomnia do tratamento do currículo



CORE DA APLICAÇÃO

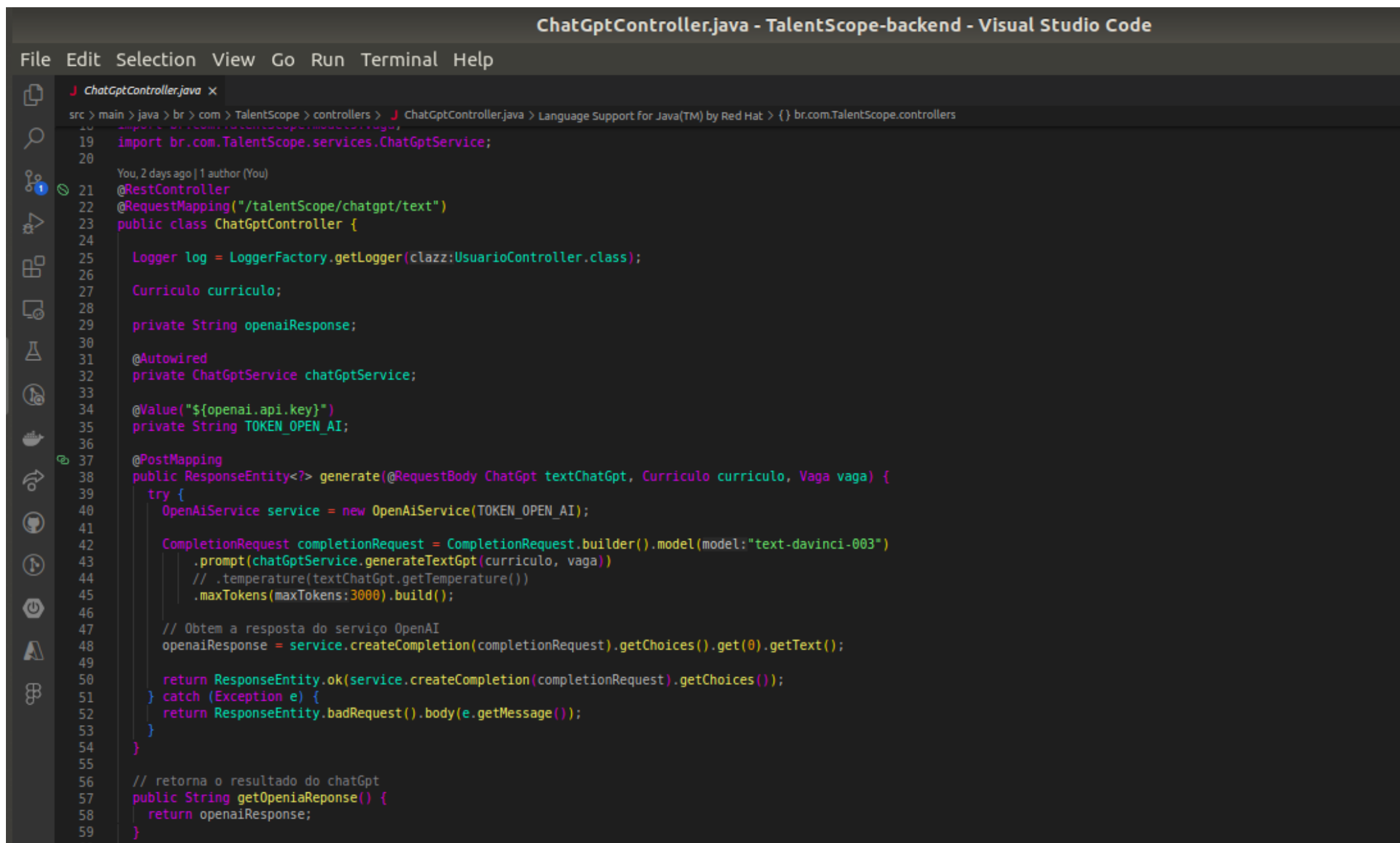
Classe service do ChatGpt que faz a geração do texto que iremos usar na API. É uma concatenação do texto extraído do currículo, das habilidades e da descrição da vaga.



```
ChatGptService.java - TalentScope-backend - Visual Studio Code
File Edit Selection View Go Run Terminal Help
J ChatGptService.java x
src > main > java > br > com > TalentScope > services > J ChatGptService.java > Language Support for Java(TM) by Red Hat > {} br.com.TalentScope.services
You, 2 days ago | 1 author (You)
1 package br.com.TalentScope.services;
2
3 import org.springframework.stereotype.Service;
4
5 import br.com.TalentScope.models.Curriculo;
6 import br.com.TalentScope.models.Vaga;
7
8 @Service
9 public class ChatGptService {
10
11     public String generateTextGpt(Curriculo curriculo, Vaga vaga) {
12         String textoChatGpt;
13
14         textoChatGpt = "Analise o currilo a seguir: " + curriculo.getTexto() + ". A vaga tem essa descrição: "
15             + vaga.getDescricaoVaga() + " e requer essas habilidades:" + vaga.getHabilidades()
16             + ". Com base na vaga e habilidades, com a analise do curriculo, em um ranking de 1 a 5, "
17             + "qual é a posição desse candidato para essa vaga?"
18             + "Retorne também o nome do candidato e o e-mail. Retorne somente o ranking, o nome e o e-mail"
19             + "exatamente como esse exemplo e separado por pipe. Exemplo: ranking:5|nome:Jose|email:jose@email";
20         return textoChatGpt;
21     }
22 }
23
24
```

