

PROJETO CRUD - BACKEND

SUMÁRIO

PROJETO CRUD - BACKEND.....	1
OBJETIVO.....	7
DEPENDÊNCIAS.....	7
CASO DE USO.....	7
DIAGRAMA DE CLASSE.....	8
TESTES A SEREM REALIZADOS NO POSTMAN.....	9
BUSCA PAGINADA DE ALUNOS.....	9
BUSCA DE ALUNO POR ID.....	9
INSERIR NOVO ALUNO.....	9
ATUALIZAR ALUNO.....	9
DELETAR ALUNO.....	9
IMPLEMENTAR O BACKEND.....	10
DEFINIR O WORKSPACE PARA O PROJETO.....	10
CRIAR O PROJETO COM O INITIALIZR.....	10
Salvar dentro do workspace.....	11
Descompactar e Renomear.....	11
IMPORTAR O PROJETO PARA O STS.....	12
TESTAR NO BROWSER.....	14
VERSIONAMENTO.....	14
Criar o projeto no Github.....	14
Sincronizar com o projeto local via git.....	16
Adicionar o README ao projeto.....	20
Atualizar o projeto local.....	22
INICIAR O DESENVOLVIMENTO DO PROJETO.....	24
STUDENT_CLASSE.....	24
DIAGRAMA DE CLASSE.....	24
IMPLEMENTAR A ESTRUTURA E A CLASSE STUDENT NO STS.....	25
Criar a estrutura e classe.....	25
Definir o Serializable e os atributos da Classe.....	25
Criar o construtor.....	27
Criar os Getters and Setters.....	28

Criar o hashCode and equals.....	30
Github-2.....	31
STUDENT_CONTROLLER.....	32
CONCEITUAL.....	32
IMPLEMENTAR A ESTRUTURA.....	33
Criar a estrutura e o recurso StudentController.....	33
Implementar as Notações Rest.....	33
Criar o Endpoint findAll para teste.....	34
Rodar o projeto.....	34
Testar com o Postman.....	35
Github-3.....	35
STUDENT_REPOSITORY.....	36
CONCEITUAL.....	36
IMPLEMENTAR A ESTRUTURA E O STUDENT_REPOSITORY.....	37
Criar a estrutura e a interface de acesso ao banco.....	37
Implementar a notação e estender a JPA.....	37
Github-4.....	38
STUDENT_SERVICE.....	39
CONCEITUAL.....	39
IMPLEMENTAR A ESTRUTURA E O STUDENT_SERVICE.....	40
Criar a estrutura e a classe de serviço.....	40
Implementar a lógica para o endpoint findAll.....	41
Github-5.....	42
INTEGRAÇÃO COM O BANCO.....	43
AJUSTAR AS CAMADAS.....	43
Implementar o StudentController.....	43
Implementar a classe Student.....	44
Rodar o projeto.....	44
Testar com o Postman.....	44
BANCO H2.....	45
Configurar o perfil de teste no application.properties.....	45
Criar o arquivo de configuração application-test.properties.....	46
Implementar o application-test.properties.....	47
Rodar o projeto.....	47
Acessar o banco H2, via web.....	48
Teste de inserção.....	49
Testar no postman.....	49
Github-6.....	50
SEEDING DA BASE DE DADOS.....	51

IMPLEMENTAR A CARGA PARA O BANCO.....	51
Criar o import.sql, no src/main/resources.....	51
Implementar os inserts para a carga inicial.....	52
Rodar o projeto.....	52
TESTAR A CARGA.....	53
Testar no banco H2.....	53
Testar no Postman.....	54
Github-7.....	55
DTO.....	56
CONCEITUAL.....	56
IMPLEMENTAR A ESTRUTURA E O STUDENT_DTO.....	57
Criar a estrutura e a classe DTO.....	57
Implementar o Serializable e os mesmos atributos da Classe Student.....	57
Implementar os construtores.....	58
Vazio.....	58
de Classe.....	58
De Entidade.....	59
Implementar os Getters and Setters.....	59
REALIZAR OS AJUSTES PARA O DTO.....	61
Implementar o DTO na classe StudentService.....	61
Implementar o DTO na classe StudentController.....	62
Rodar o projeto.....	62
TESTAR COM O POSTMAN.....	63
Github-8.....	64
ENDPOINT: FIND_BY_ID.....	65
BUSCAR ALUNOS POR ID COM GET.....	65
Implementar busca por Id, no StudentController.....	65
Implementar o método findById, no StudentService.....	65
Rodar o projeto.....	66
TESTAR NO POSTMAN.....	67
Github-9.....	68
TRATAMENTO DE EXCEÇÕES PARA O FIND_BY_ID.....	69
Rodar o projeto.....	69
Simular erro no Postman.....	69
Verificar o erro no console.....	69
Criar a estrutura de exceções no service.....	70
Implementar o ResourceNotFoundException.....	70
Implementar a exceção no findById, do StudentService.....	71
Criar a estrutura de exceções e uma classe personalizada no controller.....	71

Implementar o Serializable e os atributos do StandardError conforme erro visto anteriormente.....	72
Implementar um construtor vazio.....	73
Implementar os Getters and Setters.....	73
Criar um controller advice para manipular a exceção.....	75
Implementar o ResourceExceptionHandler.....	76
Rodar o projeto.....	76
TESTAR NO POSTMAN.....	77
Github-10.....	77
PAGINAÇÃO.....	78
AJUSTAR O FIND_ALL PARA BUSCA PAGINADA.....	78
Implementar a busca paginada, no StudentController.....	78
Ajustar a busca paginada, no StudentService.....	79
Expandir o seed do banco para teste de paginação.....	80
Rodar o projeto.....	80
TESTAR NO POSTMAN.....	81
Github-11.....	82
ENDPOINT - INSERT.....	83
INSERIR NOVO ALUNO COM POST.....	83
Implementar o insert, no StudentController.....	83
Implementar a metodologia REST ao método.....	83
Implementar o método insert, convertendo o DTO para uma entidade, no StudentService.....	84
Rodar o projeto.....	84
TESTAR NO POSTMAN.....	85
Inserir.....	85
Buscar por Id.....	86
Github-12.....	86
ENDPOINT - UPDATE.....	87
ATUALIZAR ALUNO COM PUT.....	87
Implementar o update, no StudentController.....	87
Implementar o método update, no StudentService.....	87
Rodar o projeto.....	87
TESTAR NO POSTMAN.....	88
Update.....	88
Busca por ID.....	88
TRATAMENTO DE ERRO PARA O UPDATE.....	89
Implementar o tratamento para ID Não encontrado.....	89
Rodar o projeto.....	89

TESTAR NO POSTMAN.....	90
Update.....	90
Github-13.....	90
ENDPOINT - DELETE.....	91
DELETAR UM ALUNO COM O MÉTODO REST DELETE.....	91
Implementar o update, no StudentController.....	91
Implementar o método update, no StudentService.....	91
Rodar o projeto.....	91
TESTAR NO POSTMAN.....	92
TRATAMENTO DE ERRO DO DELETE.....	92
Implementar o tratamento para ID Não encontrado.....	92
Rodar o projeto.....	93
TESTAR NO POSTMAN.....	93
Github-14.....	93
LIBERAÇÃO CORS PARA FRONTEND.....	94
IMPLEMENTAR O CONFIG.....	94
Criar o CorsConfig.....	94
Implementar o CorsConfig.....	94
Github-15.....	95
IMPLEMENTAR O SWAGGER.....	96
PROJETO.....	96
IMPLEMENTAÇÃO DO SWAGGER.....	97
DEPENDÊNCIA MAVEN.....	97
IMPLEMENTAR O MAIN PRINCIPAL.....	97
IMPLEMENTAR O CONTROLLER.....	98
FindAll.....	98
FindByld.....	98
Insert.....	98
Update.....	99
Delete.....	99
ACESSAR O SWAGGER.....	99
Rodar o projeto.....	99
Acesso online.....	100
Testar o Swagger.....	101
Buscar todos.....	101
Busca por ID.....	102
Atualizar.....	103
Inserir.....	104
Deletar.....	105

Github-16.....	105
Fim.....	105

OBJETIVO

Criar uma API RESTful com as funcionalidades de CRUD. Criar um front end intuitivo, sincronizado com o backend, para realizar as operações de visualização, inserção, edição e deleção dos dados.

DEPENDÊNCIAS

O projeto será implementado conforme abaixo:

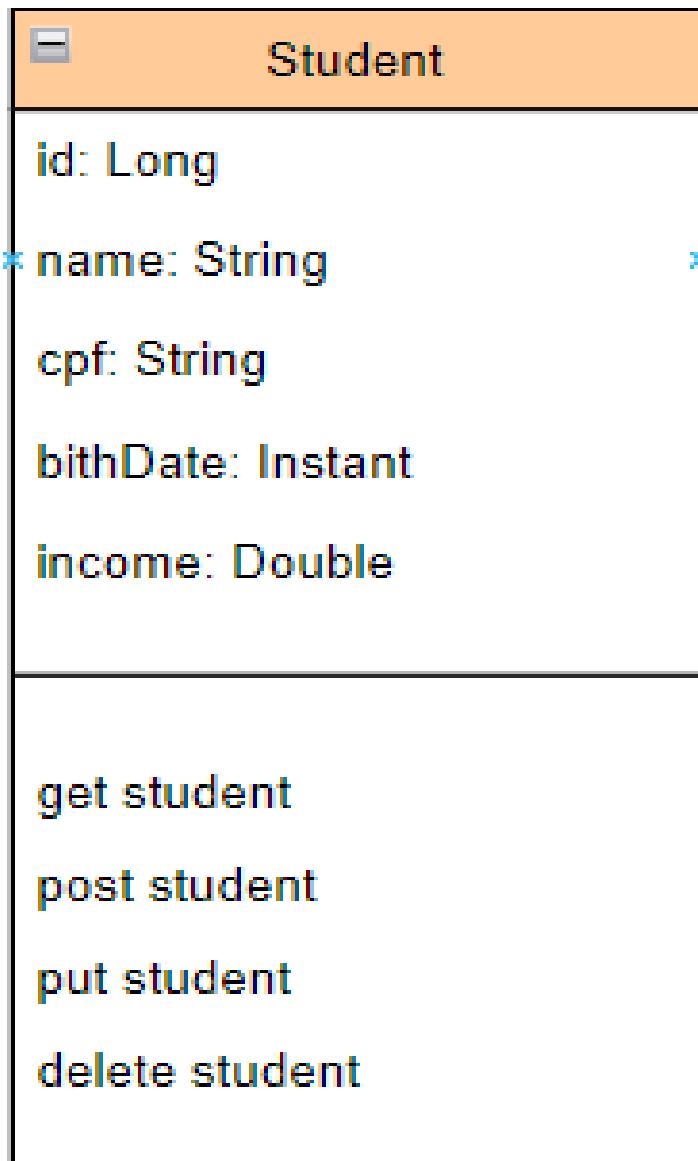
- **IDE:** Spring Tool Suite 4 (STS)
- **Banco de Dados:** H2
- **Gerenciador de Dependências:** Maven
- **Linguagem:** Java
- **Versionamento:** GitHub
- **Testes da API:** Postman
- **Documentação:** Swagger OpenApi

CASO DE USO

Uma Universidade precisa cadastrar os seus alunos, conforme segue abaixo:

- NOME
- CPF
- DATA DE NASCIMENTO
- RENDA

DIAGRAMA DE CLASSE



TESTES A SEREM REALIZADOS NO POSTMAN

BUSCA PAGINADA DE ALUNOS

- GET /students?page=0&sort=name,asc

BUSCA DE ALUNO POR ID

- GET /students/1

INSERIR NOVO ALUNO

- POST /students
{
 "name": "Auricelio Freitas",
 "cpf": "12345678901",
 "birthDate": "1982-08-28T10:30:00Z",
 "income": 15089.0
}

ATUALIZAR ALUNO

- PUT /students/1
{
 "name": "Auricelio Moreira",
 "cpf": "12345678901",
 "birthDate": "1982-08-28T10:30:00Z",
 "income": 15089.0
}

DELETAR ALUNO

- DELETE /students/1

IMPLEMENTAR O BACKEND

DEFINIR O WORKSPACE PARA O PROJETO

- C:\PROJETOS\ProjCrudBackFront

CRIAR O PROJETO COM O INITIALIZR

ACESSAR A URL

- <https://start.spring.io/>

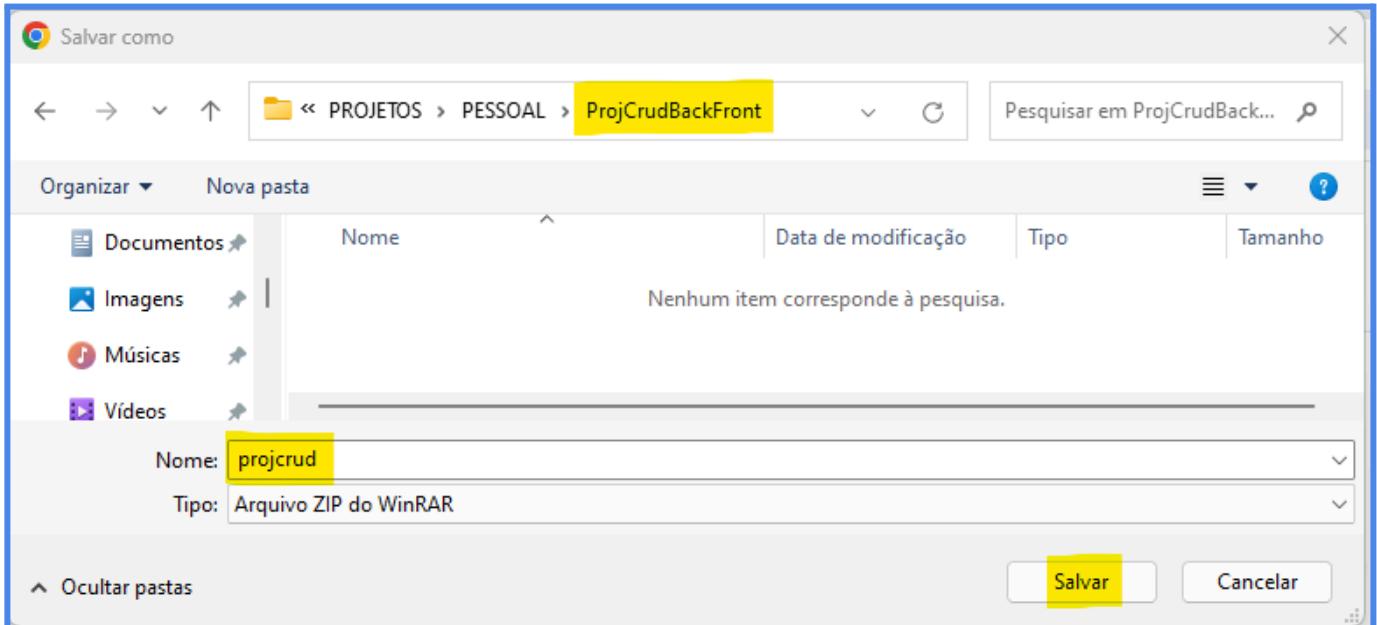
GERAR O PROJETO

The screenshot shows the Spring Initializr web interface at <https://start.spring.io/>. The configuration is as follows:

- Project:** Maven (selected)
- Language:** Java (selected)
- Spring Boot:** 3.3.1 (selected)
- Dependencies:**
 - Spring Web**: Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Spring Data JPA**: Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
 - H2 Database**: Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.
 - PostgreSQL Driver**: A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.
- Project Metadata:**
 - Group: pessoal
 - Artifact: projcrud
 - Name: projcrud
 - Description: Projeto CRUD Restful
 - Package name: pessoal.projcrud
 - Packaging: Jar (selected)
- Java Version:** 17 (selected)

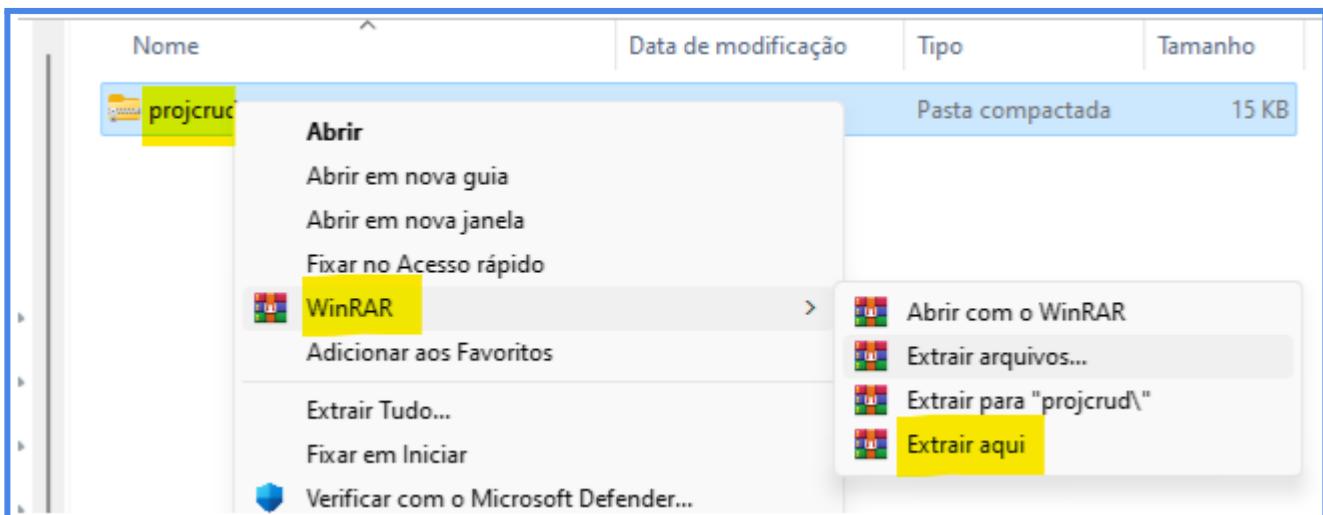
At the bottom, there are buttons for **GENERATE** (CTRL + ⌘), **EXPLORE** (CTRL + SPACE), and **SHARE...**.

Salvar dentro do workspace

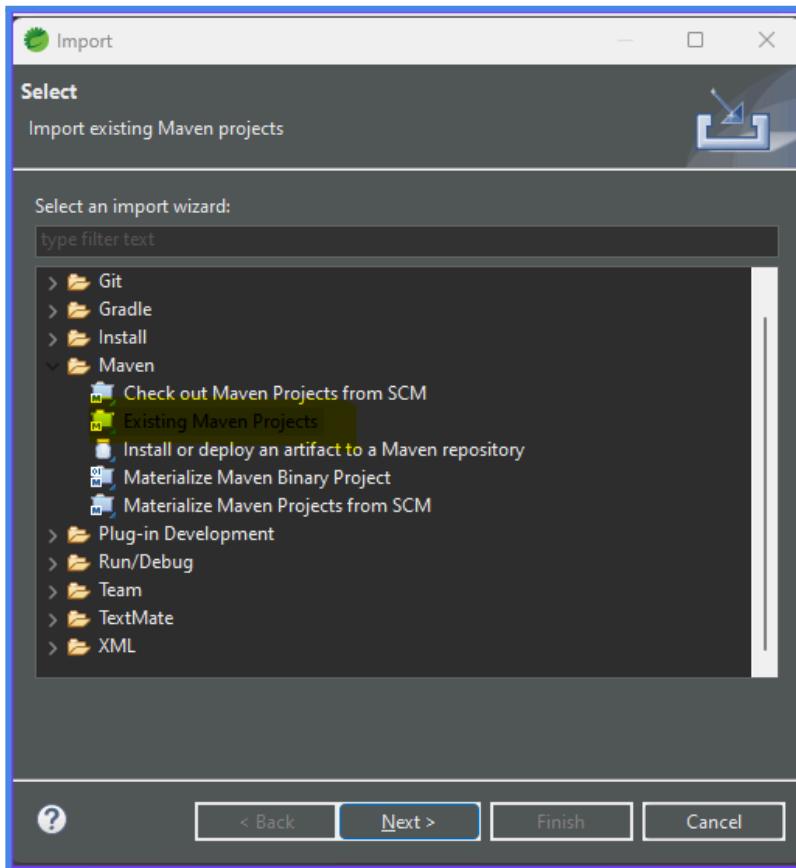
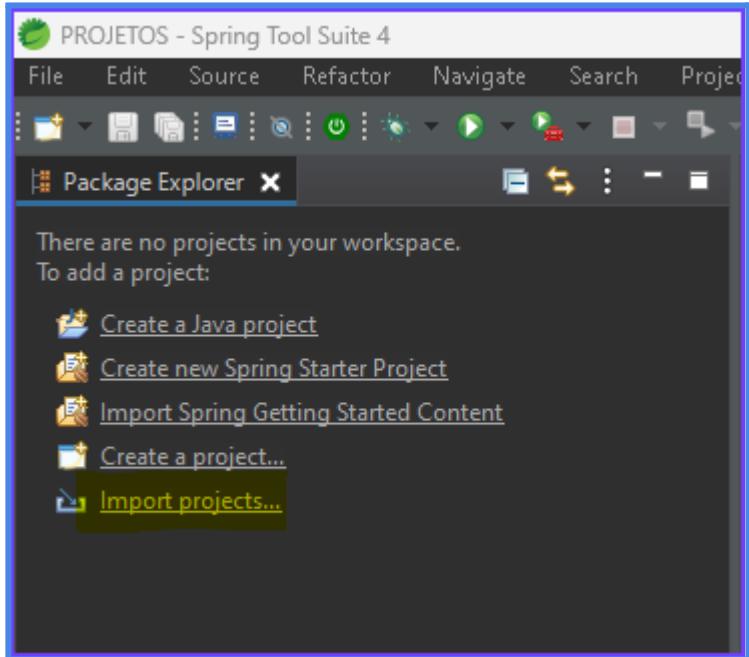