CORE DA APLICAÇÃO

Classe controller do ChatGpt. ChatGpt recebe o texto, faz a análise, extrai as informações do Ranking, nome e email e salva em uma variável para que possamos salvar os dados do currículo no banco de dados.

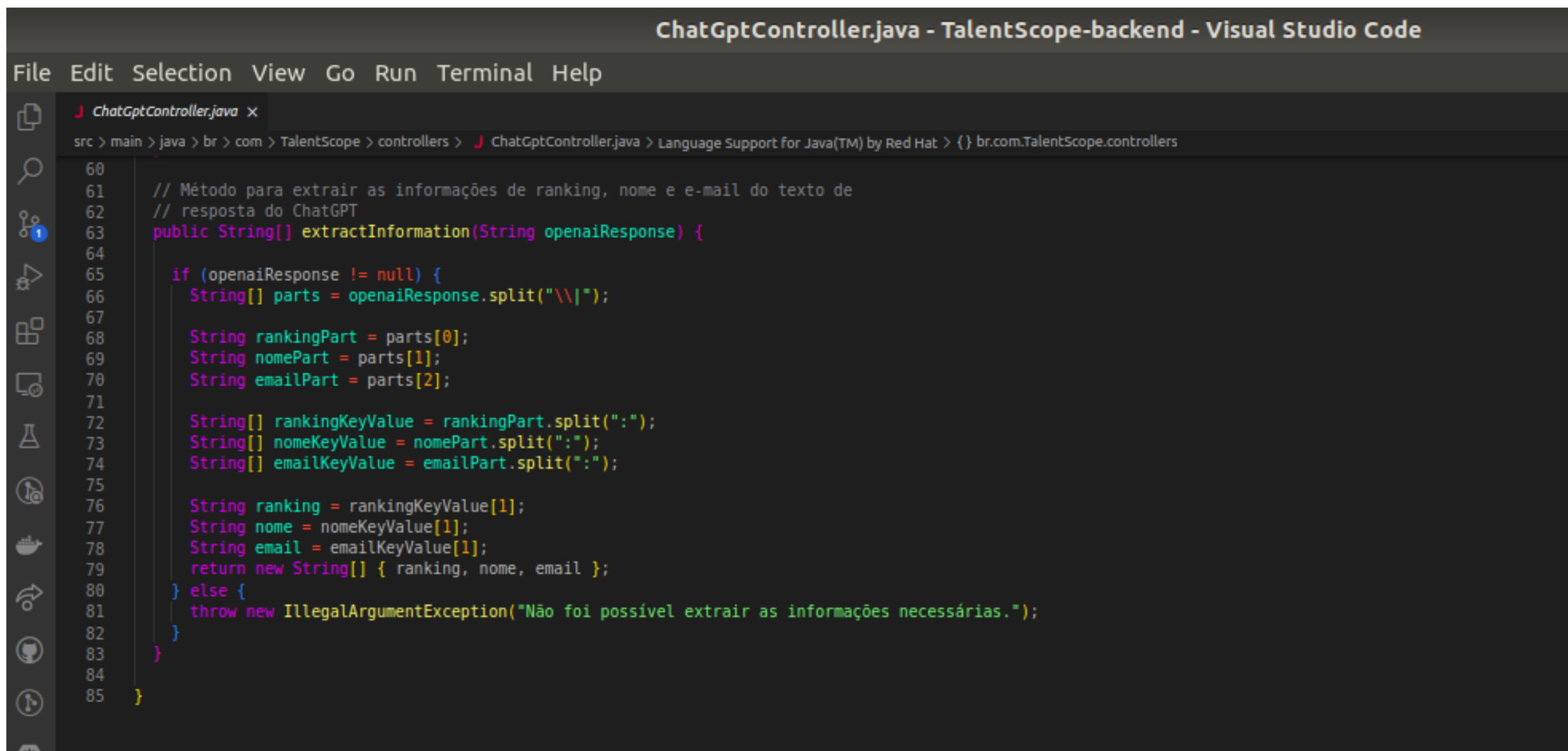


```
ChatGptController.java - TalentScope-backend - Visual Studio Code
File Edit Selection View Go Run Terminal Help
src > main > java > br > com > TalentScope > controllers > ChatGptController.java | Language Support for Java(TM) by Red Hat > {} br.com.TalentScope.controllers
import br.com.TalentScope.models.Vaga;
import br.com.TalentScope.services.ChatGptService;
import br.com.TalentScope.controllers.UsuarioController;

You, 2 days ago | 1 author (You)
21 @RestController
22 @RequestMapping("/talentScope/chatgpt/text")
23 public class ChatGptController {
24
25     Logger log = LoggerFactory.getLogger(clazz:UsuarioController.class);
26
27     Curriculo curriculo;
28
29     private String openaiResponse;
30
31     @Autowired
32     private ChatGptService chatGptService;
33
34     @Value("${openai.api.key}")
35     private String TOKEN_OPEN_AI;
36
37     @PostMapping
38     public ResponseEntity<?> generate(@RequestBody ChatGpt textChatGpt, Curriculo curriculo, Vaga vaga) {
39         try {
40             OpenAIService service = new OpenAIService(TOKEN_OPEN_AI);
41
42             CompletionRequest completionRequest = CompletionRequest.builder().model(model:"text-davinci-003")
43                 .prompt(chatGptService.generateTextGpt(curriculo, vaga))
44                 // .temperature(textChatGpt.getTemperature())
45                 .maxTokens(maxTokens:3000).build();
46
47             // Obtem a resposta do serviço OpenAI
48             openaiResponse = service.createCompletion(completionRequest).getChoices().get(0).getText();
49
50             return ResponseEntity.ok(service.createCompletion(completionRequest).getChoices());
51         } catch (Exception e) {
52             return ResponseEntity.badRequest().body(e.getMessage());
53         }
54     }
55
56     // retorna o resultado do chatGpt
57     public String getOpeniaReponse() {
58         return openaiResponse;
59     }
60 }
```


CORE DA APLICAÇÃO

Classe controller do ChatGpt. ChatGpt recebe o texto, faz a análise, extrai as informações do Ranking, nome e email e salva em uma variável para que possamos salvar os dados do currículo no banco de dados.



```
ChatGptController.java - TalentScope-backend - Visual Studio Code

File Edit Selection View Go Run Terminal Help

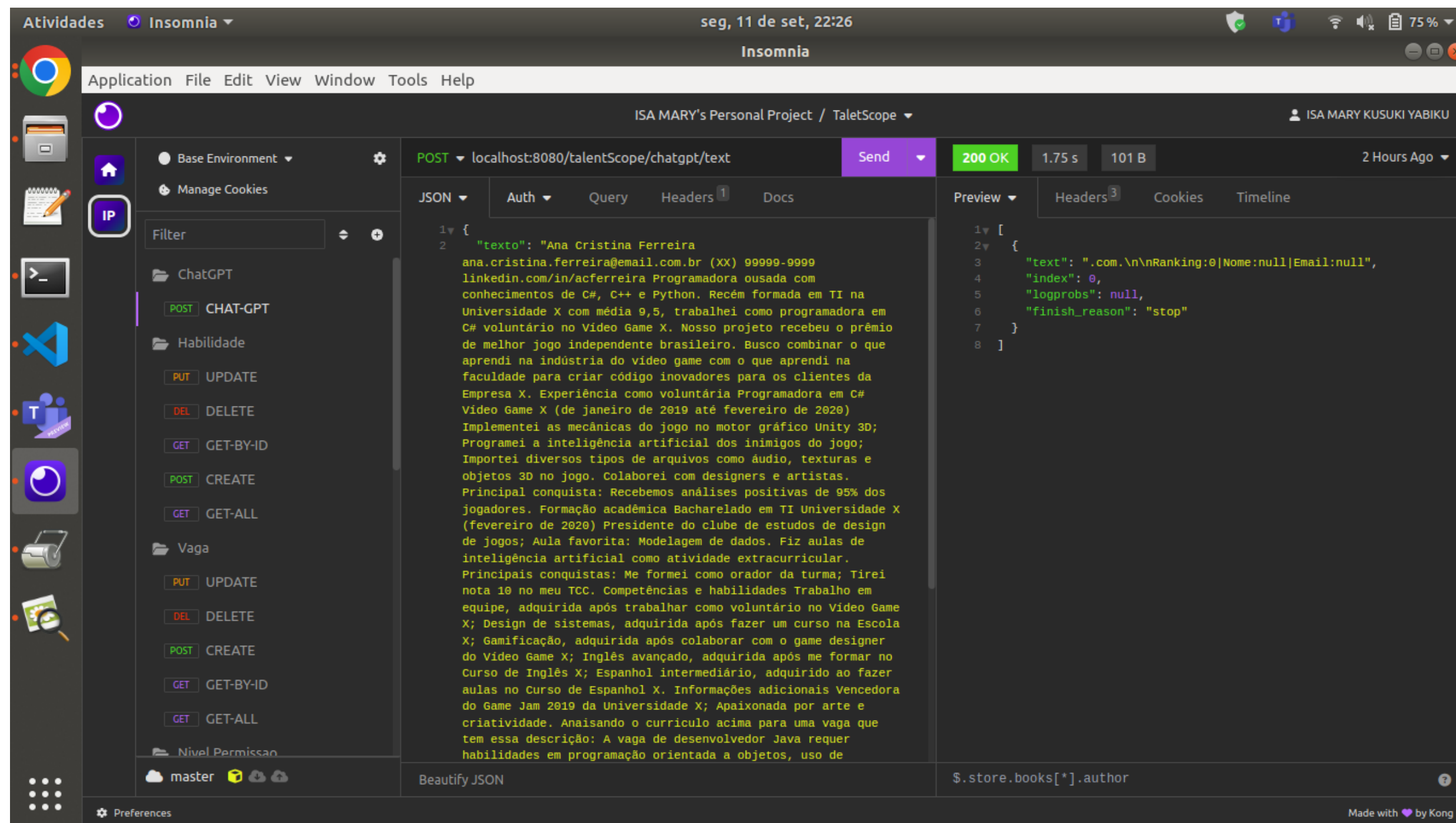
src > main > java > br > com > TalentScope > controllers > ChatGptController.java > Language Support for Java(TM) by Red Hat > {} br.com.TalentScope.controllers

60
61 // Método para extrair as informações de ranking, nome e e-mail do texto de
62 // resposta do ChatGPT
63 public String[] extractInformation(String openaiResponse) {
64
65     if (openaiResponse != null) {
66         String[] parts = openaiResponse.split("\\\\");
67
68         String rankingPart = parts[0];
69         String nomePart = parts[1];
70         String emailPart = parts[2];
71
72         String[] rankingKeyValue = rankingPart.split(":");
73         String[] nomeKeyValue = nomePart.split(":");
74         String[] emailKeyValue = emailPart.split(":");
75
76         String ranking = rankingKeyValue[1];
77         String nome = nomeKeyValue[1];
78         String email = emailKeyValue[1];
79         return new String[] { ranking, nome, email };
80     } else {
81         throw new IllegalArgumentException("Não foi possível extrair as informações necessárias.");