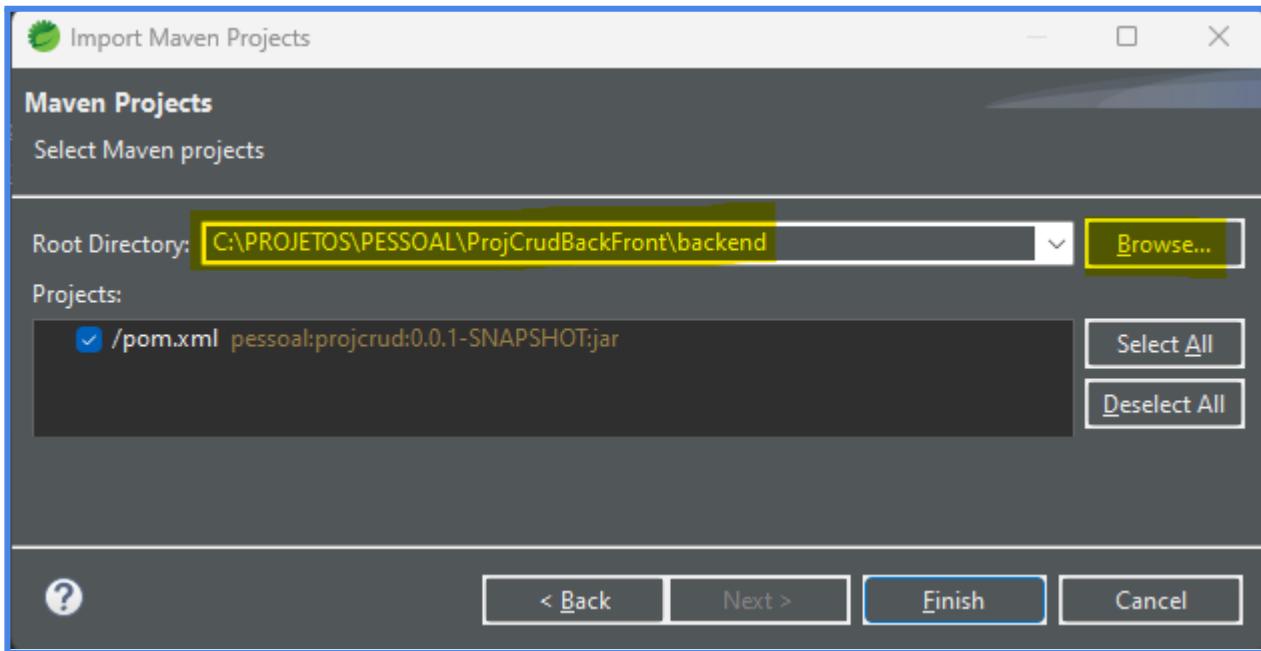
Descompactar e Renomear

- backend

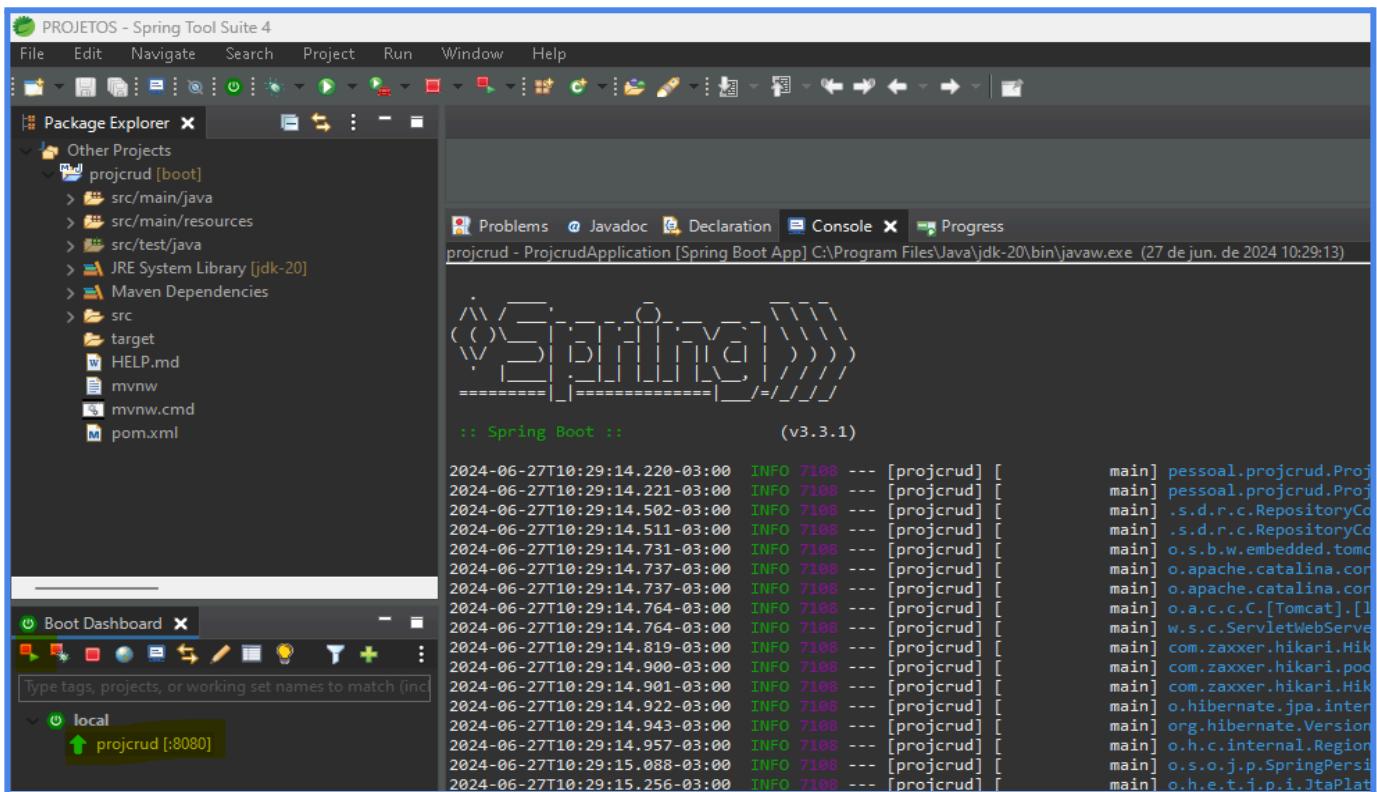


IMPORTAR O PROJETO PARA O STS



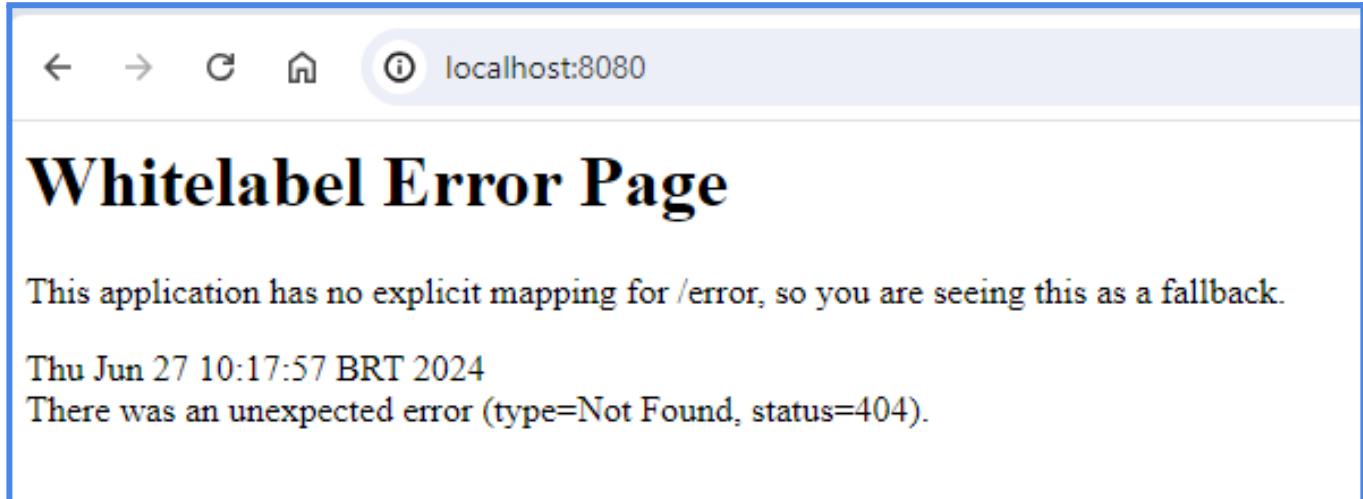


- Rodar o projeto



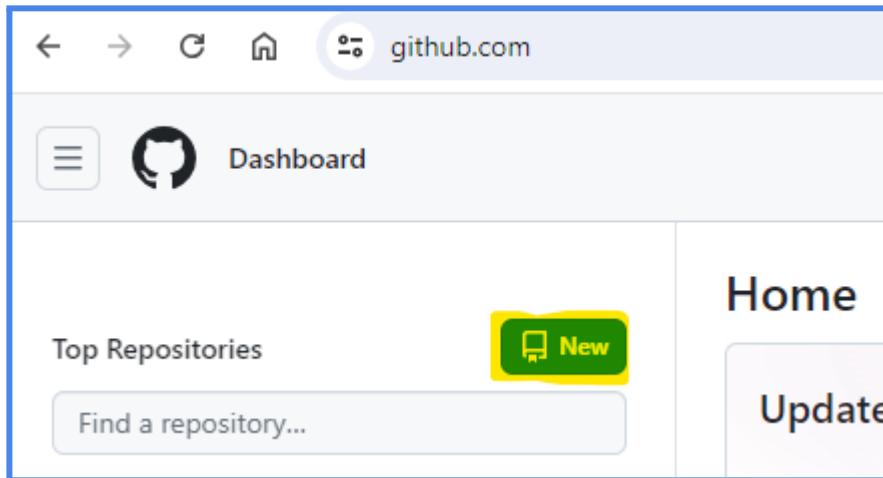
TESTAR NO BROWSER

- URL: <http://localhost:8080/>



VERSIONAMENTO

Criar o projeto no Github



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

 auriceliof / pessoal-ProjCrudBackFront
✓ pessoal-ProjCrudBackFront is available.

Great repository names are short and memorable. Need inspiration? How about [fuzzy-octo-giggle](#) ?

Description (optional)
Projeto CRUD Restful com Backend e Frontend

 Public
Anyone on the internet can see this repository. You choose who can commit.

 Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
[.gitignore template: None](#)

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
[License: None](#)

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.

Create repository

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/auriceliof/pessoal-ProjCrudBackFront.git> [Copy](#)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# pessoal-ProjCrudBackFront" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
git push -u origin main
```

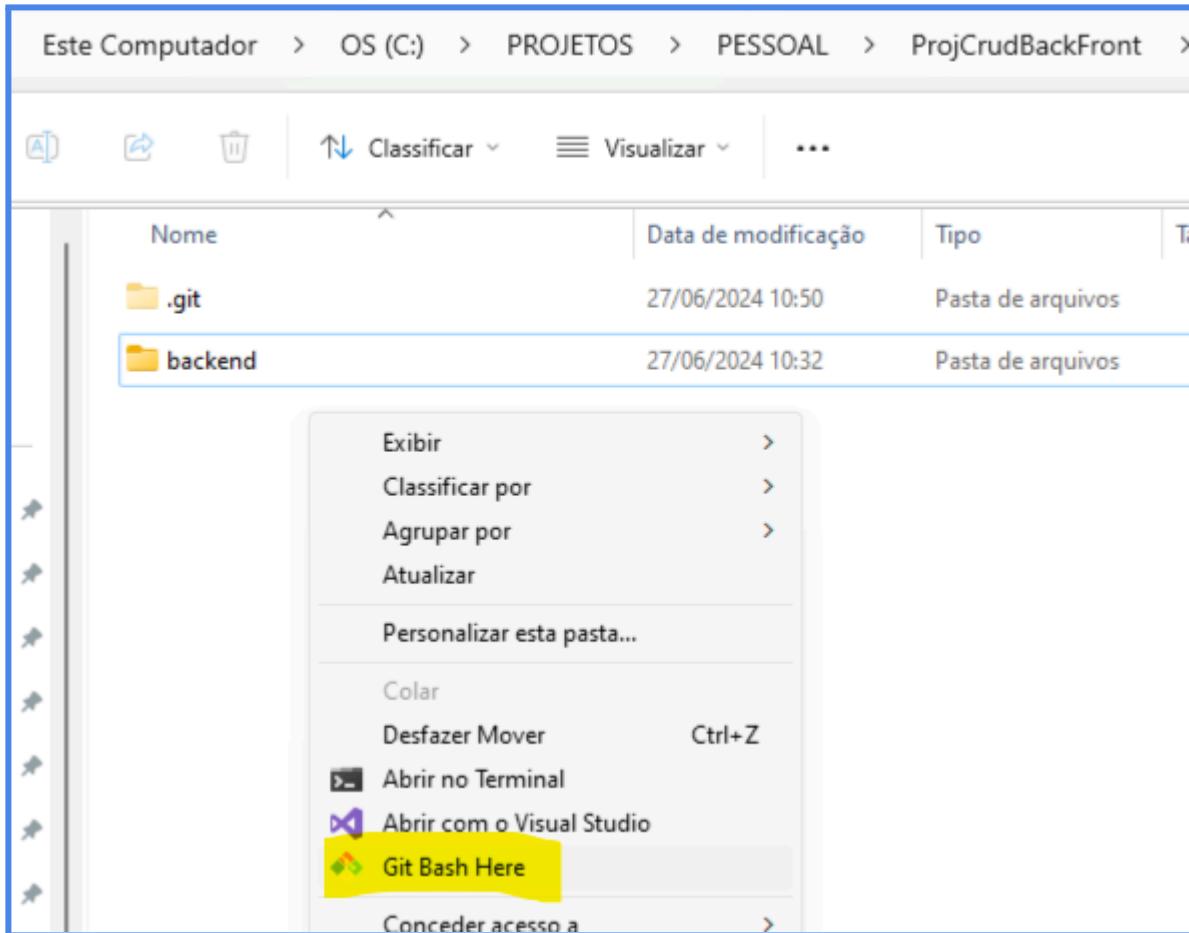
...or push an existing repository from the command line

```
git remote add origin https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
git branch -M main
git push -u origin main
```

Sincronizar com o projeto local via git

NOTA: O Git deve estar instalado no computador. Caso não tenha ainda, baixe a versão mais atual e instale, dando next até o final.

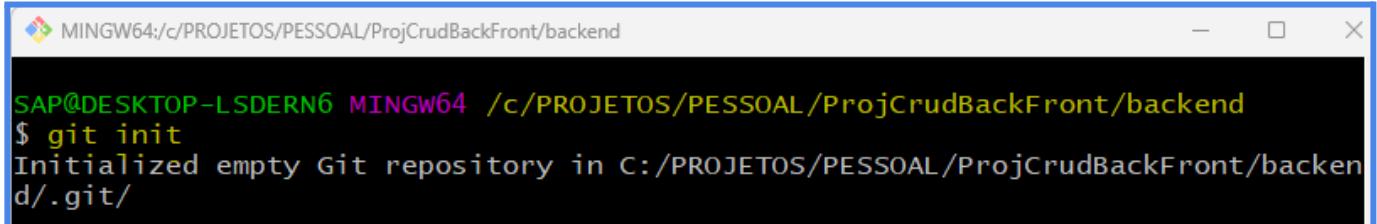
- Botão direito no projeto e clicar em “Open Git bash here”



ProjCrudBackFront

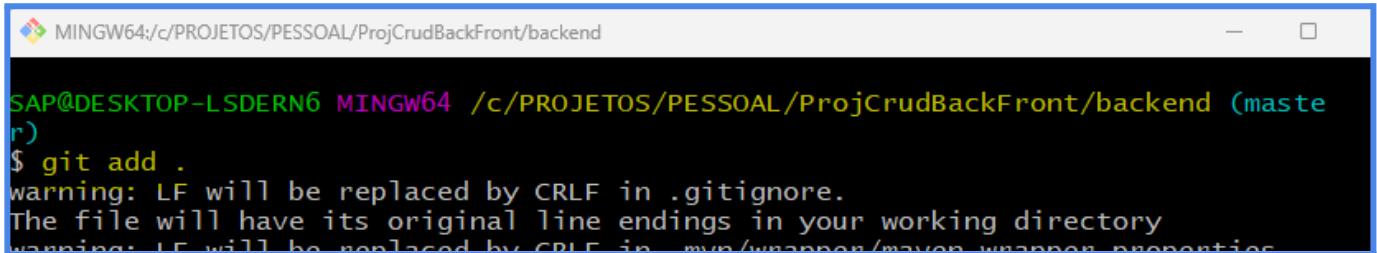
- Efetuar os comando abaixo

- git init



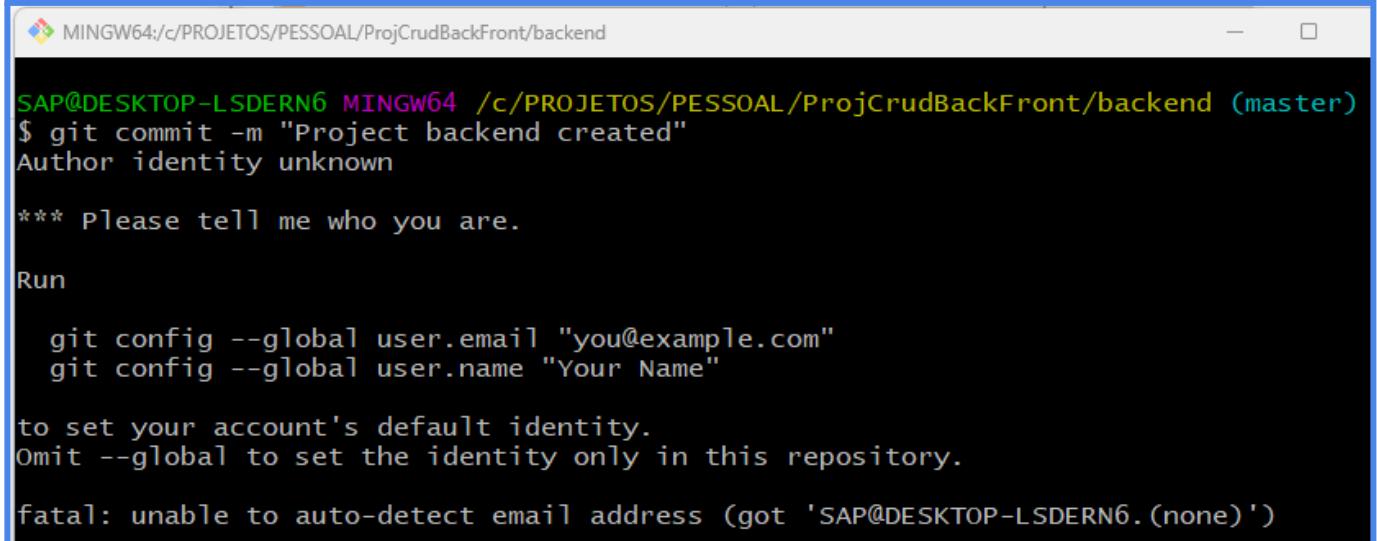
```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend
$ git init
Initialized empty Git repository in C:/PROJETOS/PESSOAL/ProjCrudBackFront/backend/.git/
```

- git add .



```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in mvn/wrapper/maven-wrapper.properties
```

- git commit -m "Project backend created"



```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend (master)
$ git commit -m "Project backend created"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

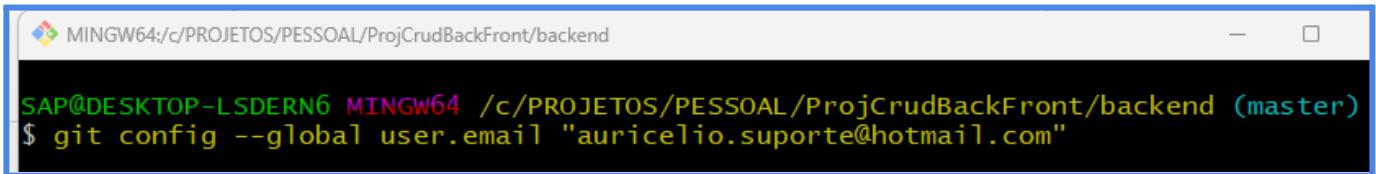
to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'SAP@DESKTOP-LSDERN6.(none)')
```

NOTA: Caso seja a primeira vez que sincronize no github no computador atual, irá pedir para informar o email e o nome da conta GitHub. Na próxima vez isso não irá aparecer.

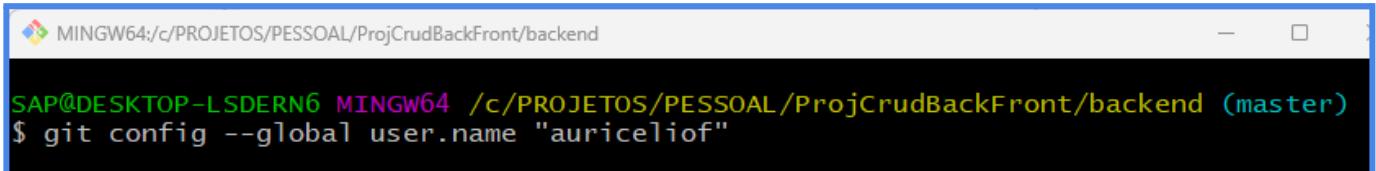
ProjCrudBackFront

- git config --global user.email "auricelio.suporte@hotmail.com"



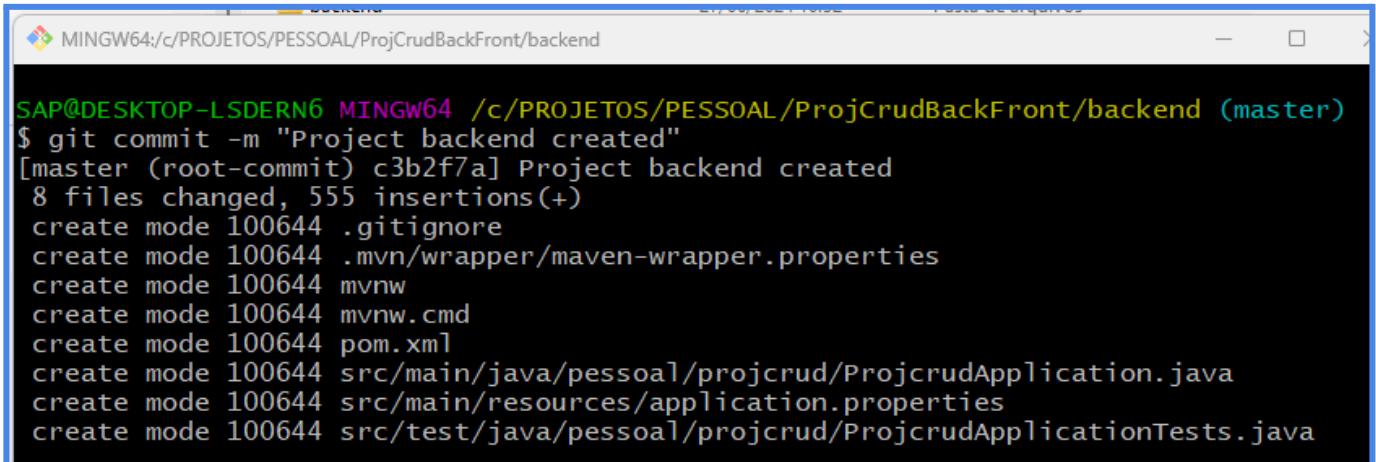
```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront/backend
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend (master)
$ git config --global user.email "auricelio.suporte@hotmail.com"
```

- git config --global user.name "auriceliof"



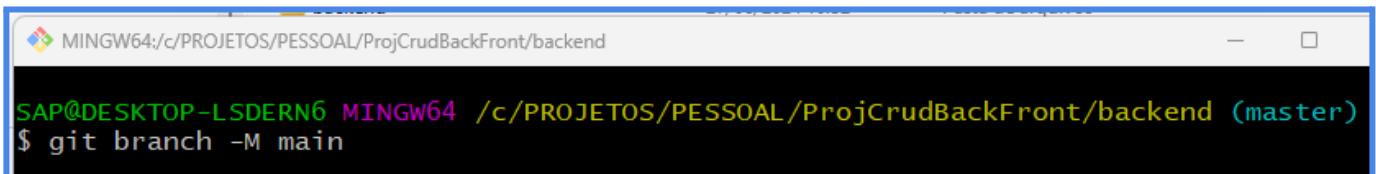
```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront/backend
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend (master)
$ git config --global user.name "auriceliof"
```

- git commit -m "Project backend created"



```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront/backend
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend (master)
$ git commit -m "Project backend created"
[master (root-commit) c3b2f7a] Project backend created
 8 files changed, 555 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .mvn/wrapper/maven-wrapper.properties
 create mode 100644 mvnw
 create mode 100644 mvnw.cmd
 create mode 100644 pom.xml
 create mode 100644 src/main/java/pessoal/projcrud/ProjcrudApplication.java
 create mode 100644 src/main/resources/application.properties
 create mode 100644 src/test/java/pessoal/projcrud/ProjcrudApplicationTests.java
```

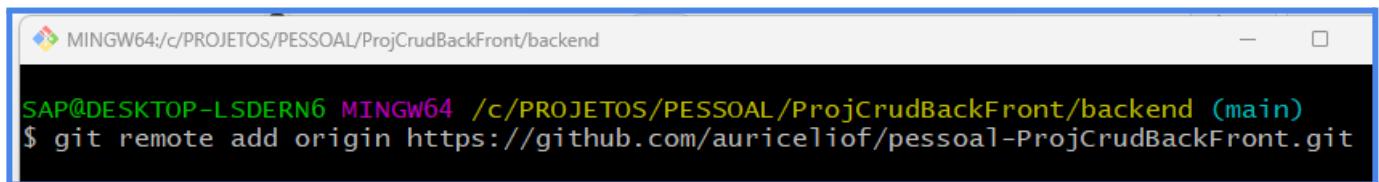
- git branch -M main



```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront/backend
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend (master)
$ git branch -M main
```

ProjCrudBackFront

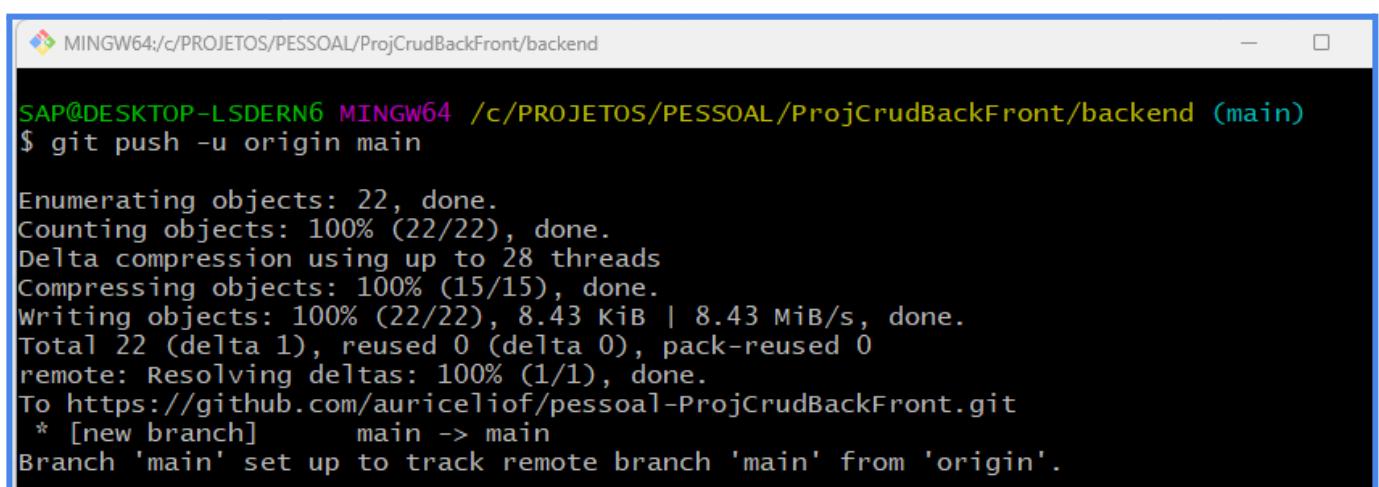
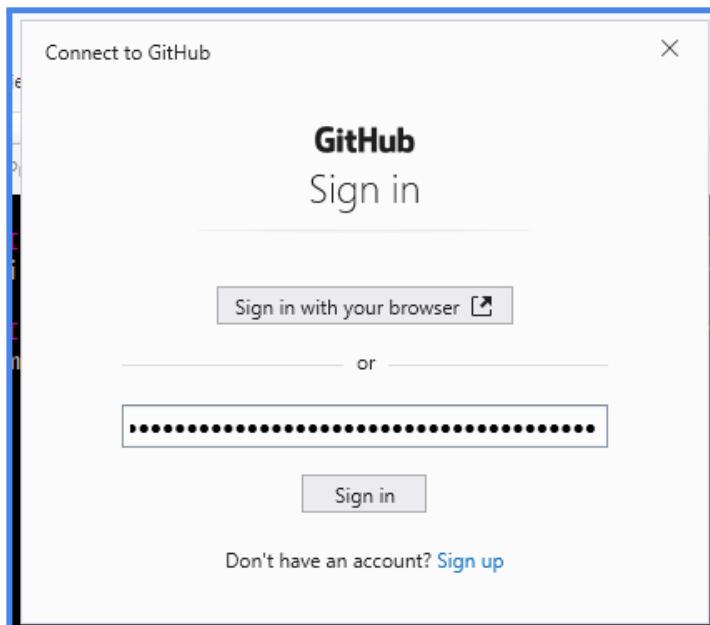
- git remote add origin https://github.com/auriceliof/pessoal-ProjCrudBackFront.git



```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend (main)
$ git remote add origin https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
```

- git push -u origin main

NOTA: Caso seja a primeira vez que sincronize no github no computador atual, irá pedir para realizar a autenticação no GitHub. Na próxima vez isso não irá aparecer.



```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront/backend (main)
$ git push -u origin main

Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 28 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (22/22), 8.43 KiB | 8.43 MiB/s, done.
Total 22 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

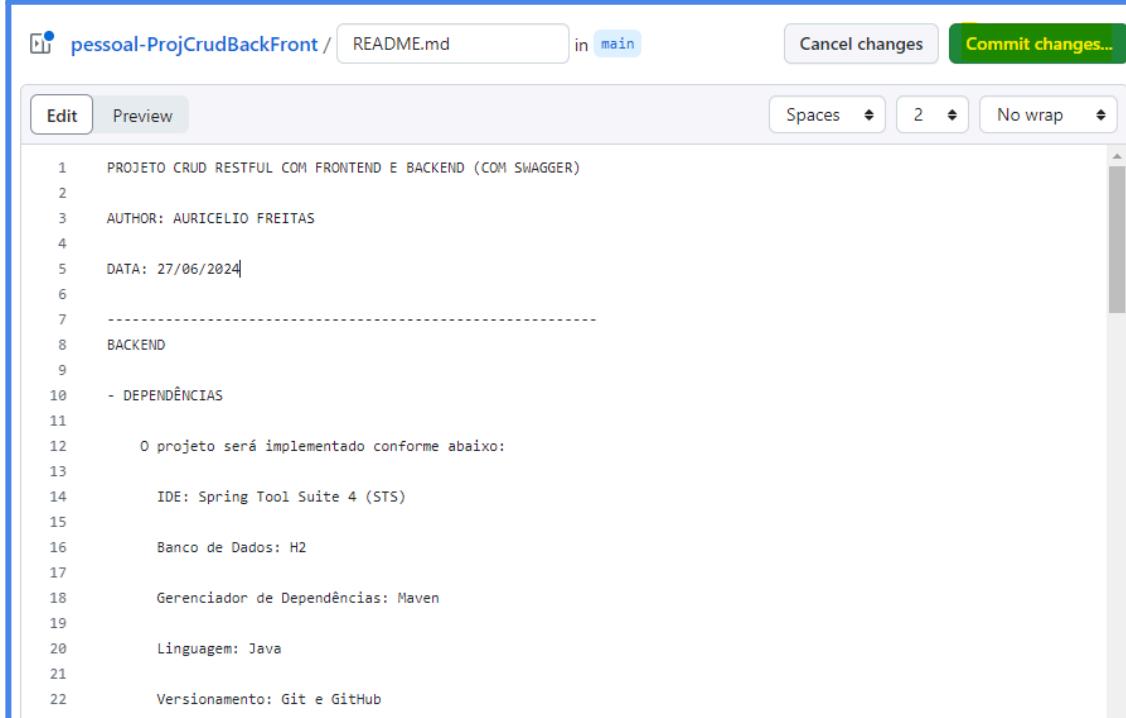
- Visualizar no GitHub (F5)

The screenshot shows a GitHub repository page. At the top, it displays the repository name "pessoal-ProjCrudBackFront" and the status "Public". To the right are buttons for "Pin" and "Unwatch". Below this, there are navigation links for "main", "1 Branch", and "0 Tags", along with search and file addition tools. A list of commits is shown, with the most recent one being "auriceliof Refactor" made 1 minute ago, which includes 2 commits. A folder named "backend" is also listed under the commit. On the left, there's a "README" section with a "Edit" button. In the center, there's a "Add a README" button with a book icon above it. Below the button, a subtitle says "Help people interested in this repository understand your project by adding a README."

Adicionar o README ao projeto

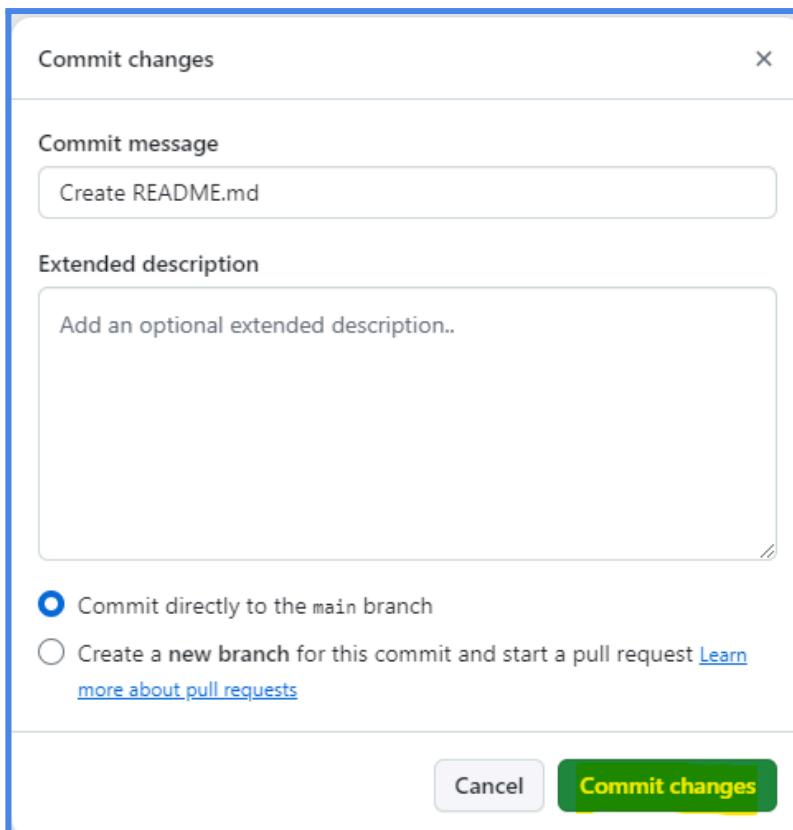
This screenshot shows a modal window titled "Add a README". It features a "README" section with an "Edit" button, a central area with a book icon, and a large "Add a README" button with a yellow gradient background. Below the button, a subtitle says "Help people interested in this repository understand your project by adding a README."

ProjCrudBackFront

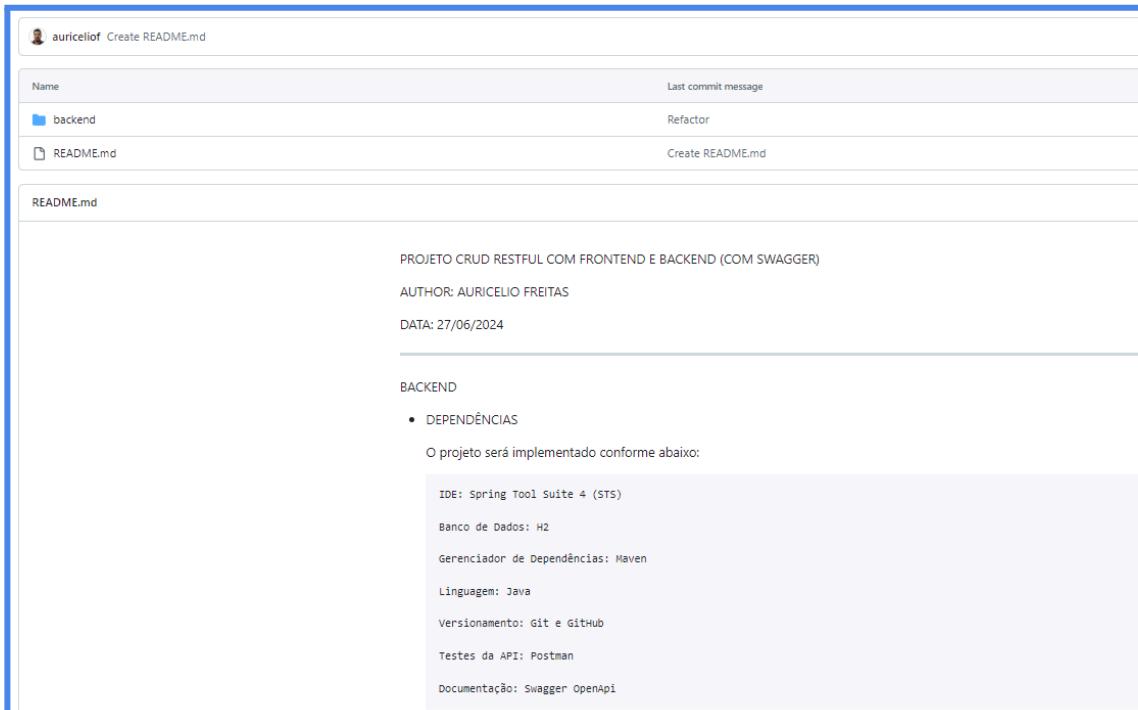


A screenshot of a GitHub commit interface. At the top, it shows the repository path "pessoal-ProjCrudBackFront / README.md" and a status "in main". There are buttons for "Cancel changes" and a prominent green "Commit changes..." button. Below this is a code editor window with the following content:

```
1 PROJETO CRUD RESTFUL COM FRONTEND E BACKEND (COM SWAGGER)
2
3 AUTHOR: AURICELIO FREITAS
4
5 DATA: 27/06/2024
6
7 -----
8 BACKEND
9
10 - DEPENDÊNCIAS
11
12 O projeto será implementado conforme abaixo:
13
14 IDE: Spring Tool Suite 4 (STS)
15
16 Banco de Dados: H2
17
18 Gerenciador de Dependências: Maven
19
20 Linguagem: Java
21
22 Versionamento: Git e GitHub
```

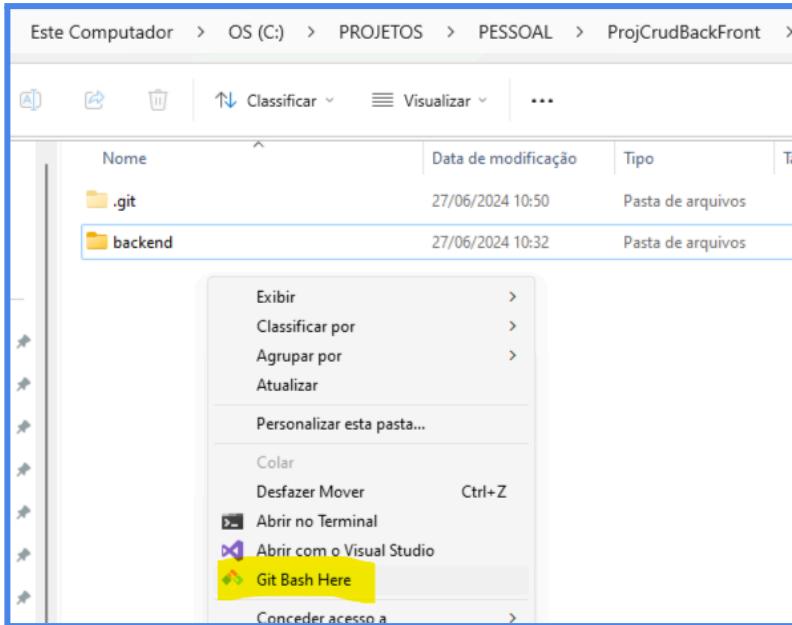


ProjCrudBackFront

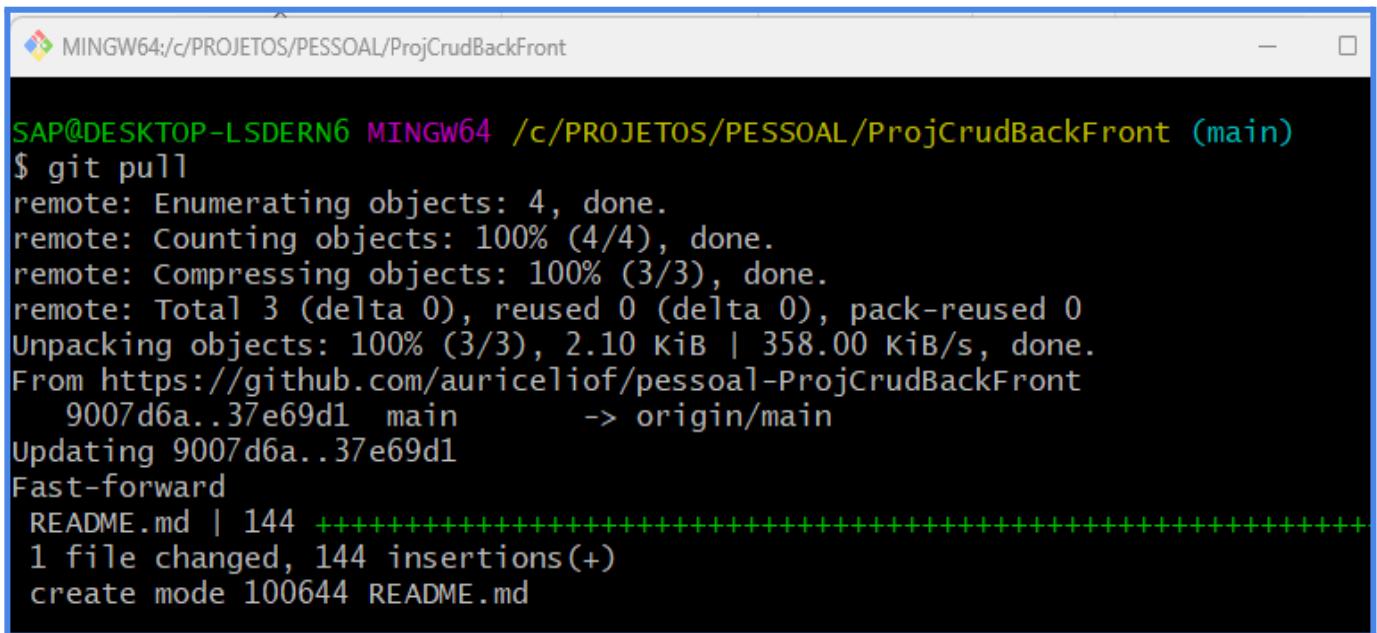


Atualizar o projeto local

- Botão direito no projeto e clicar em “Open Git bash here”



- git pull



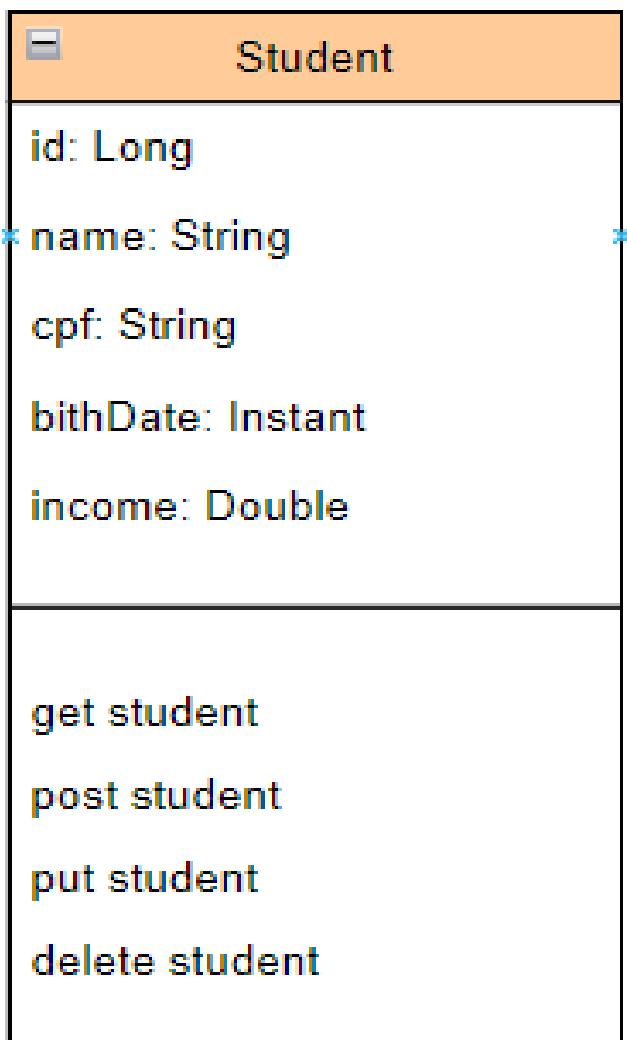
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)

```
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 2.10 KiB | 358.00 KiB/s, done.
From https://github.com/auriceliof/pessoal-ProjCrudBackFront
  9007d6a..37e69d1 main      -> origin/main
Updating 9007d6a..37e69d1
Fast-forward
 README.md | 144 ++++++
1 file changed, 144 insertions(+)
 create mode 100644 README.md
```

INICIAR O DESENVOLVIMENTO DO PROJETO

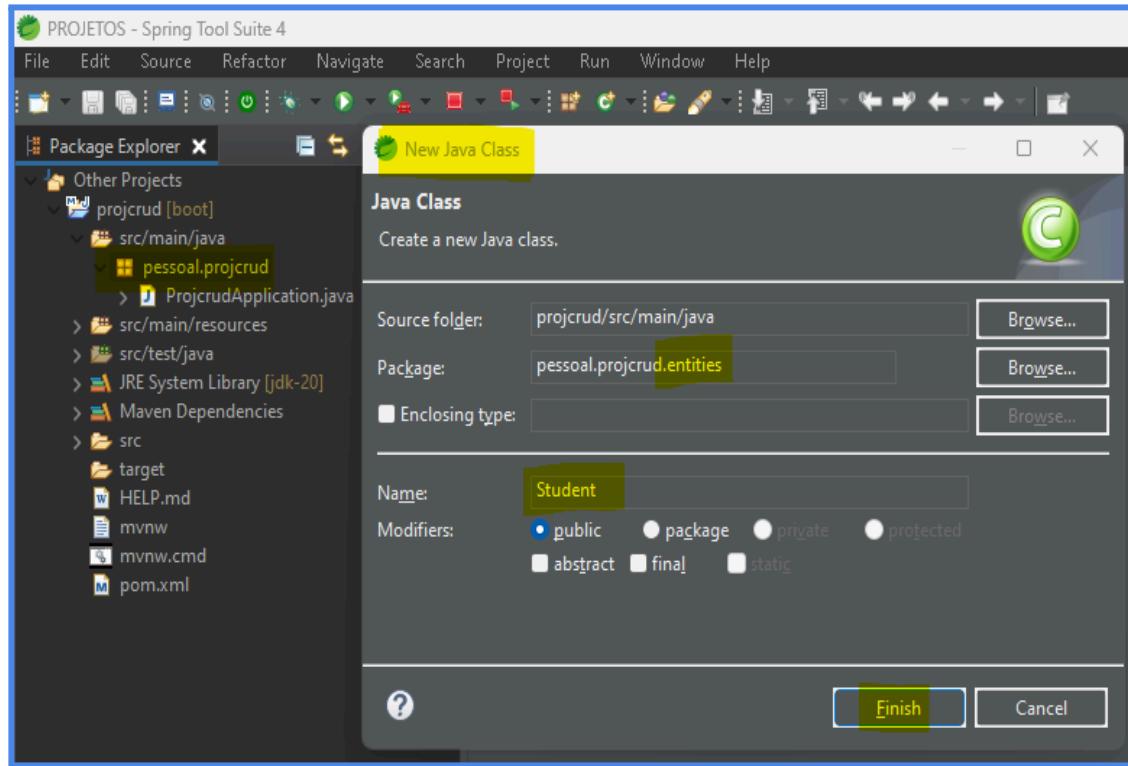
STUDENT_CLASSE

DIAGRAMA DE CLASSE



IMPLEMENTAR A ESTRUTURA E A CLASSE STUDENT NO STS

Criar a estrutura e classe



Definir o Serializable e os atributos da Classe

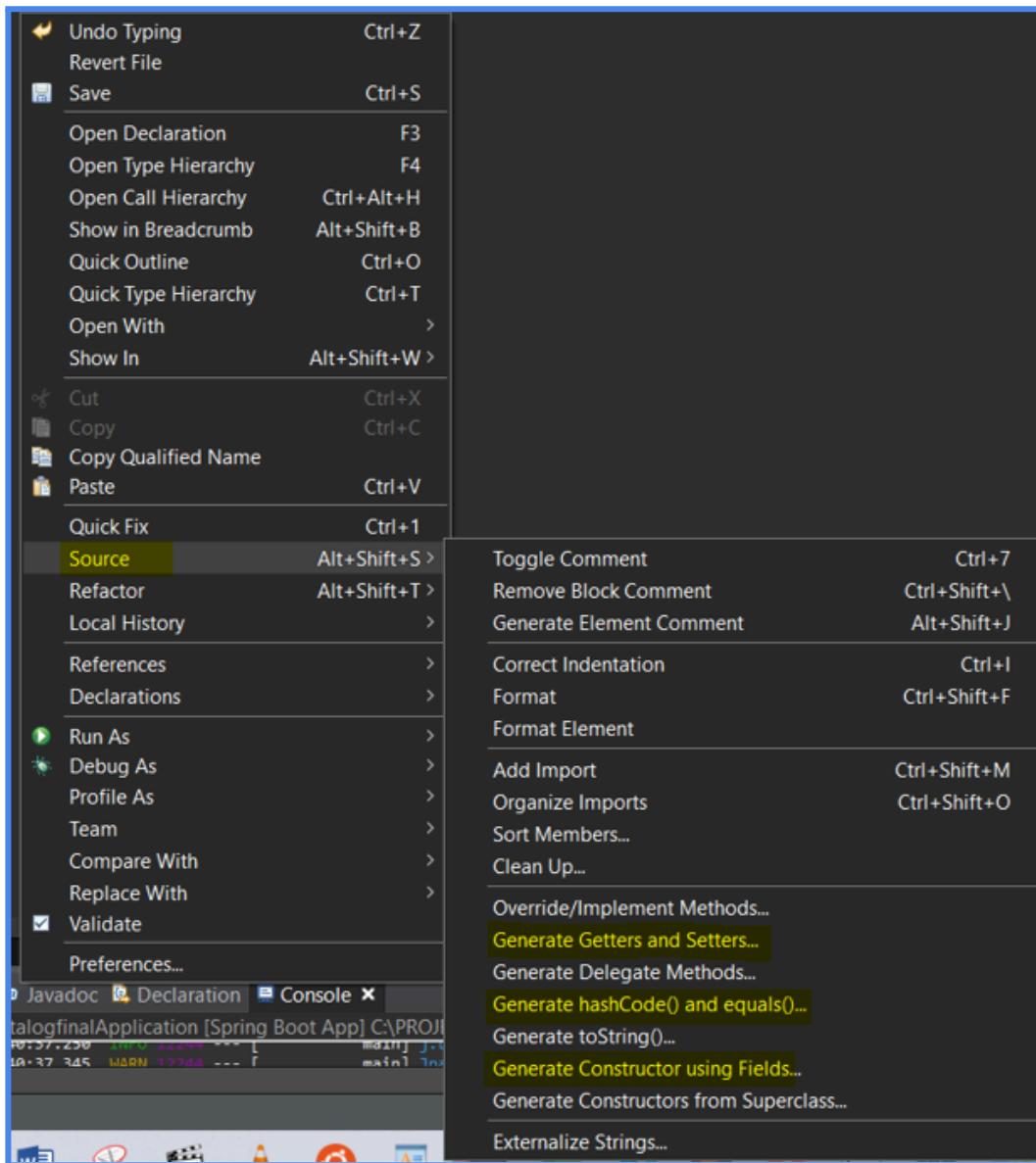
```
1 package pessoal.projcrud.entities;
2
3 import java.io.Serializable;
4 import java.time.LocalDate;
5
6 public class Student implements Serializable{
7     private static final long serialVersionUID = 1L;
8
9     private Long id;
10    private String name;
11    private String cpf;
12    private LocalDate birthDate;
13    private Double income;
14 }
```

The screenshot shows the code editor with the file 'Student.java' open. The code defines a public class 'Student' that implements the 'Serializable' interface. The class contains five private attributes: 'id', 'name', 'cpf', 'birthDate', and 'income', each annotated with the '@Column' annotation. The 'serialVersionUID' is set to 1L. The code editor interface is visible, showing the file path and line numbers.