82     }
83 }
84
85 }
```

CORE DA APLICAÇÃO

Teste com Insomnia do ChatGpt.

Obs.: Ainda estamos fazendo o desenvolvimento da aplicação, por isso ainda não está retornando os valores esperados.



CORE DA APLICAÇÃO

Em nosso programa iremos persistir os dados utilizando JPA Hibernate, relacionando as classes entre si da seguinte forma:

Um usuário terá um nível de permissão, relação 1:N.

Um usuário poderá criar uma ou mais vagas, relação 1:N.

Uma vaga poderá ter várias habilidades que o recrutador escolherá conforme a vaga necessita, e as habilidades poderão estar em várias vagas, fazendo desta, uma ligação M:N, com a tabela associativa “TB_VAGA_HABILIDADE”.

A Vaga também estará associada à tabela “TB_CURRICULO”, podendo ter vários currículos, relação 1:N.

A tabela currículo está relacionada com a tabela de feedback, a qual armazenará a resposta do candidato.

DIFERENCIAL

- 1 TRIAGEM REALIZADA DE FORMA IMEDIATA
- 2 NÃO NECESSITA DE PLATAFORMA EXTERNA DE VAGAS
- 3 FEEDBACK PARA TODOS OS CANDIDATOS REPROVADOS
- 4 OPÇÃO DE FEEDBACK PROGRAMADO OU AUTOMÁTICO

FONTE DE RECEITA

COMO FONTE DE RECEITA, AS EMPRESAS
PODERÃO CONTRATAR O NOSSO
APLICATIVO POR MEIO DE ASSINATURA
MENSAL, SEMESTRAL OU ANUAL. O VALOR
AINDA SERÁ DEFINIDO.

INTEGRANTES

BEATRIZ TORRES LOPES – RM 95838

GIOVANA GOUVEA ABREU – RM 95896

ISA MARY KUSUKI – RM 93307

JONATAS LIMA BARBOSA – RM 93902

RUAN SANTOS DIAS – RM 94528