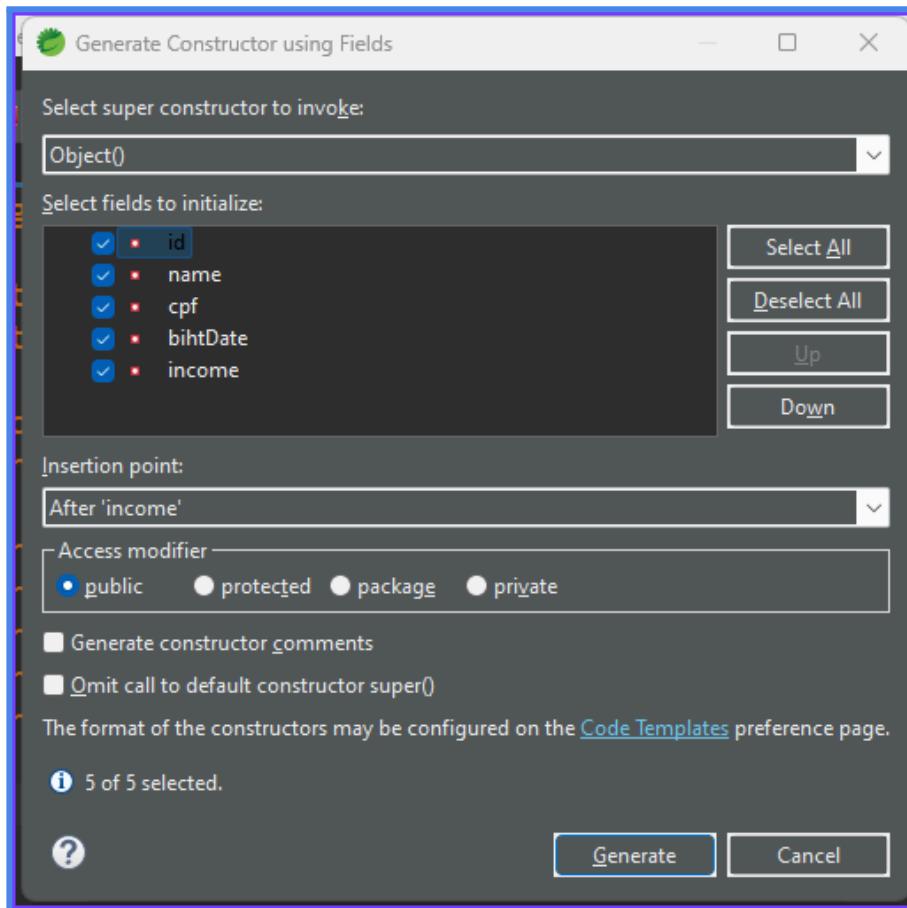
DICA:

Para criar os construtores, getters and setters e o hashCode, proceder conforme segue:

- Clicar com o botão direito aonde quer inserir
 - Source
 - “Escolher o método que deseja implementar”
 - Selecionar os itens
 - Clicar em: Generate



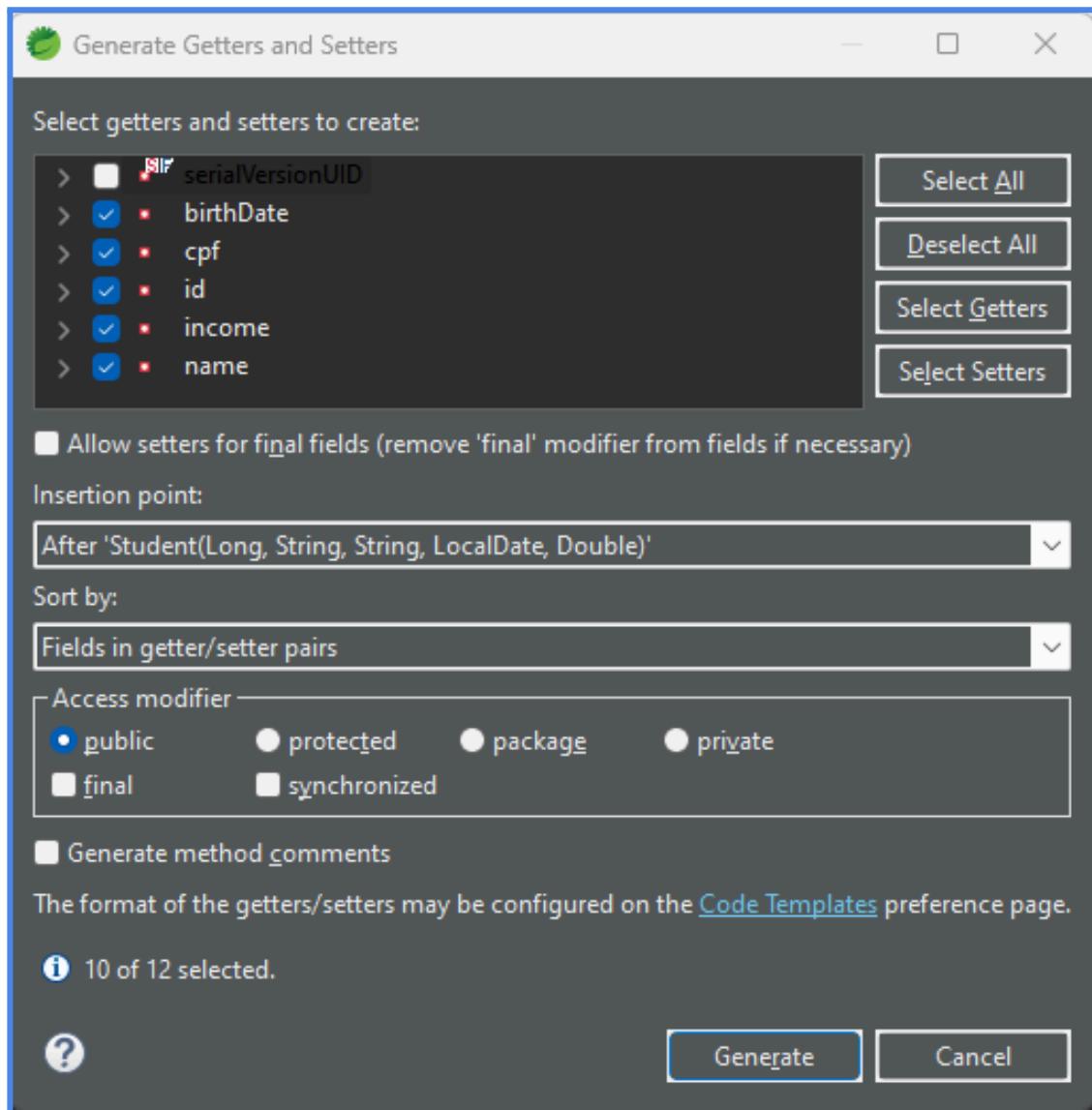
Criar o construtor



```
14
15*     public Student() {
16
17* }
18
19*     public Student(Long id, String name, String cpf, LocalDate birthDate, Double income) {
20*         super();
21*         this.id = id;
22*         this.name = name;
23*         this.cpf = cpf;
24*         this.birthDate = birthDate;
25*         this.income = income;
26*     }
27 }
```

NOTA: Por melhores práticas, também criaremos um construtor vazio.

Criar os Getters and Setters



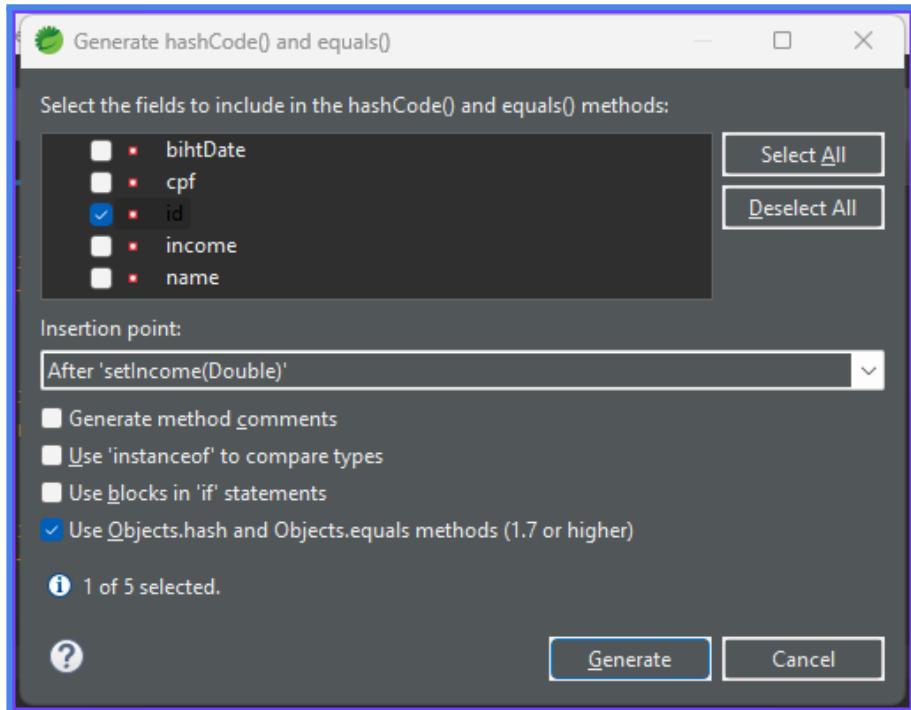
The screenshot shows a Java code editor with a dark theme. The file is named *Student.java. The code defines a class with four methods: getId(), setId(Long id), getName(), and setName(String name). The code is color-coded, with keywords in blue, identifiers in orange, and comments in green.

```
23  
24•     public Long getId() {  
25         return id;  
26     }  
27  
28•     public void setId(Long id) {  
29         this.id = id;  
30     }  
31  
32•     public String getName() {  
33         return name;  
34     }  
35  
36•     public void setName(String name) {  
37         this.name = name;  
38     }  
39
```

The screenshot shows the same Java code editor with the *Student.java file. It now includes additional methods: getCpf(), setCpf(String cpf), getBirthDate(), setBirthDate(LocalDate birthDate), getIncome(), and setIncome(Double income). The code is color-coded, with keywords in blue, identifiers in orange, and comments in green.

```
45  
44•     public String getCpf() {  
45         return cpf;  
46     }  
47  
48•     public void setCpf(String cpf) {  
49         this.cpf = cpf;  
50     }  
51  
52•     public LocalDate getBirthDate() {  
53         return birthDate;  
54     }  
55  
56•     public void setBirthDate(LocalDate birthDate) {  
57         this.birthDate = birthDate;  
58     }  
59  
60•     public Double getIncome() {  
61         return income;  
62     }  
63  
64•     public void setIncome(Double income) {  
65         this.income = income;  
66     }  
67 }
```

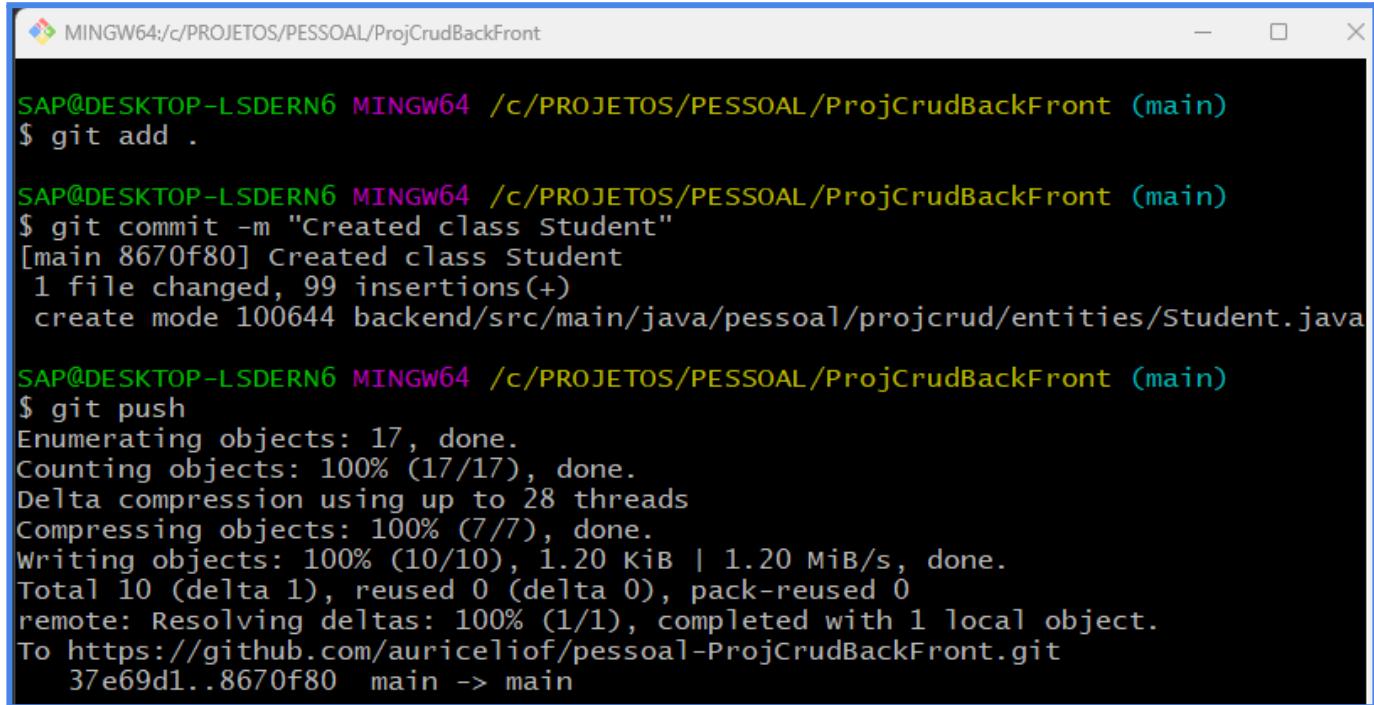
Criar o hashCode and equals



```
*Student.java
64
65
66 @Override
67 public int hashCode() {
68     return Objects.hash(id);
69 }
70
71 @Override
72 public boolean equals(Object obj) {
73     if (this == obj)
74         return true;
75     if (obj == null)
76         return false;
77     if (getClass() != obj.getClass())
78         return false;
79     Student other = (Student) obj;
80     return Objects.equals(id, other.id);
81 }
```

Github-2

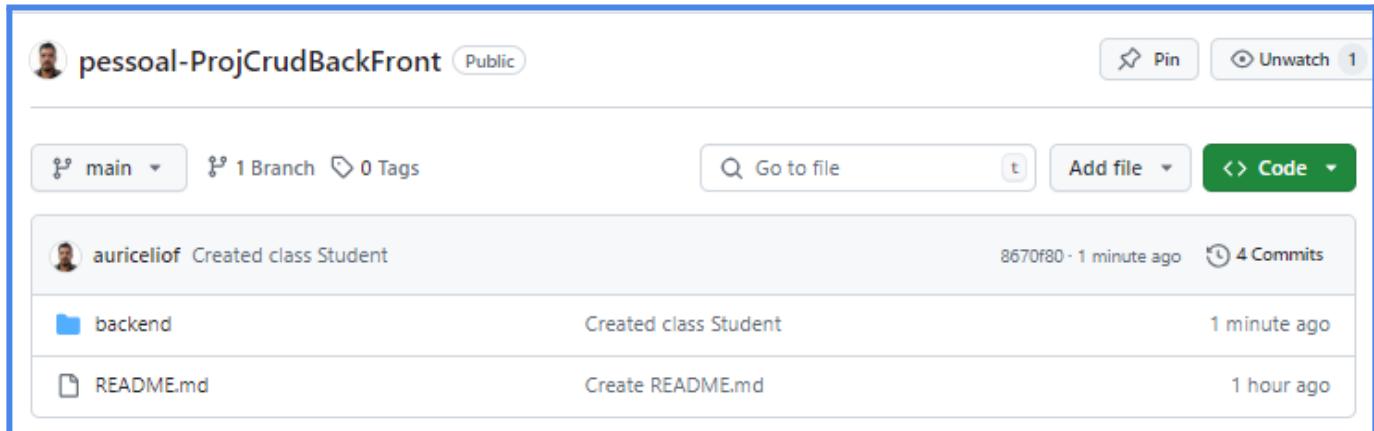
- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Created class Student”
 - git push



```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git add .

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Created class Student"
[main 8670f80] Created class Student
 1 file changed, 99 insertions(+)
 create mode 100644 backend/src/main/java/pessoal/projcrud/entities/Student.java

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 28 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 1.20 KiB | 1.20 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
 37e69d1..8670f80 main -> main
```



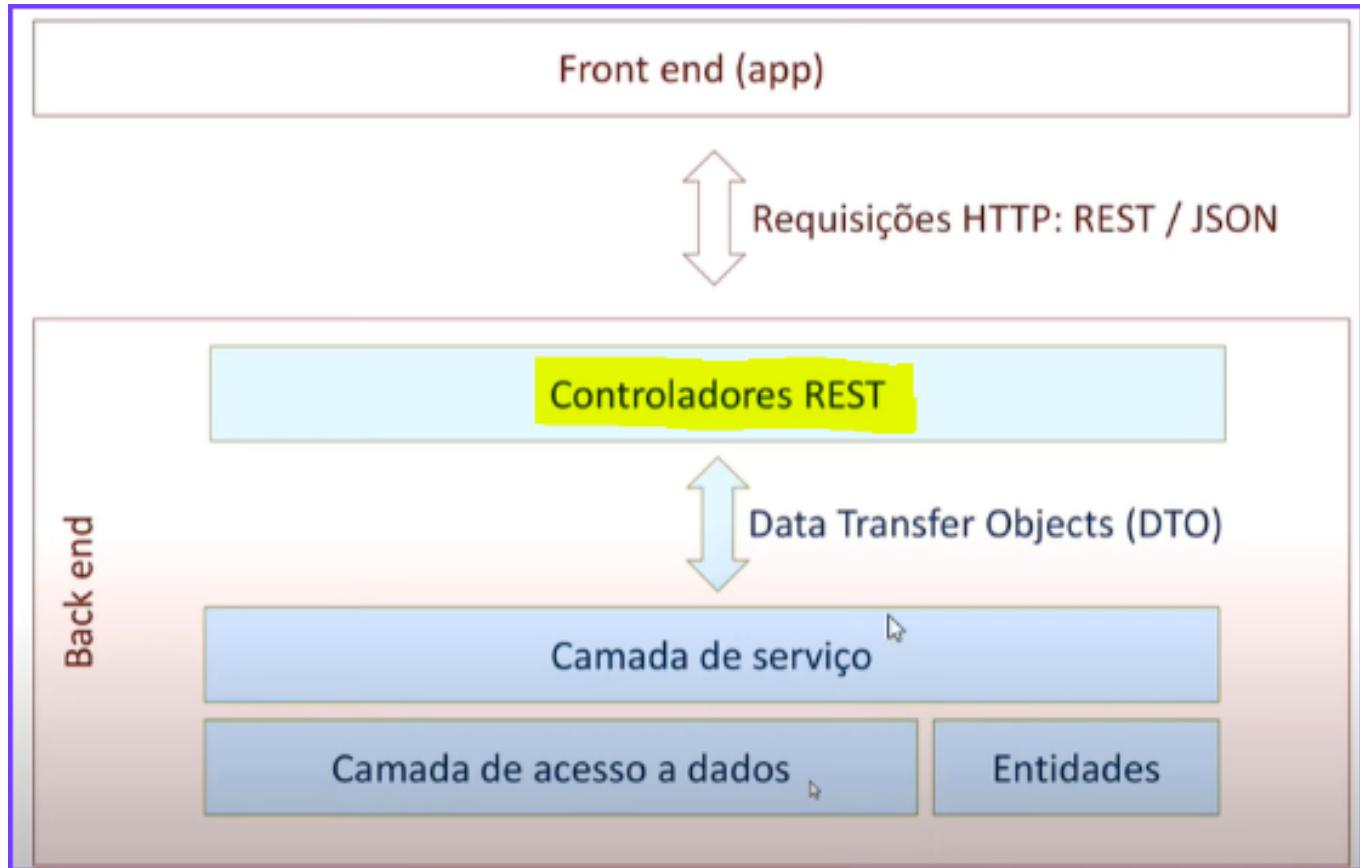
The screenshot shows a GitHub repository page for the user 'pessoal'. The repository name is 'ProjCrudBackFront' and it is marked as 'Public'. At the top, there are buttons for 'Pin' and 'Unwatch'. Below that, there are dropdown menus for 'main' (branch), '1 Branch' (branch count), '0 Tags' (tag count), and a search bar with 'Go to file' and a 't' icon. To the right of the search bar are buttons for 'Add file' and 'Code'. The main content area displays three commits:

Author	Commit Message	Date	Commits
auriceliof	Created class Student	8670f80 · 1 minute ago	4 Commits
	backend	Created class Student	1 minute ago
	README.md	Create README.md	1 hour ago

STUDENT_CONTROLLER

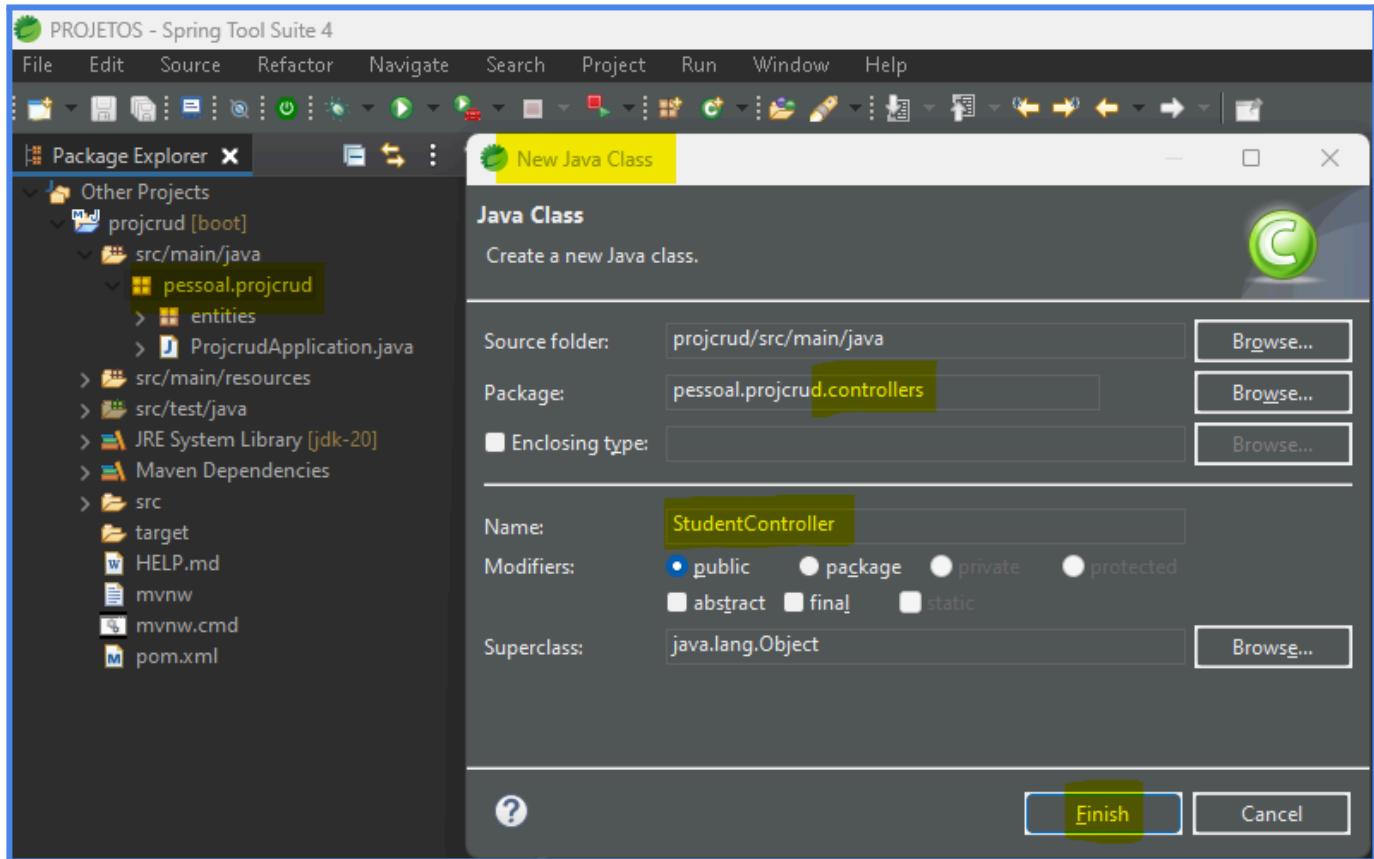
- O controlador é quem gerencia as requisições, podendo ser chamado de Controller ou Resource. É onde implantamos nossos endpoints.

CONCEITUAL



IMPLEMENTAR A ESTRUTURA

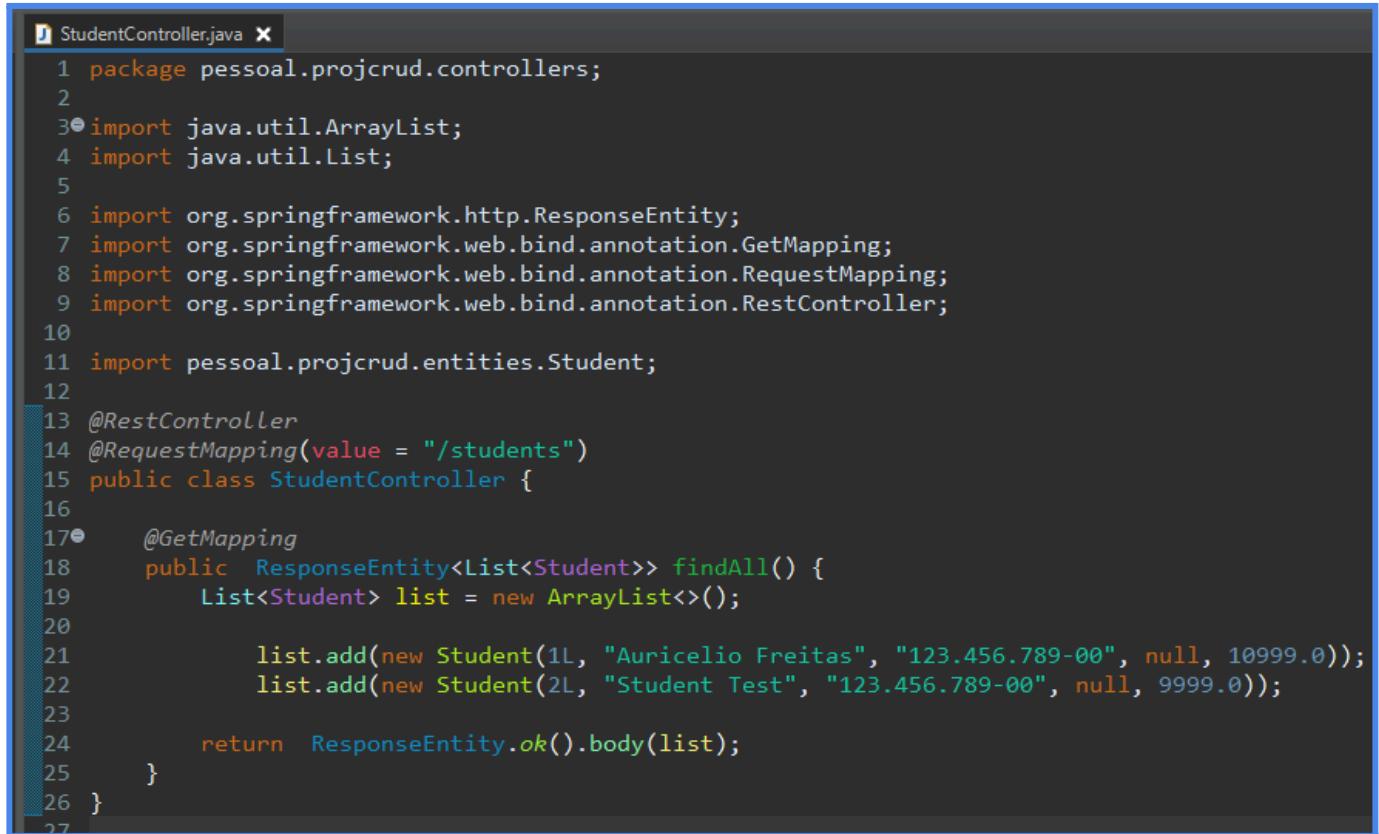
Criar a estrutura e o recurso StudentController



Implementar as Notações Rest

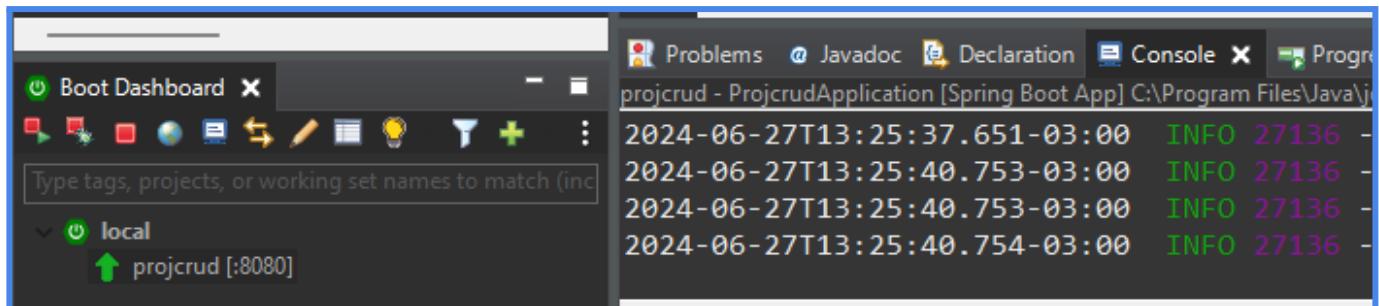
```
1 package pessoal.projcrud.controllers;
2
3 import org.springframework.web.bind.annotation.RequestMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 @RequestMapping(value = "/students")
8 public class StudentController {
```

Criar o Endpoint findAll para teste

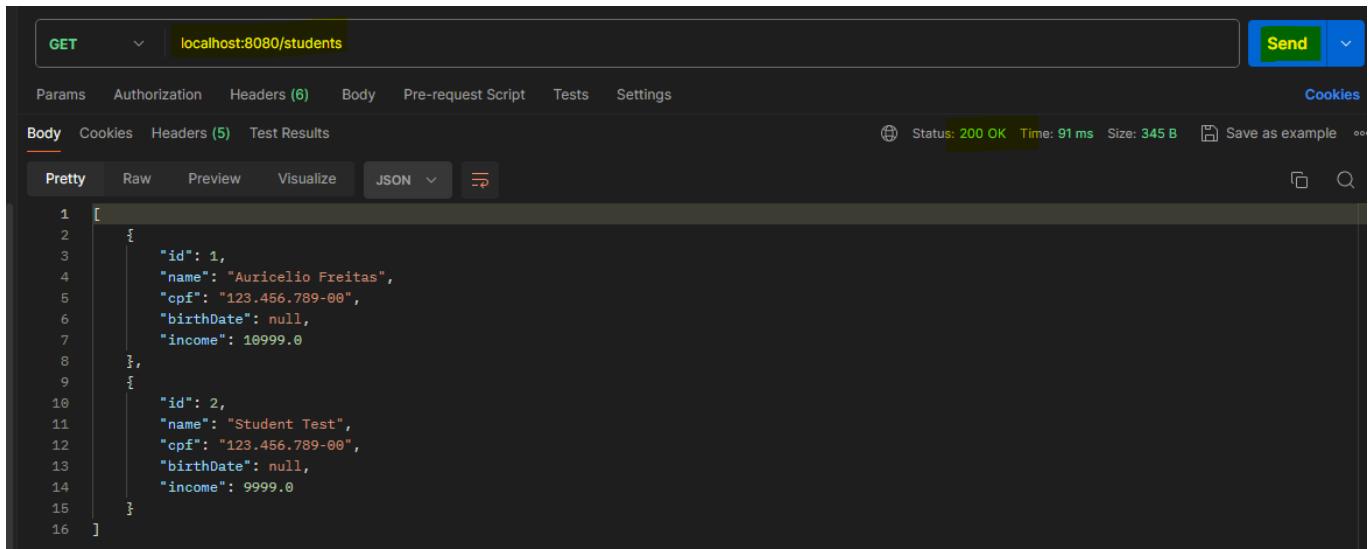


```
StudentController.java
1 package pessoal.projcrud.controllers;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RestController;
10
11 import pessoal.projcrud.entities.Student;
12
13 @RestController
14 @RequestMapping(value = "/students")
15 public class StudentController {
16
17     @GetMapping
18     public ResponseEntity<List<Student>> findAll() {
19         List<Student> list = new ArrayList<>();
20
21         list.add(new Student(1L, "Auricelio Freitas", "123.456.789-00", null, 10999.0));
22         list.add(new Student(2L, "Student Test", "123.456.789-00", null, 9999.0));
23
24         return ResponseEntity.ok().body(list);
25     }
26 }
27
```

Rodar o projeto



Testar com o Postman



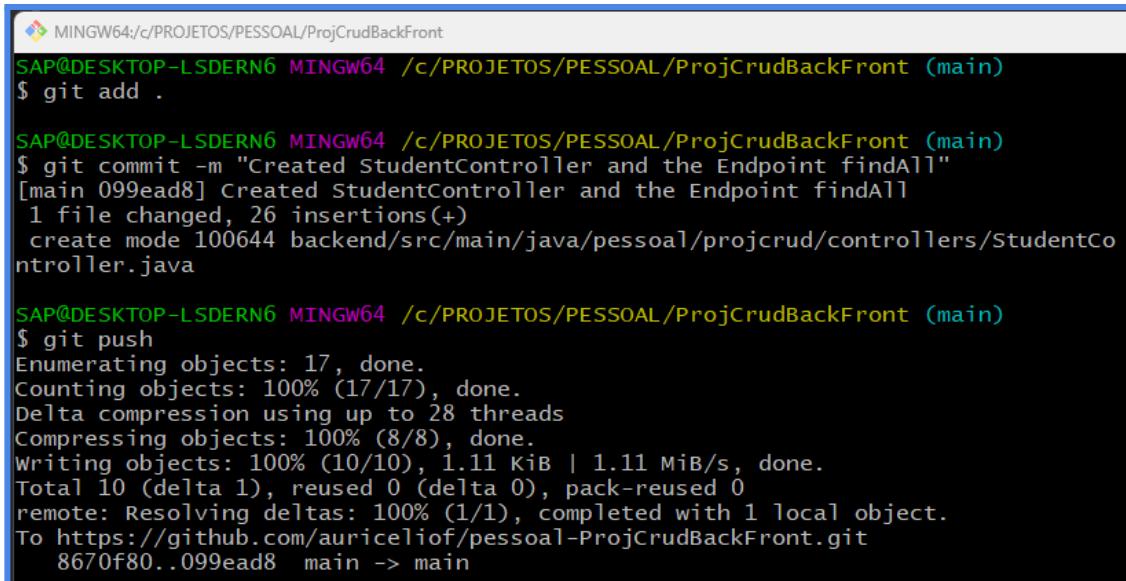
The screenshot shows the Postman interface with a successful API call. The URL is `localhost:8080/students`. The response body is a JSON array containing two student objects:

```
1 [  
2   {  
3     "id": 1,  
4     "name": "Auricelio Freitas",  
5     "cpf": "123.456.789-00",  
6     "birthDate": null,  
7     "income": 10999.0  
8   },  
9   {  
10    "id": 2,  
11    "name": "Student Test",  
12    "cpf": "123.456.789-00",  
13    "birthDate": null,  
14    "income": 9999.0  
15  }  
16 ]
```

The status bar indicates `Status: 200 OK Time: 91 ms Size: 345 B`.

Github-3

- “Git bash here” no diretório do projeto
 - `git add .`
 - `git commit -m “Created StudentController and the Endpoint findAll”`
 - `git push`

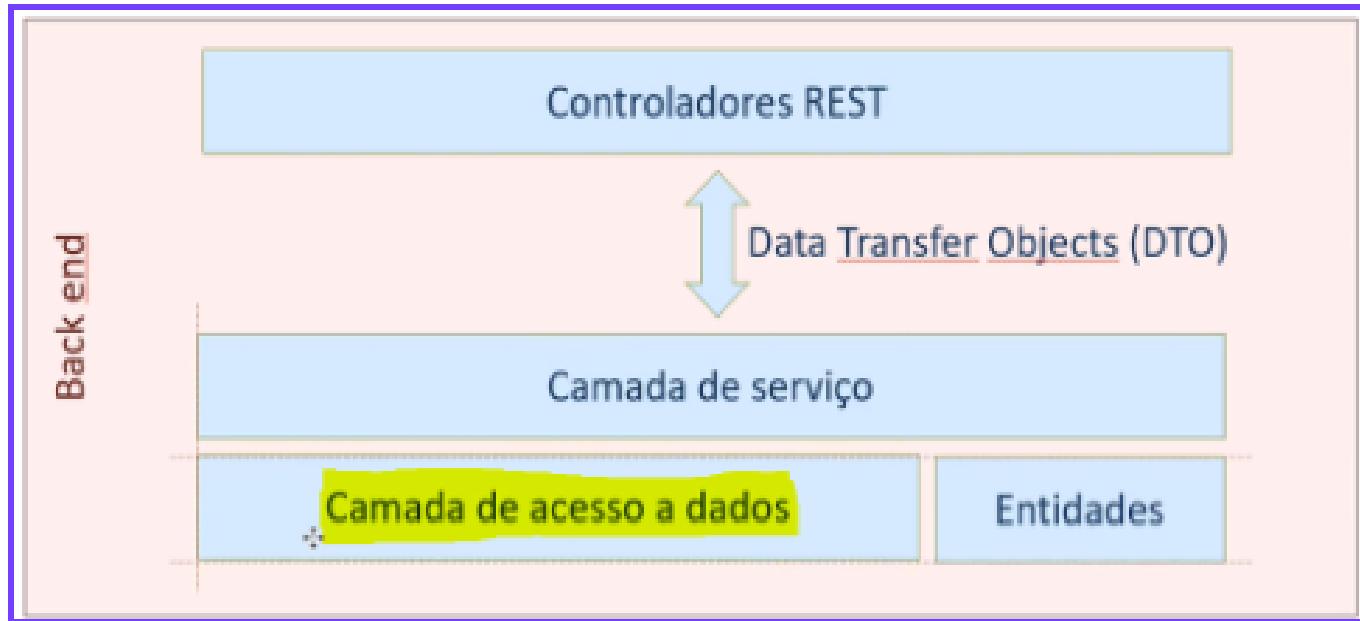


```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront  
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)  
$ git add .  
  
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)  
$ git commit -m "Created StudentController and the Endpoint findAll"  
[main 099ead8] Created StudentController and the Endpoint findAll  
1 file changed, 26 insertions(+)  
 create mode 100644 backend/src/main/java/pessoal/projcrud/controllers/StudentController.java  
  
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)  
$ git push  
Enumerating objects: 17, done.  
Counting objects: 100% (17/17), done.  
Delta compression using up to 28 threads  
Compressing objects: 100% (8/8), done.  
Writing objects: 100% (10/10), 1.11 KiB | 1.11 MiB/s, done.  
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git  
 8670f80..099ead8 main -> main
```

STUDENT_REPOSITORY

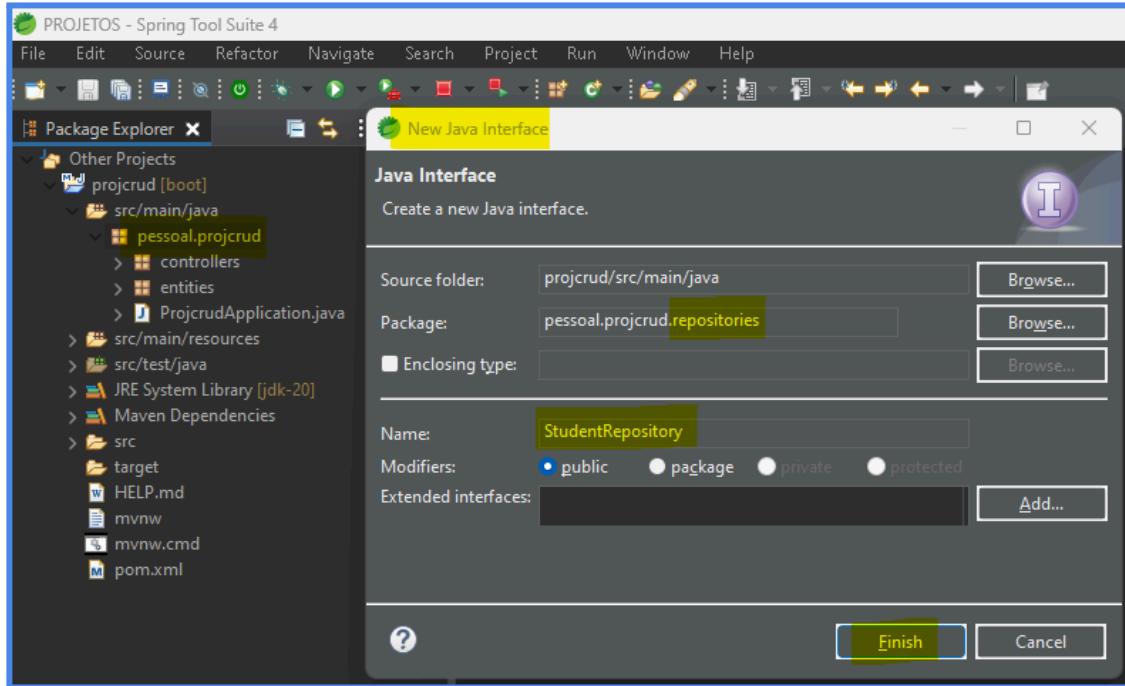
- Iremos implementar a “Camada de acesso a dados”, chamada de Repositories. É a camada responsável pelo acesso ao banco.

CONCEITUAL



IMPLEMENTAR A ESTRUTURA E O STUDENT_REPOSITORY

Criar a estrutura e a interface de acesso ao banco



Implementar a notação e estender a JPA

```
1 package pessoal.projcrud.repositories;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import pessoal.projcrud.entities.Student;
7
8 @Repository
9 public interface StudentRepository extends JpaRepository<Student, Long>{
10
11 }
```

NOTA: Apenas ao estender o `JpaRepository`, o `spring` já nos fornece vários métodos de manipulação aos dados com o banco.

Github-4

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Created StudentRepository”
 - git push

The screenshot shows a terminal window titled "MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront". The terminal output is as follows:

```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git add .

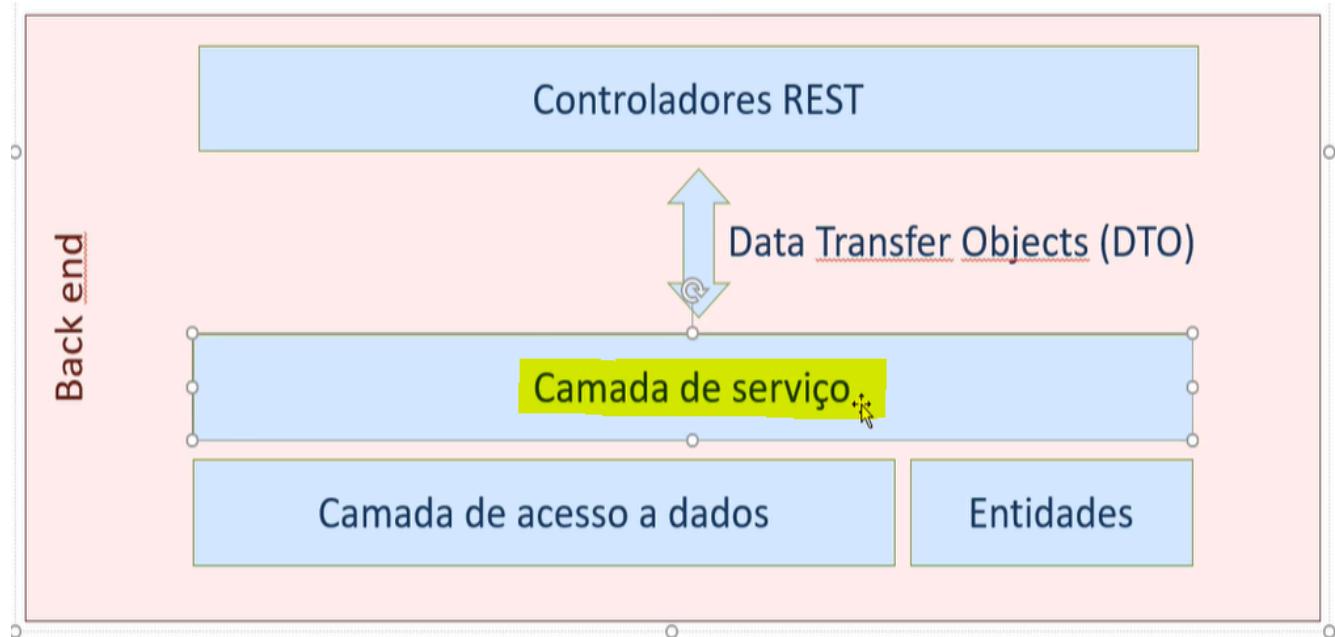
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Created StudentRepository"
[main 7411f42] Created StudentRepository
 1 file changed, 11 insertions(+)
 create mode 100644 backend/src/main/java/pessoal/projcrud/repositories/StudentRepository.java

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 28 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (10/10), 870 bytes | 870.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
 099ead8..7411f42 main -> main
```

STUDENT_SERVICE

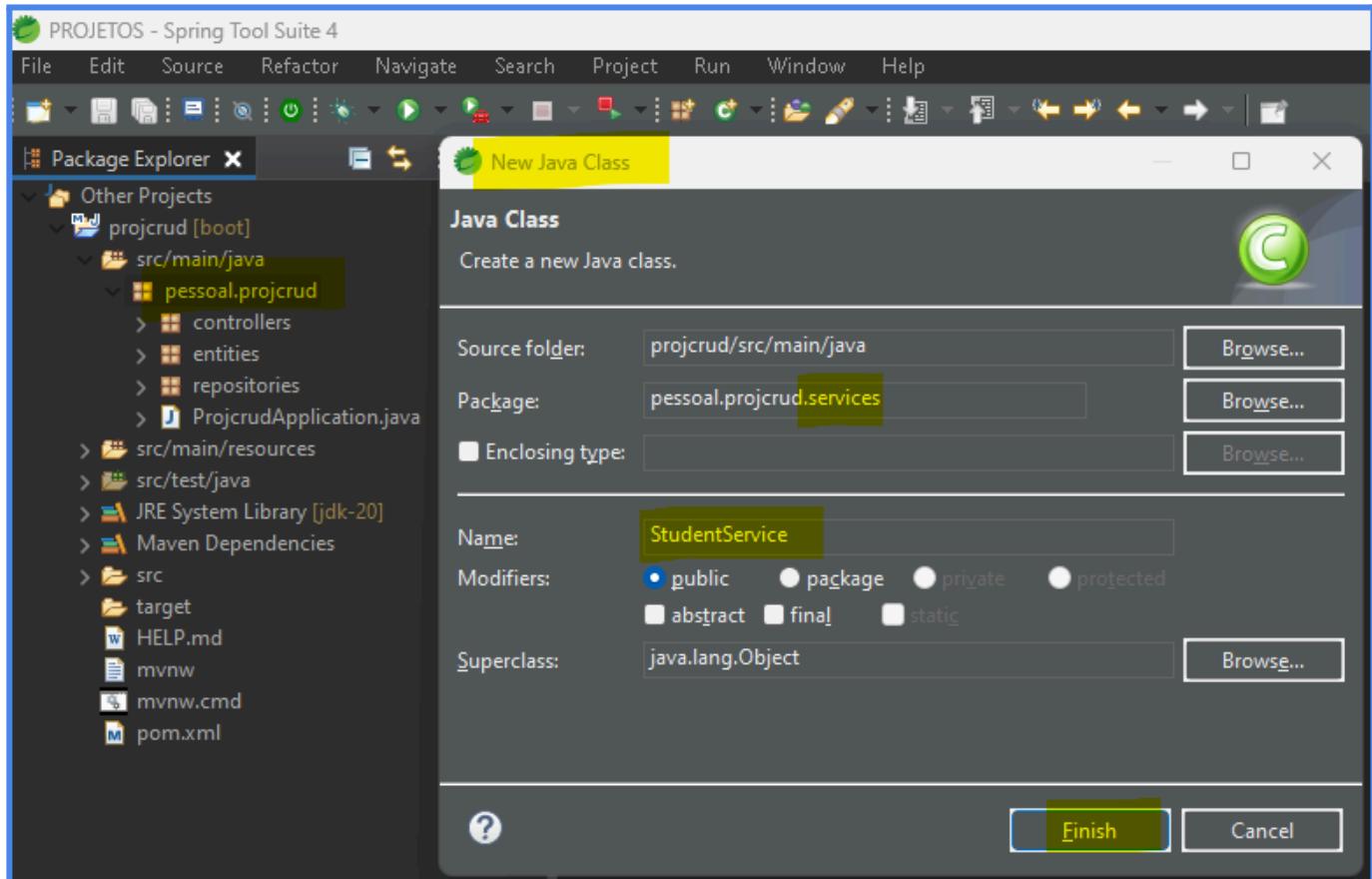
- Iremos implementar a “Camada de serviço”, onde concentraremos as regras de negócio e toda a lógica do projeto.

CONCEITUAL



IMPLEMENTAR A ESTRUTURA E O STUDENT_SERVICE

Criar a estrutura e a classe de serviço



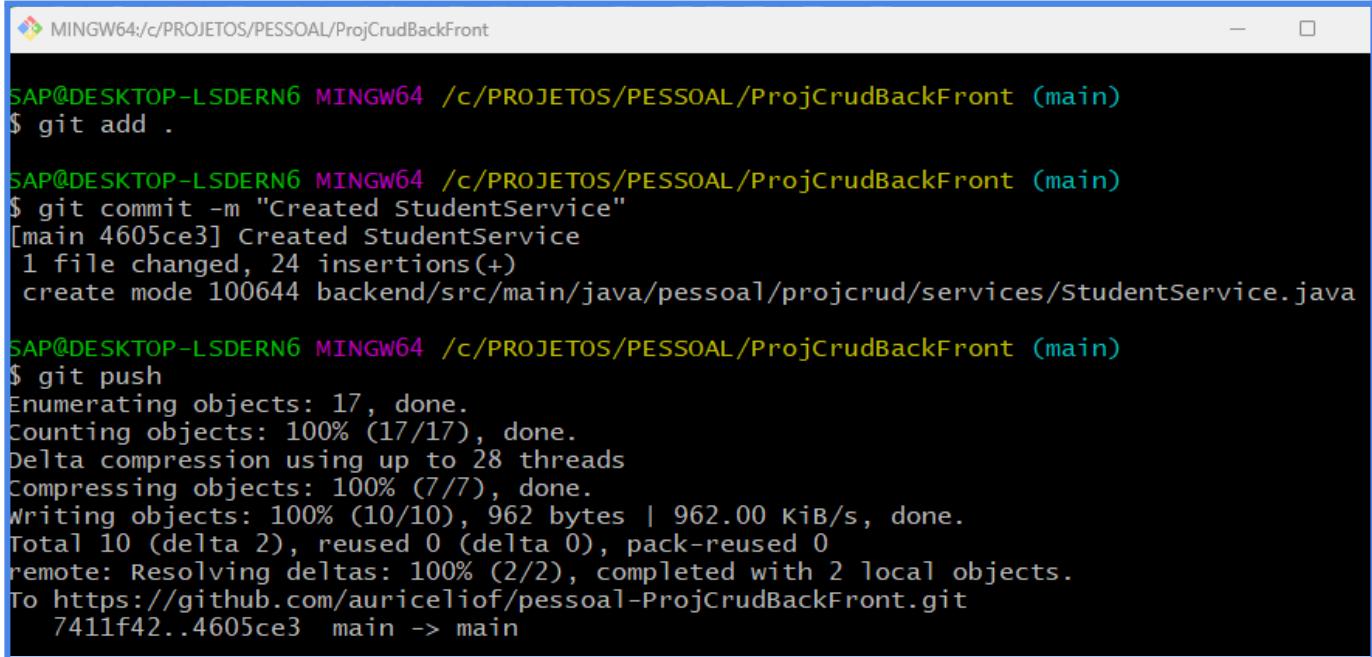
Implementar a lógica para o endpoint findAll

```
StudentService.java X
1 package pessoal.projcrud.services;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7 import org.springframework.transaction.annotation.Transactional;
8
9 import pessoal.projcrud.entities.Student;
10 import pessoal.projcrud.repositories.StudentRepository;
11
12 @Service
13 public class StudentService {
14
15     @Autowired
16     private StudentRepository repository;
17
18     @Transactional(readOnly = true)
19     public List<Student> findAll() {
20
21         return repository.findAll();
22     }
23 }
```

NOTA: O “`@Transactional`”, garante a integridade da transação de um método junto ao banco. No caso de pesquisa, utilizar o “`readOnly`” para evitar o locking desnecessário ao banco.

Github-5

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Created StudentService”
 - git push



The screenshot shows a terminal window titled "MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront". The terminal output is as follows:

```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git add .

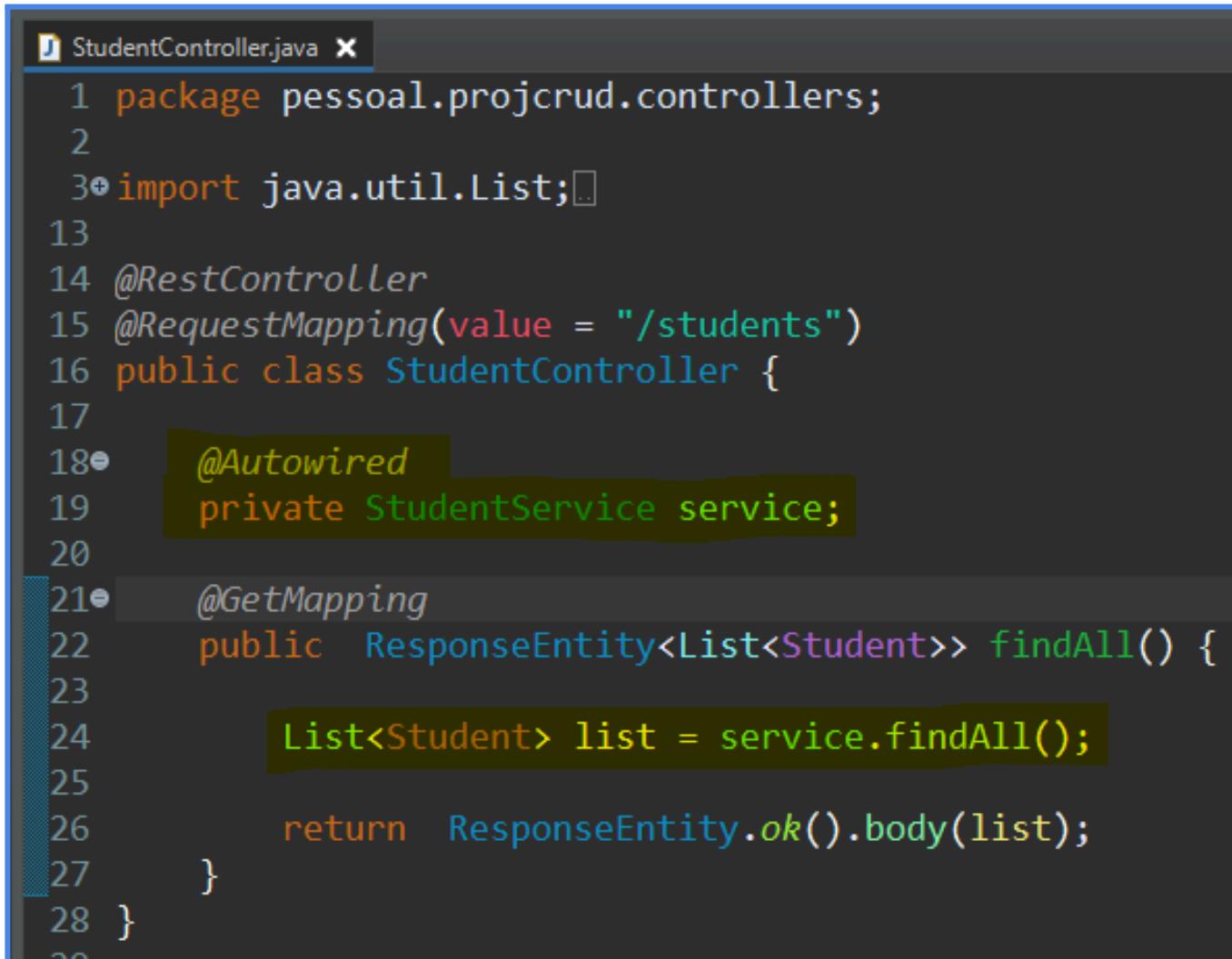
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Created StudentService"
[main 4605ce3] Created StudentService
 1 file changed, 24 insertions(+)
 create mode 100644 backend/src/main/java/pessoal/projcrud/services/StudentService.java

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 28 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 962 bytes | 962.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
 7411f42..4605ce3 main -> main
```

INTEGRAÇÃO COM O BANCO

AJUSTAR AS CAMADAS

Implementar o StudentController



The screenshot shows a code editor window with the file `StudentController.java` open. The code implements a REST controller for managing students. It uses annotations like `@RestController` and `@RequestMapping` to map URLs to methods. The `findAll()` method retrieves all students from the service and returns them as a response entity. The `service` field is injected via the `@Autowired` annotation.

```
1 package pessoal.projcrud.controllers;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping(value = "/students")
7 public class StudentController {
8
9     @Autowired
10    private StudentService service;
11
12    @GetMapping
13    public ResponseEntity<List<Student>> findAll() {
14
15        List<Student> list = service.findAll();
16
17        return ResponseEntity.ok().body(list);
18    }
19}
```

Implementar a classe Student

```
13
14 @Entity
15 @Table(name = "tb_student")
16 public class Student implements Serializable{
17     private static final long serialVersionUID = 1L;
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.IDENTITY)
21     private Long id;
22     private String name;
23     private String cpf;
24
25     @Column(columnDefinition = "TIMESTAMP WITH TIME ZONE")
26     private LocalDate birthDate;
27     private Double income;
28 }
```

NOTA: Ao instanciar as notações em questão, sempre escolher o pacote referente à especificação “jakarta.persistence”.

Rodar o projeto

```
2024-06-27T16:03:33.017-03:00 INFO 5580 --- [projcrud] [main] j. 2024-06-27T16:03:33.282-03:00 INFO 5580 --- [projcrud] [main] o. 2024-06-27T16:03:33.287-03:00 INFO 5580 --- [projcrud] [main] pe 2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o. 2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o. 2024-06-27T16:03:41.016-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
```

Testar com o Postman

```
GET localhost:8080/students
Status: 200 OK Time: 136 ms Size: 166 B
```

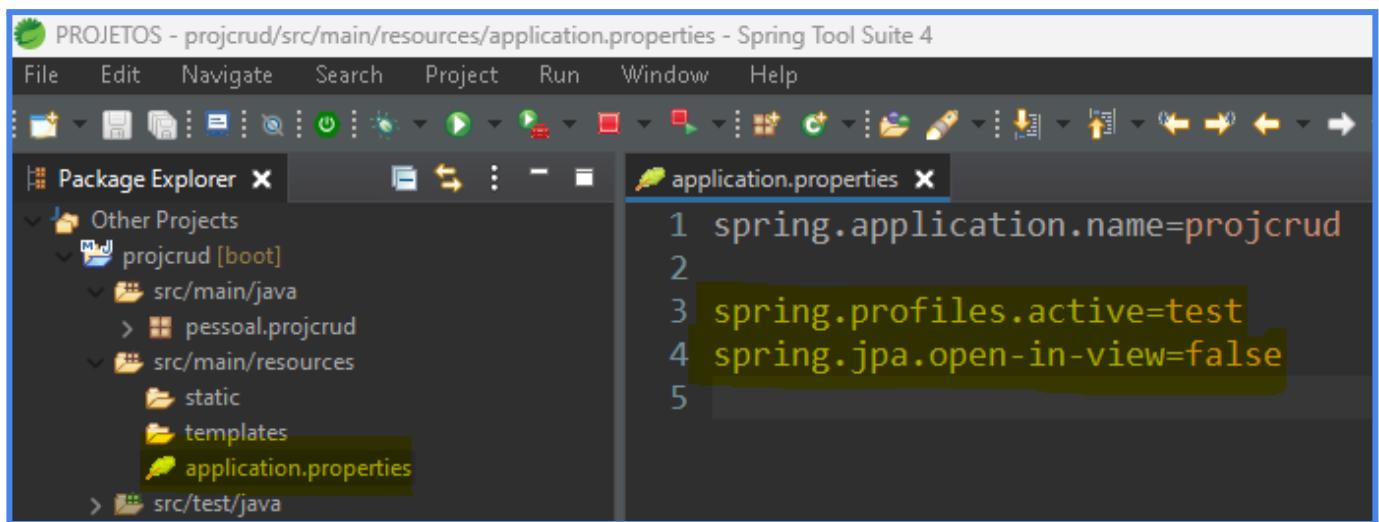
NOTA: Como não configuramos o banco ainda, nos é retornada, apenas, uma lista vazia.

BANCO H2

Configurar o perfil de teste no application.properties

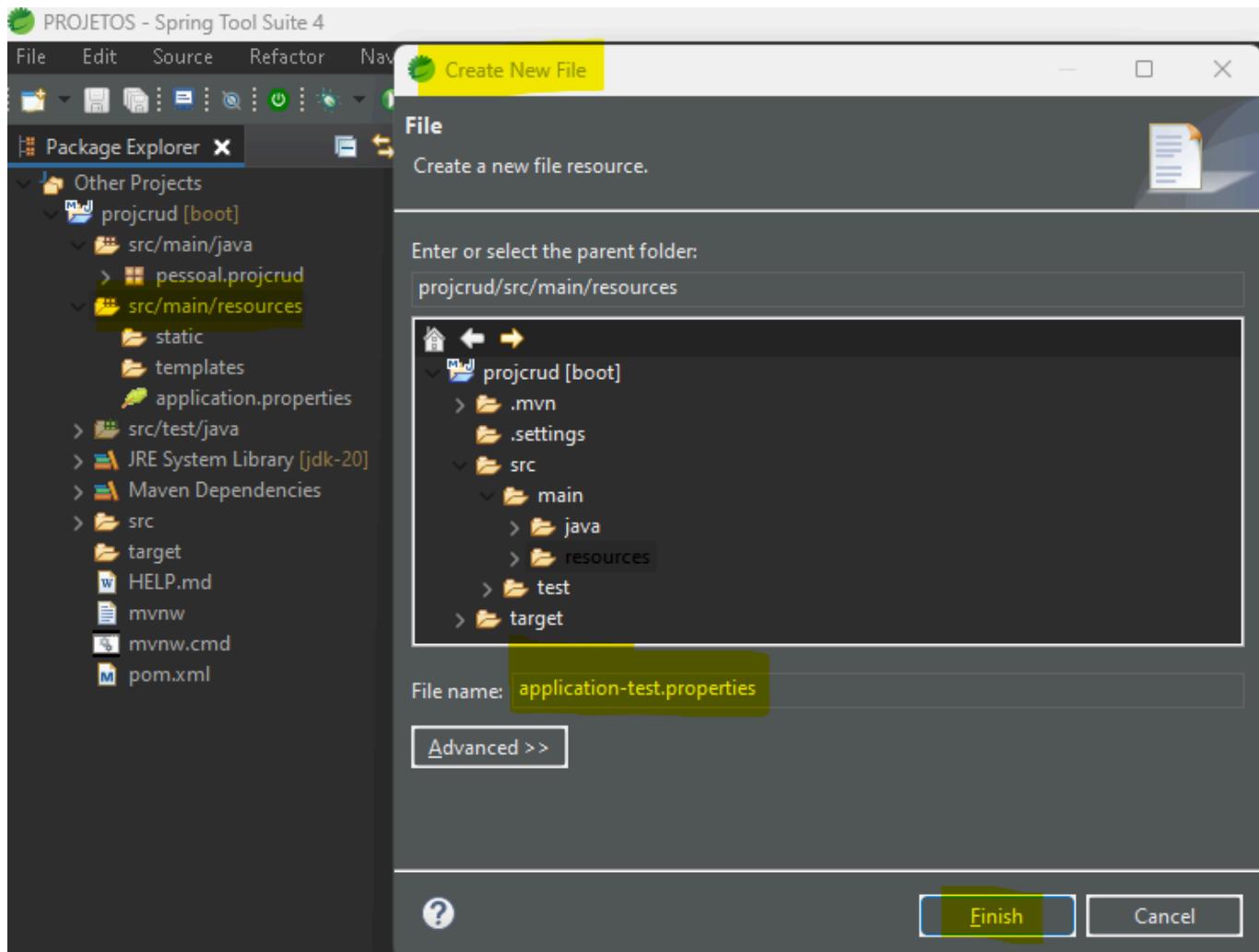
Perfil de teste

```
spring.profiles.active=test  
spring.jpa.open-in-view=false
```



NOTA: O “`spring.jpa.open-in-view`”, faz com que as transações ao banco com JPA sejam encerradas na camada de Serviço. Não passando para a camada de Controle.

Criar o arquivo de configuração application-test.properties



Implementar o application-test.properties

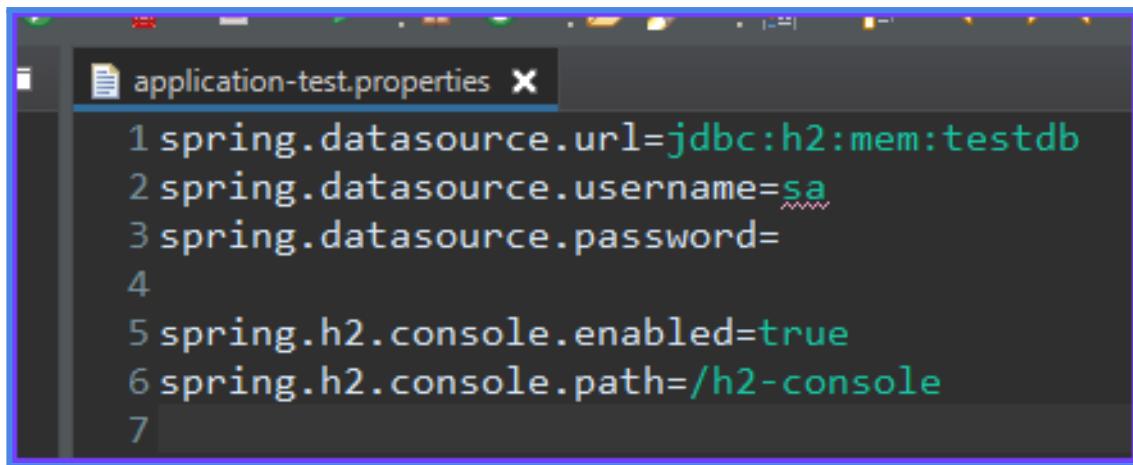
```
spring.datasource.url=jdbc:h2:mem:testdb
```

```
spring.datasource.username=sa
```

```
spring.datasource.password=
```

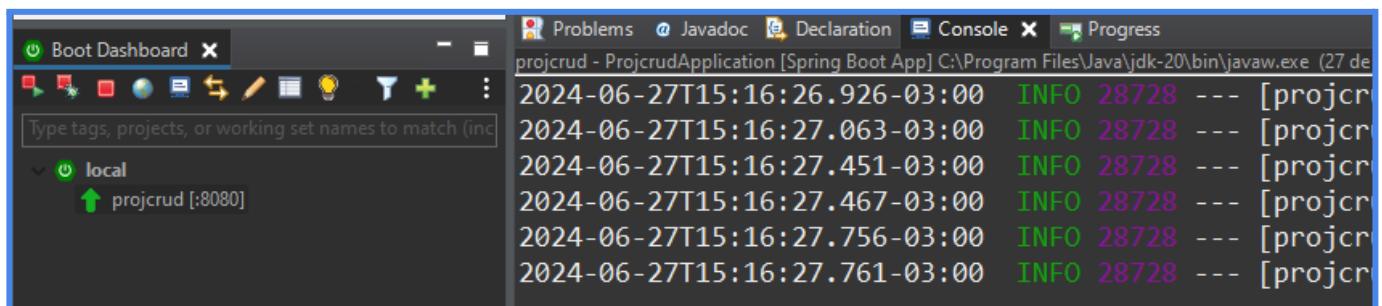
```
spring.h2.console.enabled=true
```

```
spring.h2.console.path=/h2-console
```



```
application-test.properties
1 spring.datasource.url=jdbc:h2:mem:testdb
2 spring.datasource.username=sa
3 spring.datasource.password=
4
5 spring.h2.console.enabled=true
6 spring.h2.console.path=/h2-console
7
```

Rodar o projeto

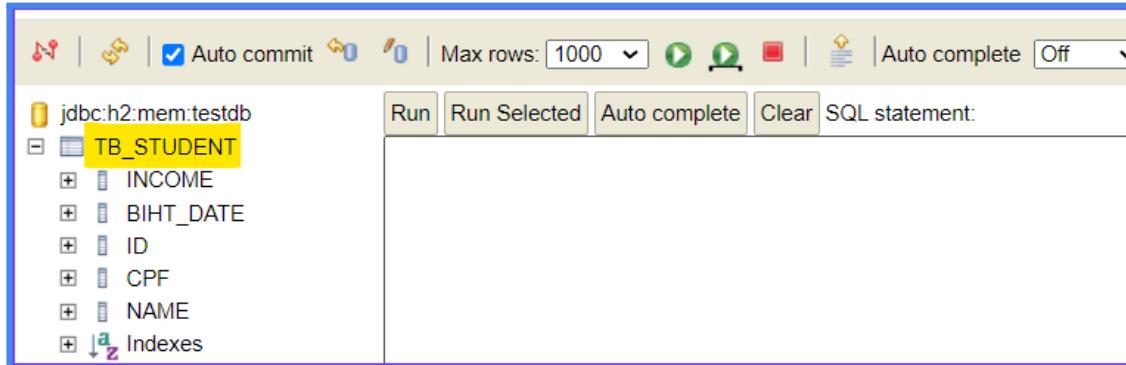
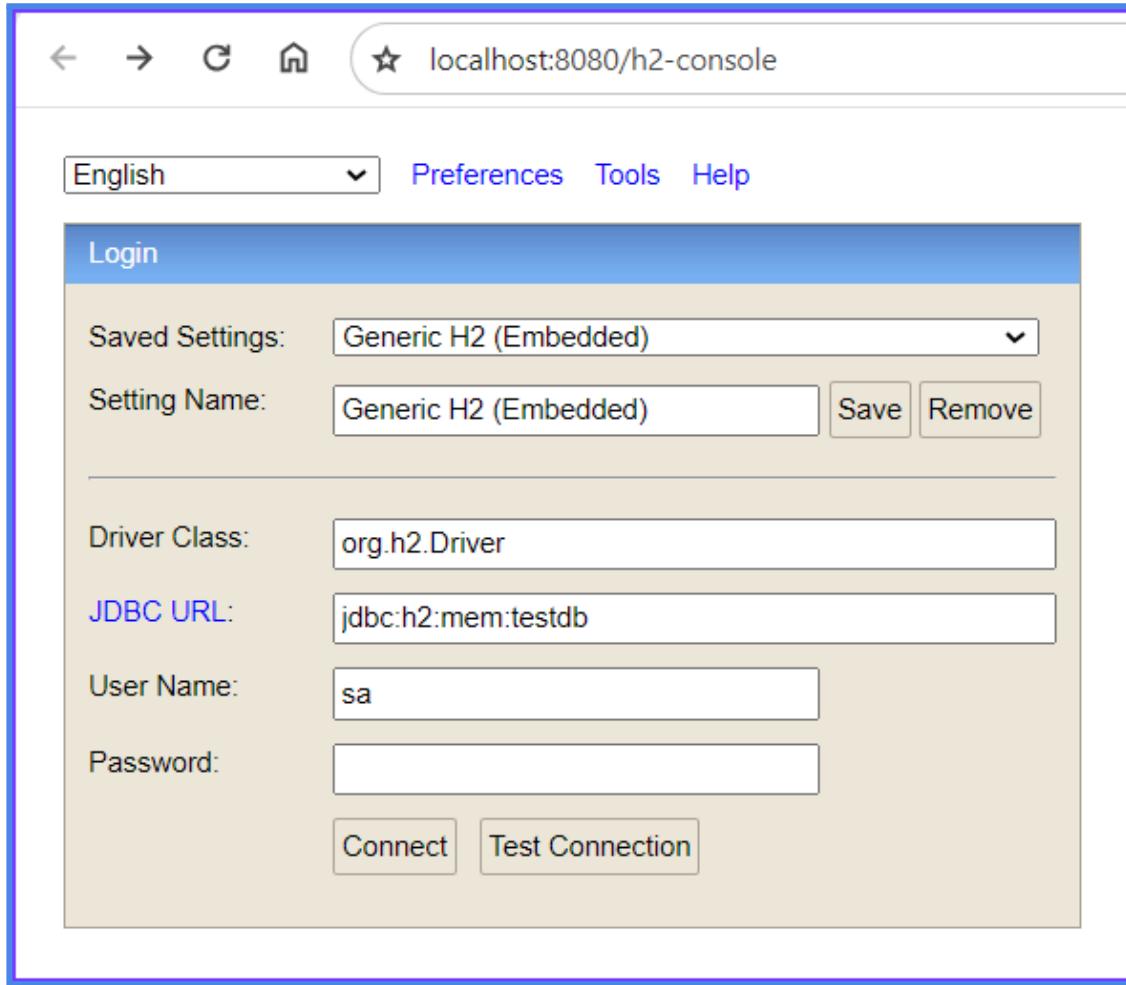


The screenshot shows the Eclipse IDE interface. On the left, the 'Boot Dashboard' view displays a local project named 'projcrud' running on port 8080. On the right, the 'Console' tab is active, showing the application's log output:

```
2024-06-27T15:16:26.926-03:00 INFO 28728 --- [projcr
2024-06-27T15:16:27.063-03:00 INFO 28728 --- [projcr
2024-06-27T15:16:27.451-03:00 INFO 28728 --- [projcr
2024-06-27T15:16:27.467-03:00 INFO 28728 --- [projcr
2024-06-27T15:16:27.756-03:00 INFO 28728 --- [projcr
2024-06-27T15:16:27.761-03:00 INFO 28728 --- [projcr
```

Acessar o banco H2, via web

- <http://localhost:8080/h2-console>



Teste de inserção

The screenshot shows a SQL interface with two main sections. The top section contains a toolbar with buttons for Run, Run Selected, Auto complete, Clear, and a text input field for SQL statements. Below the toolbar is a code editor with the following SQL statement:

```
INSERT INTO TB_STUDENT (NAME) VALUES ('Auricelio')
```

The bottom section displays the results of the execution:

```
INSERT INTO TB_STUDENT (NAME) VALUES ('Auricelio');
Update count: 1
(3 ms)
```

The second screenshot shows a similar interface with a toolbar and a code editor containing a SELECT statement:

```
SELECT * FROM TB_STUDENT
```

The results show one row in a table:

INCOME	BITH_DATE	ID	CPF	NAME
null	null	1	null	Auricelio

(1 row, 3 ms)

Testar no postman

The screenshot shows the Postman application interface. At the top, there is a header bar with a GET method, the URL localhost:8080/students, a Send button, and a dropdown menu.

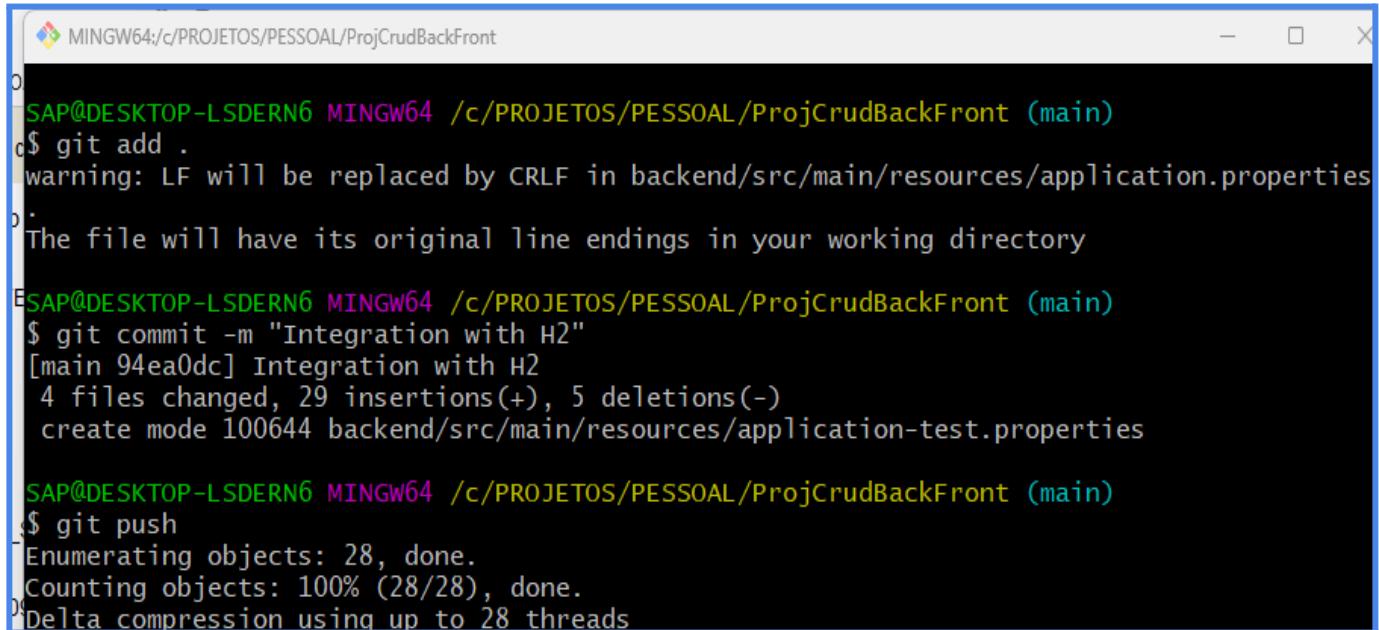
Below the header, there are tabs for Params, Auth, Headers (7), Body, Pre-req., Tests, and Settings. The Headers tab is currently selected, showing 5 entries. To the right of these tabs, there are status indicators: 200 OK, 51 ms, 234 B, and a Save as example button.

The Body tab is selected, displaying a JSON response. The response is a single array element:

```
1 [ 
2   {
3     "id": 1,
4     "name": "Auricelio",
5     "cpf": null,
6     "bithDate": null,
7     "income": null
8   }
9 ]
```

Github-6

- “Git bash here” no diretório do projeto
 - git add .
 - git commit -m “Integration with H2”
 - git push



A screenshot of a terminal window titled "MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront (main)". The window contains the following command-line session:

```
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git add .
warning: LF will be replaced by CRLF in backend/src/main/resources/application.properties
.
The file will have its original line endings in your working directory

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Integration with H2"
[main 94ea0dc] Integration with H2
 4 files changed, 29 insertions(+), 5 deletions(-)
  create mode 100644 backend/src/main/resources/application-test.properties

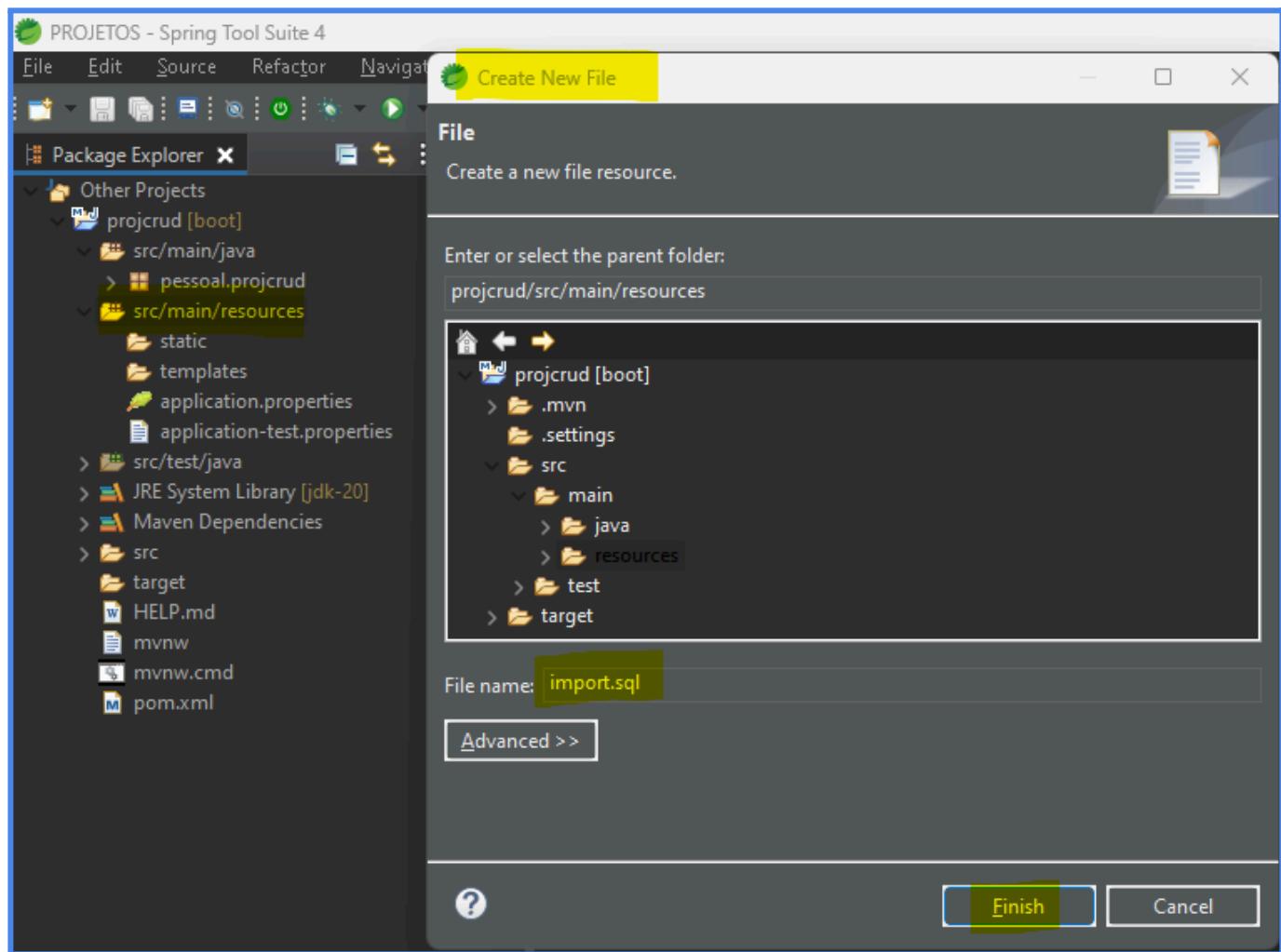
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 28 threads
```

SEEDING DA BASE DE DADOS

- É o termo utilizado para dar uma carga inicial de dados ao banco, a fim de conseguirmos realizar testes com dados. Toda vez que o sistema reiniciar a carga inicial é gerada novamente.

IMPLEMENTAR A CARGA PARA O BANCO

Criar o import.sql, no src/main/resources



NOTA: Nas versões antigas do STS, usava-se o termo *data.sql*. A partir da versão STS 3, passou-se a utilizar a nomenclatura *import.sql*.

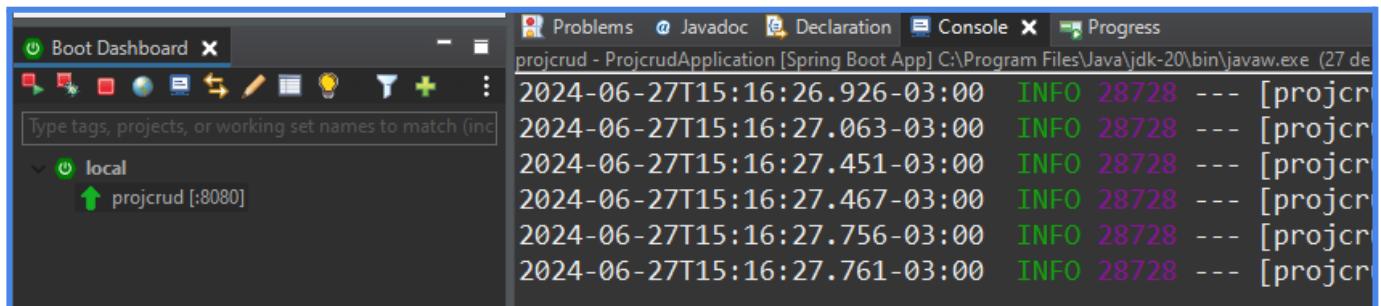
Implementar os inserts para a carga inicial

```
INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Auricelio Freitas', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1982-08-28T10:30:00Z', 10700.5);
INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Miguel Soares', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1989-02-10T10:30:00Z', 20700.5);
INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Matheus Marques', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1992-11-23T10:30:00Z', 10700.5);
INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Rafael Sales', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1988-10-13T10:30:00Z', 10700.5);
INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Bruno Freitas', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1987-04-22T10:30:00Z', 9700.5);
INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Romulo Araujo', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1991-01-09T10:30:00Z', 8700.5);
INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Ryan Sousa', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1994-03-26T10:30:00Z', 7700.5);
INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('test1', '000.111.222-33', null, 1024.0);
```



```
import.sql X
1 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Auricelio Freitas', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1982-08-28T10:30:00Z', 10700.5);
2 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Miguel Soares', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1989-02-10T10:30:00Z', 20700.5);
3 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Matheus Marques', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1992-11-23T10:30:00Z', 10700.5);
4 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Rafael Sales', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1988-10-13T10:30:00Z', 10700.5);
5 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Bruno Freitas', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1987-04-22T10:30:00Z', 9700.5);
6 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Romulo Araujo', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1991-01-09T10:30:00Z', 8700.5);
7 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Ryan Sousa', '000.111.222-33', TIMESTAMP WITH TIME ZONE '1994-03-26T10:30:00Z', 7700.5);
8 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('test1', '000.111.222-33', null, 1024.0);
```

Rodar o projeto

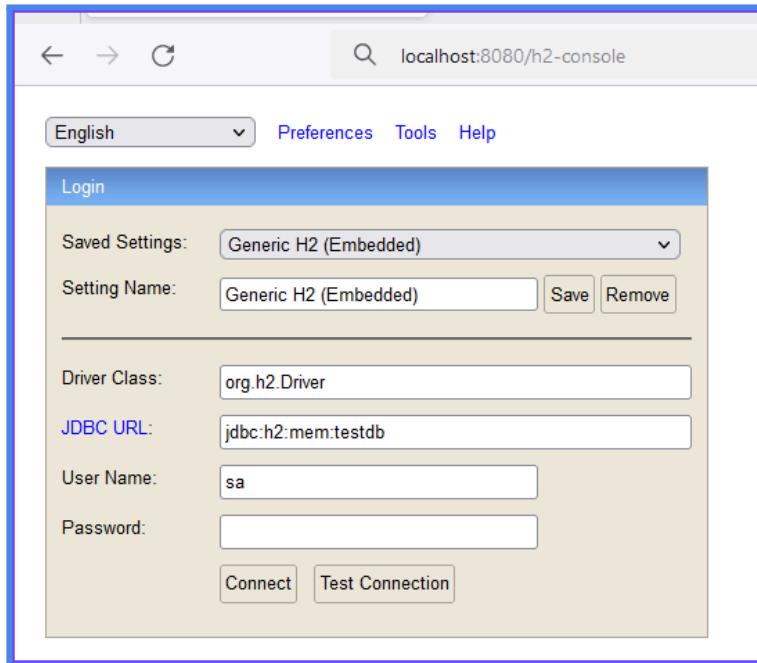


The screenshot shows the Eclipse IDE interface with the following details:

- Spring Boot Dashboard:** Shows a local project named "projcrud" running on port 8080.
- Console Tab:** Displays the application logs for the process "javaw.exe" (pid 28728). The logs show several INFO messages at 2024-06-27T15:16:26.926-03:00, 2024-06-27T15:16:27.063-03:00, 2024-06-27T15:16:27.451-03:00, 2024-06-27T15:16:27.467-03:00, 2024-06-27T15:16:27.756-03:00, and 2024-06-27T15:16:27.761-03:00.

TESTAR A CARGA

Testar no banco H2



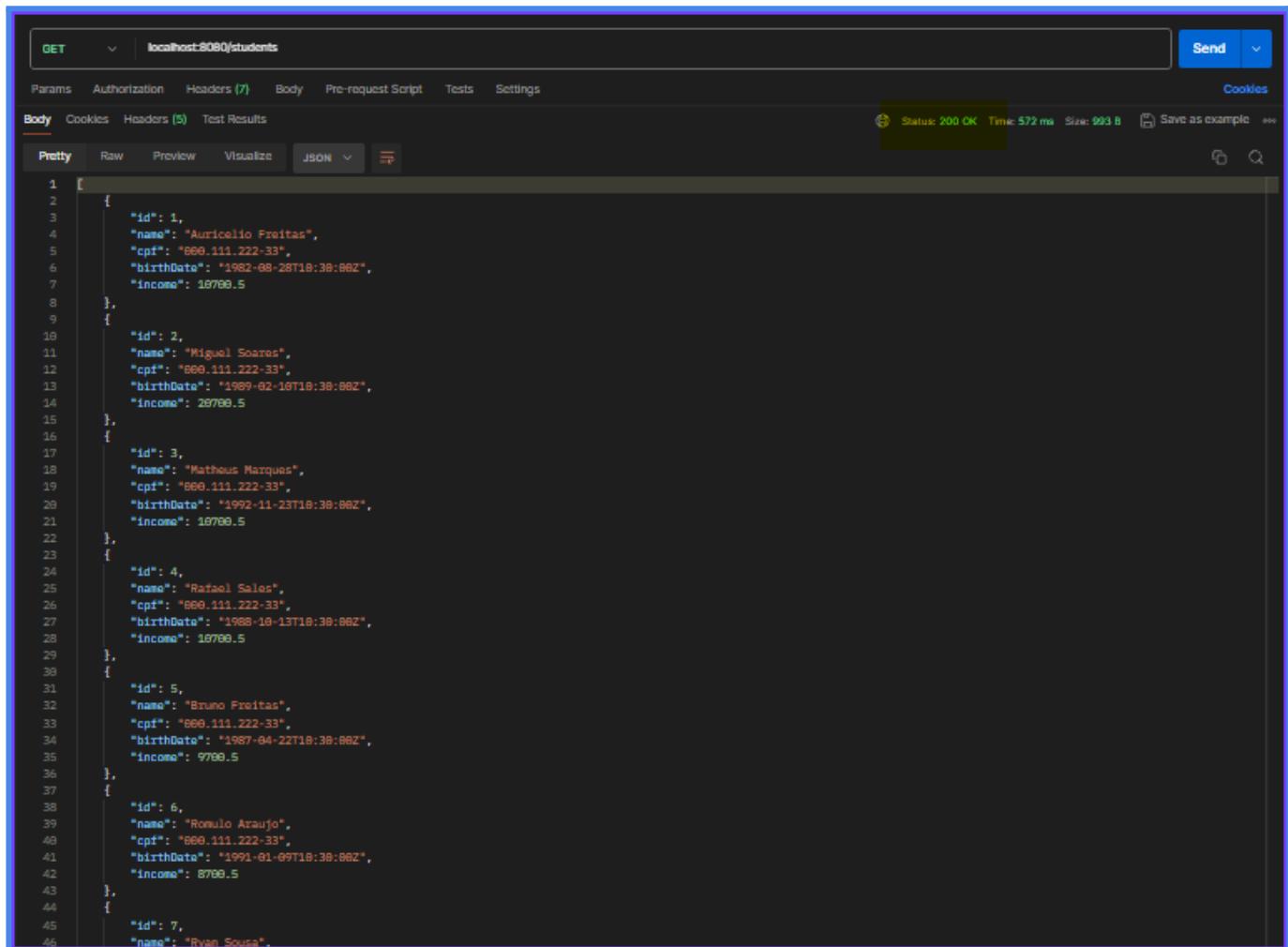
The screenshot shows the H2 Console interface after connecting. The left sidebar lists databases (jdbc:h2:mem:testdb), tables (TB_STUDENT), and schemas (INFORMATION_SCHEMA, Users). The right side shows the SQL statement 'SELECT * FROM TB_STUDENT' highlighted in yellow, and the resulting table output:

```
SELECT * FROM TB_STUDENT;
```

BIRTH_DATE	INCOME	ID	CPF	NAME
1982-08-28 10:30:00+00	10700.5	1	000.111.222-33	Auricelio Freitas
1989-02-10 10:30:00+00	20700.5	2	000.111.222-33	Miguel Soares
1992-11-23 10:30:00+00	10700.5	3	000.111.222-33	Matheus Marques
1988-10-13 10:30:00+00	10700.5	4	000.111.222-33	Rafael Sales
1987-04-22 10:30:00+00	9700.5	5	000.111.222-33	Bruno Freitas
1991-01-09 10:30:00+00	8700.5	6	000.111.222-33	Romulo Araujo
1994-03-26 10:30:00+00	7700.5	7	000.111.222-33	Ryan Sousa
null	1024.0	8	000.111.222-33	test1

(8 rows, 2 ms)

Testar no Postman



The screenshot shows a Postman interface with the following details:

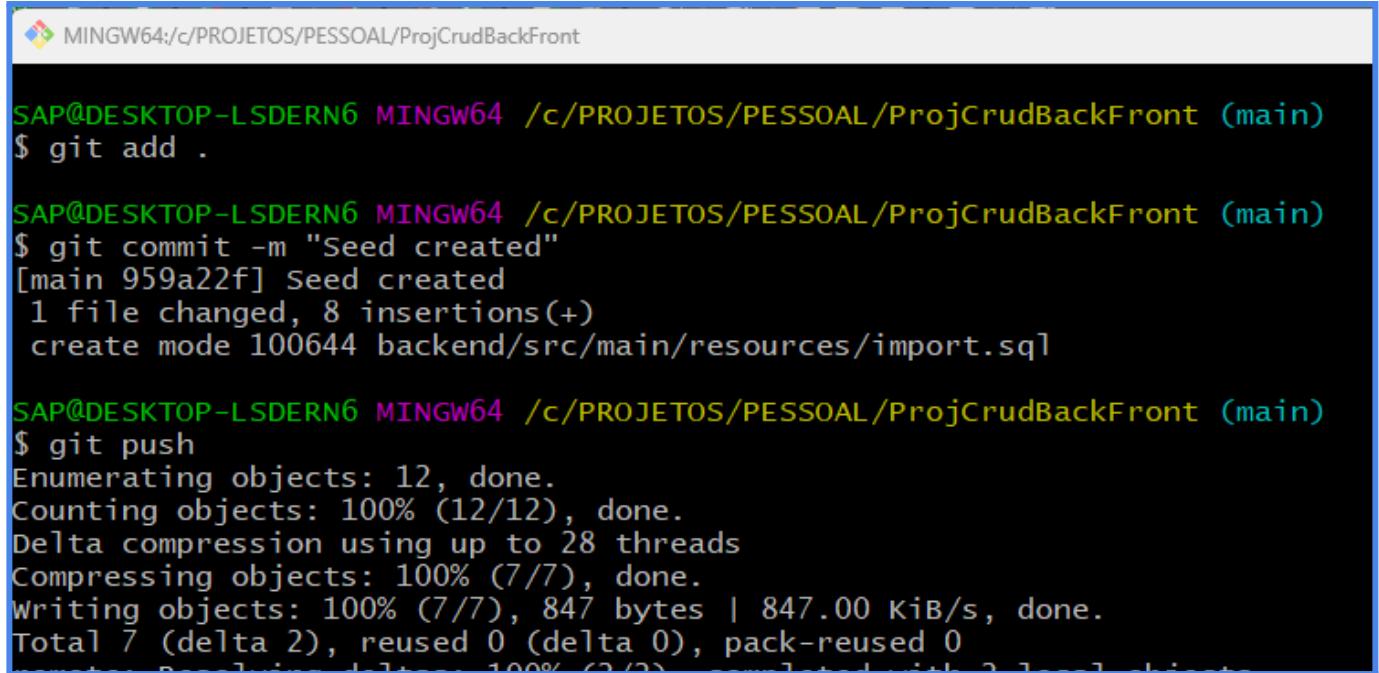
- Method: GET
- URL: localhost:8080/students
- Headers: Authorization, Headers (7), Body, Pre-request Script, Tests, Settings
- Body tab selected
- JSON response (Pretty):

```
1 [
2   {
3     "id": 1,
4     "name": "Auricelio Freitas",
5     "cpf": "000.111.222-33",
6     "birthDate": "1982-08-28T10:30:00Z",
7     "income": 10700.5
8   },
9   {
10    "id": 2,
11    "name": "Miguel Soares",
12    "cpf": "000.111.222-33",
13    "birthDate": "1989-02-10T10:30:00Z",
14    "income": 28700.5
15  },
16  {
17    "id": 3,
18    "name": "Matheus Marques",
19    "cpf": "000.111.222-33",
20    "birthDate": "1992-11-23T10:30:00Z",
21    "income": 10700.5
22  },
23  {
24    "id": 4,
25    "name": "Rafael Sales",
26    "cpf": "000.111.222-33",
27    "birthDate": "1988-10-13T10:30:00Z",
28    "income": 10700.5
29  },
30  {
31    "id": 5,
32    "name": "Bruno Freitas",
33    "cpf": "000.111.222-33",
34    "birthDate": "1987-04-22T10:30:00Z",
35    "income": 9700.5
36  },
37  {
38    "id": 6,
39    "name": "Romulo Araujo",
40    "cpf": "000.111.222-33",
41    "birthDate": "1991-01-09T10:30:00Z",
42    "income": 8700.5
43  },
44  {
45    "id": 7,
46    "name": "Ryan Sousa"
47  }
48]
```

- Status: 200 OK
- Time: 572 ms
- Size: 993 B
- Save as example

Github-7

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Seed created”
 - git push



```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git add .

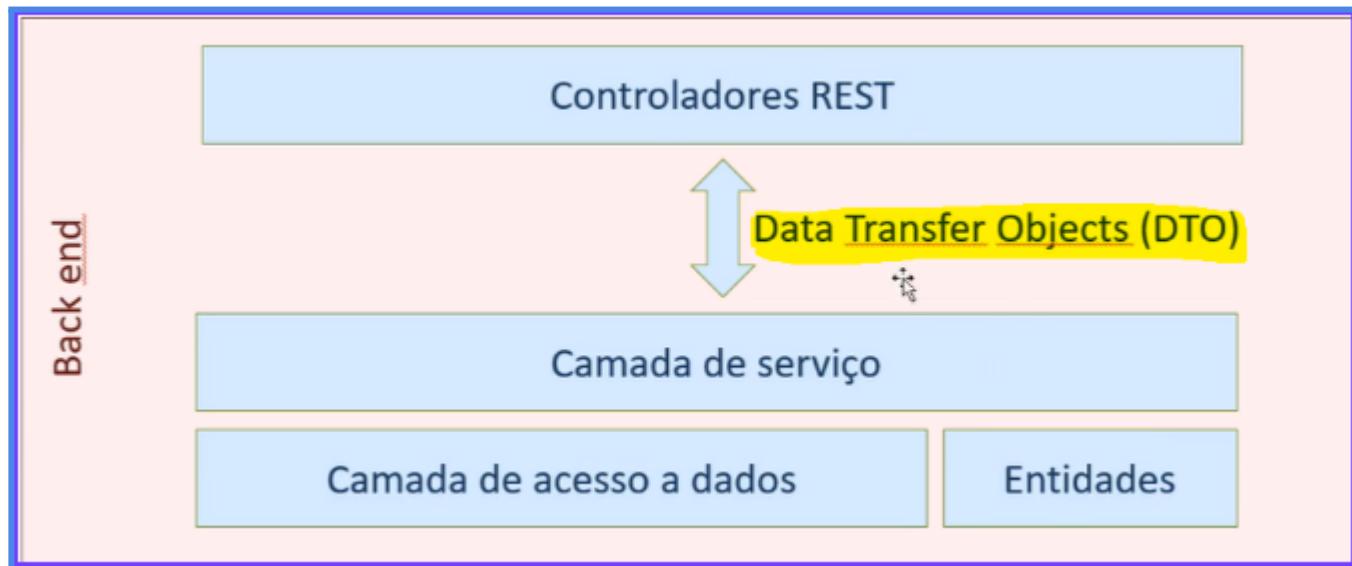
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Seed created"
[main 959a22f] Seed created
 1 file changed, 8 insertions(+)
 create mode 100644 backend/src/main/resources/import.sql

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 28 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 847 bytes | 847.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0
    Delta base blob 100% (2/2)       100% (2/2)       100% (2/2)
```

DTO

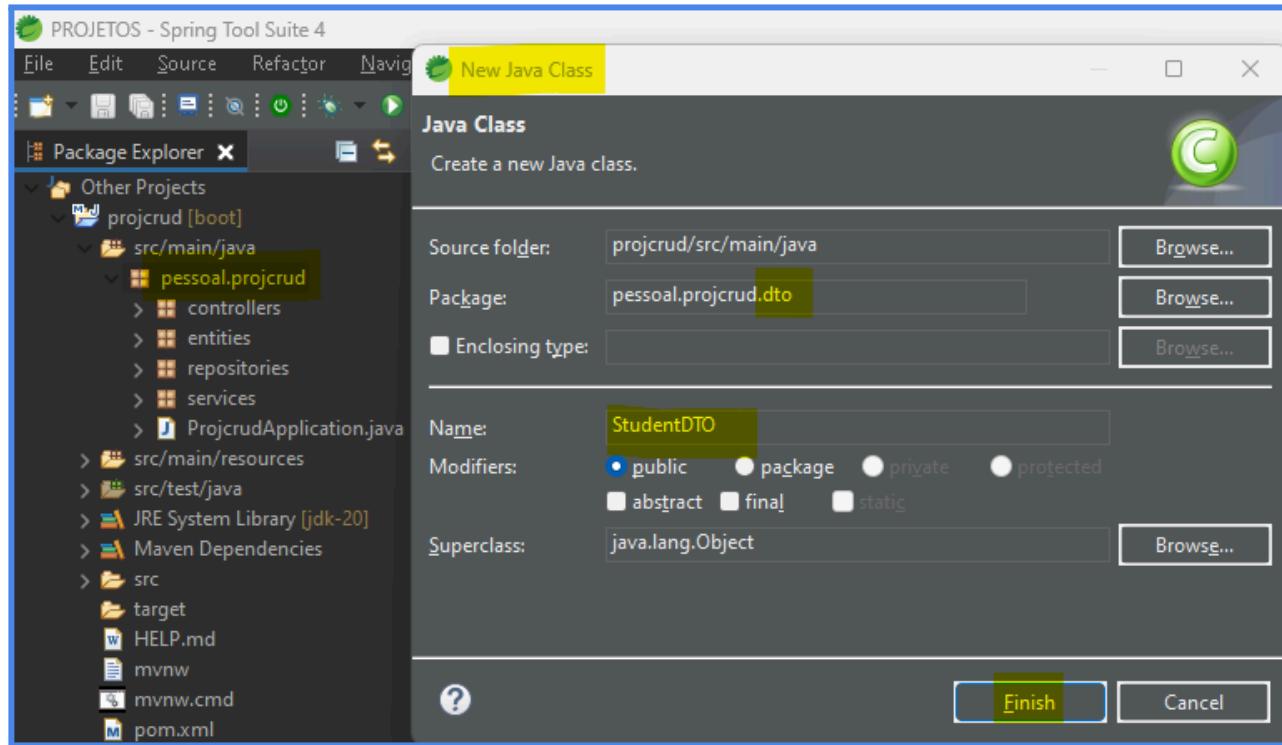
- DTO é um objeto que serve, apenas, para transferência de dados.
- Não tem relação com a JPA.
- O Controlador Rest não tem integração direta com a Entidade
- A comunicação entre o Controlador e o Serviço é feito por meio do DTO
- Podemos controlar quais dados serão entregues para a API

CONCEITUAL



IMPLEMENTAR A ESTRUTURA E O STUDENT_DTO

Criar a estrutura e a classe DTO

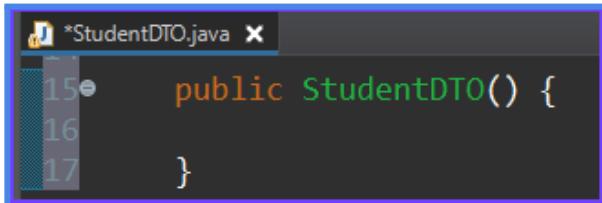


Implementar o Serializable e os mesmos atributos da Classe Student

```
*StudentDTO.java
6 public class StudentDTO implements Serializable {
7     private static final long serialVersionUID = 1L;
8
9     private Long id;
10    private String name;
11    private String cpf;
12    private Instant birthDate;
13    private Double income;
```

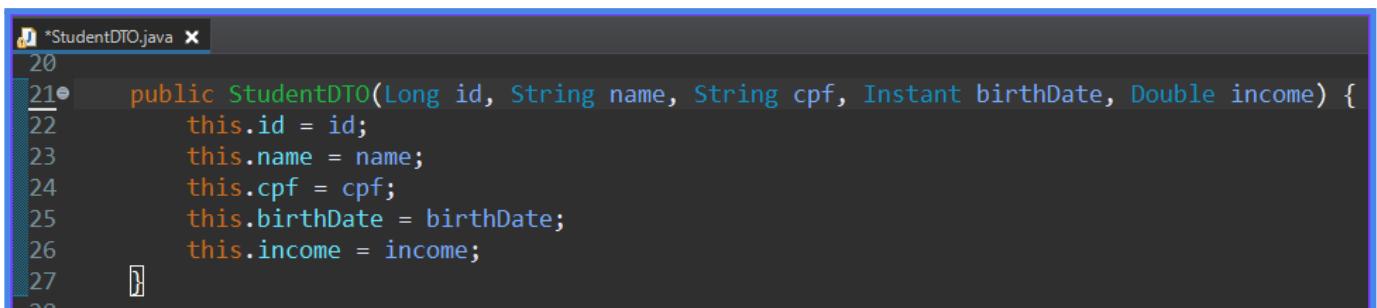
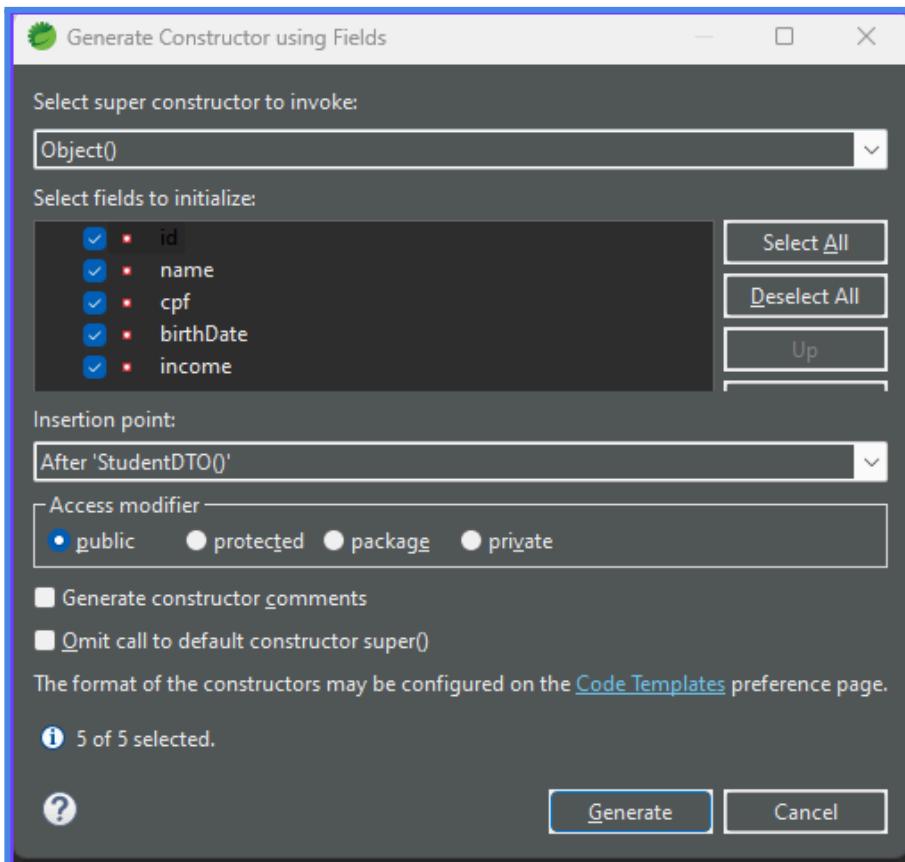
Implementar os construtores

Vazio



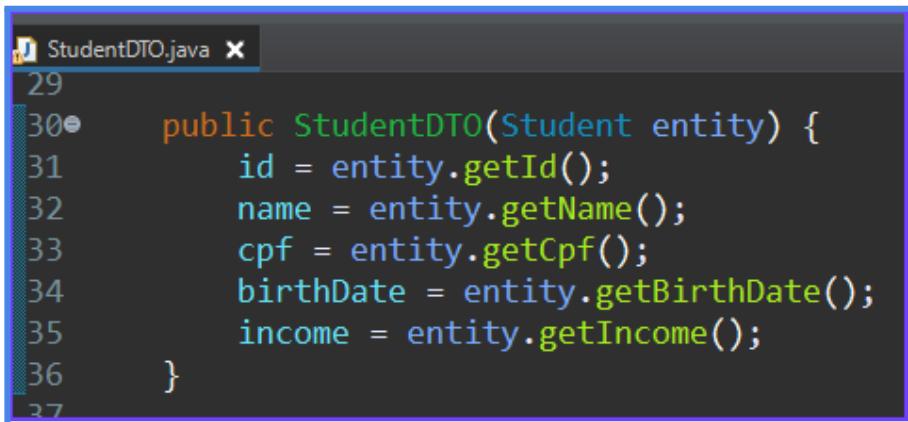
```
15*     public StudentDTO() {  
16  
17 }
```

de Classe



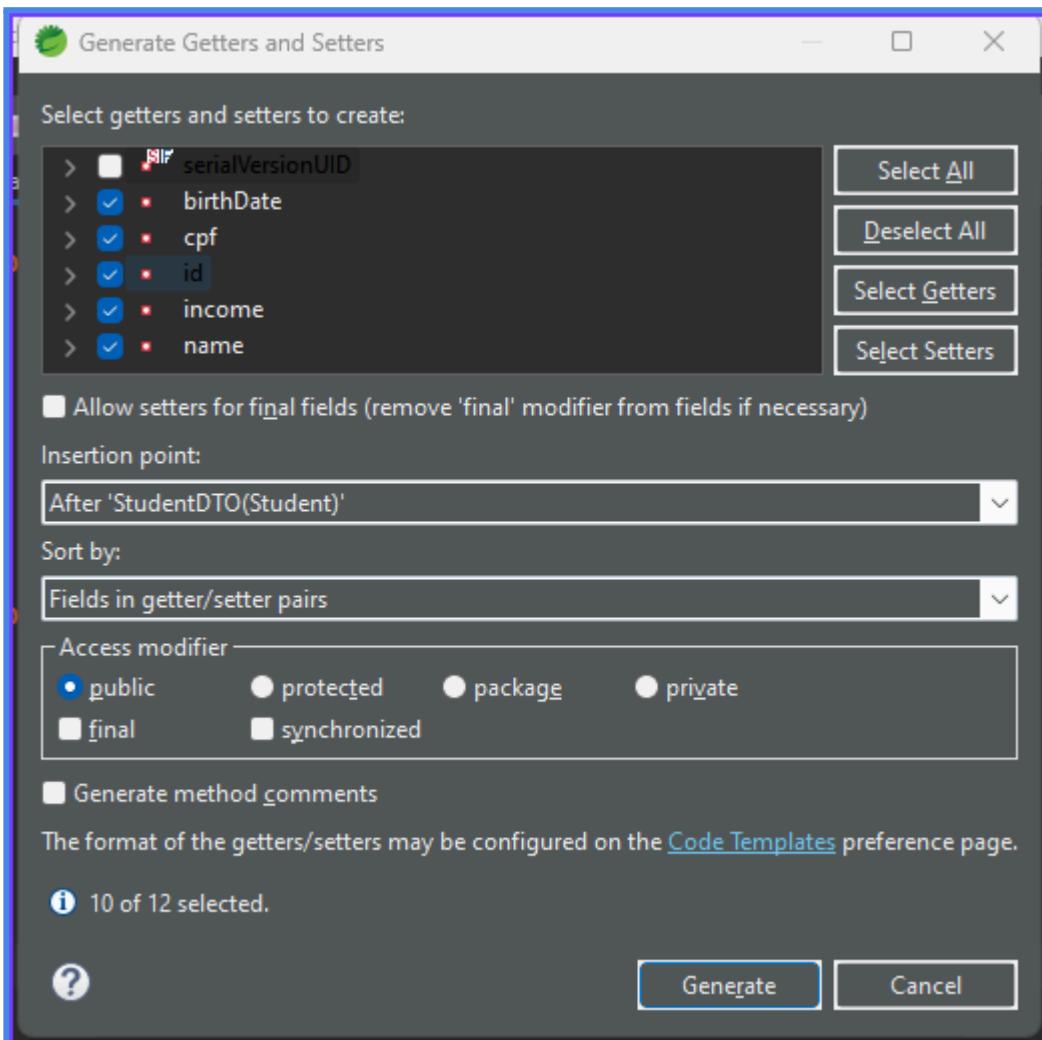
```
20  
21*     public StudentDTO(Long id, String name, String cpf, Instant birthDate, Double income) {  
22         this.id = id;  
23         this.name = name;  
24         this.cpf = cpf;  
25         this.birthDate = birthDate;  
26         this.income = income;  
27     }  
28 }
```

De Entidade



```
StudentDTO.java
29
30     public StudentDTO(Student entity) {
31         id = entity.getId();
32         name = entity.getName();
33         cpf = entity.getCpf();
34         birthDate = entity.getBirthDate();
35         income = entity.getIncome();
36     }
37
```

Implementar os Getters and Setters



```
*StudentDTO.java x
50
51
52
53
54
55
56
57
58
59
```

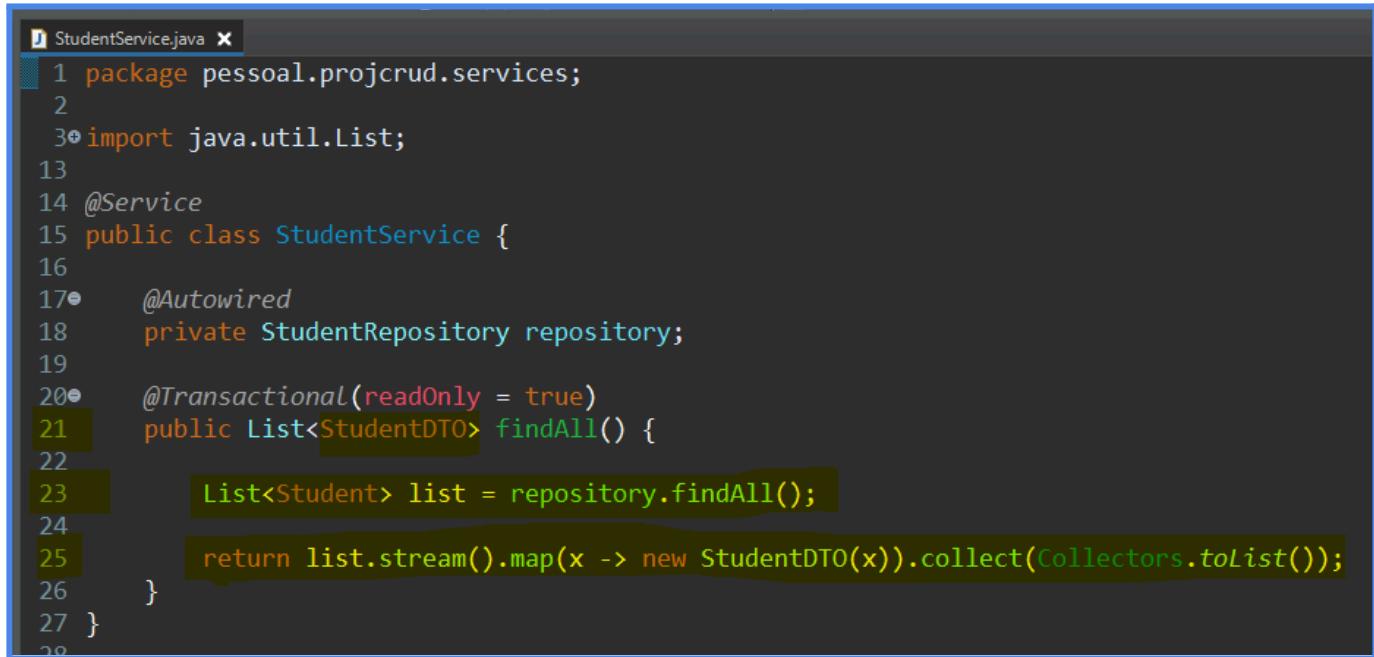
```
37•     public Long getId() {
38         return id;
39     }
40
41•     public void setId(Long id) {
42         this.id = id;
43     }
44
45•     public String getName() {
46         return name;
47     }
48
49•     public void setName(String name) {
50         this.name = name;
51     }
52
53•     public String getCpf() {
54         return cpf;
55     }
56
57•     public void setCpf(String cpf) {
58         this.cpf = cpf;
59     }
```

```
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
```

```
60
61•     public LocalDate getBirthDate() {
62         return birthDate;
63     }
64
65•     public void setBirthDate(LocalDate birthDate) {
66         this.birthDate = birthDate;
67     }
68
69•     public Double getIncome() {
70         return income;
71     }
72
73•     public void setIncome(Double income) {
74         this.income = income;
75     }
76 }
77 }
```

REALIZAR OS AJUSTES PARA O DTO

Implementar o DTO na classe StudentService



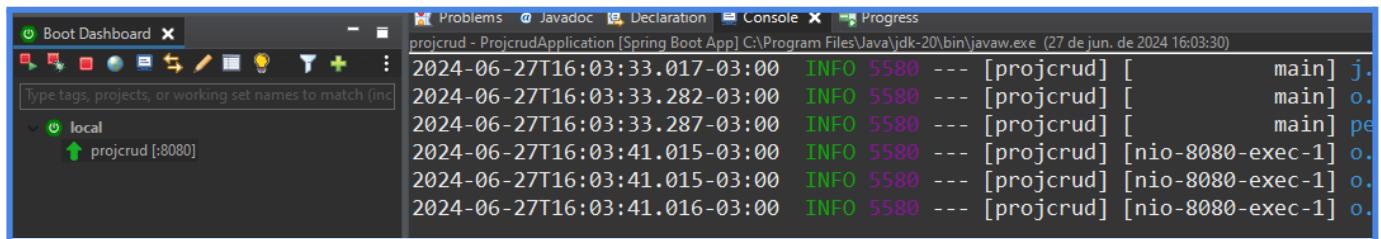
```
StudentService.java
1 package pessoal.projcrud.services;
2
3 import java.util.List;
4
5 @Service
6 public class StudentService {
7
8     @Autowired
9     private StudentRepository repository;
10
11     @Transactional(readOnly = true)
12     public List<StudentDTO> findAll() {
13
14         List<Student> list = repository.findAll();
15
16         return list.stream().map(x -> new StudentDTO(x)).collect(Collectors.toList());
17     }
18 }
```

Implementar o DTO na classe StudentController



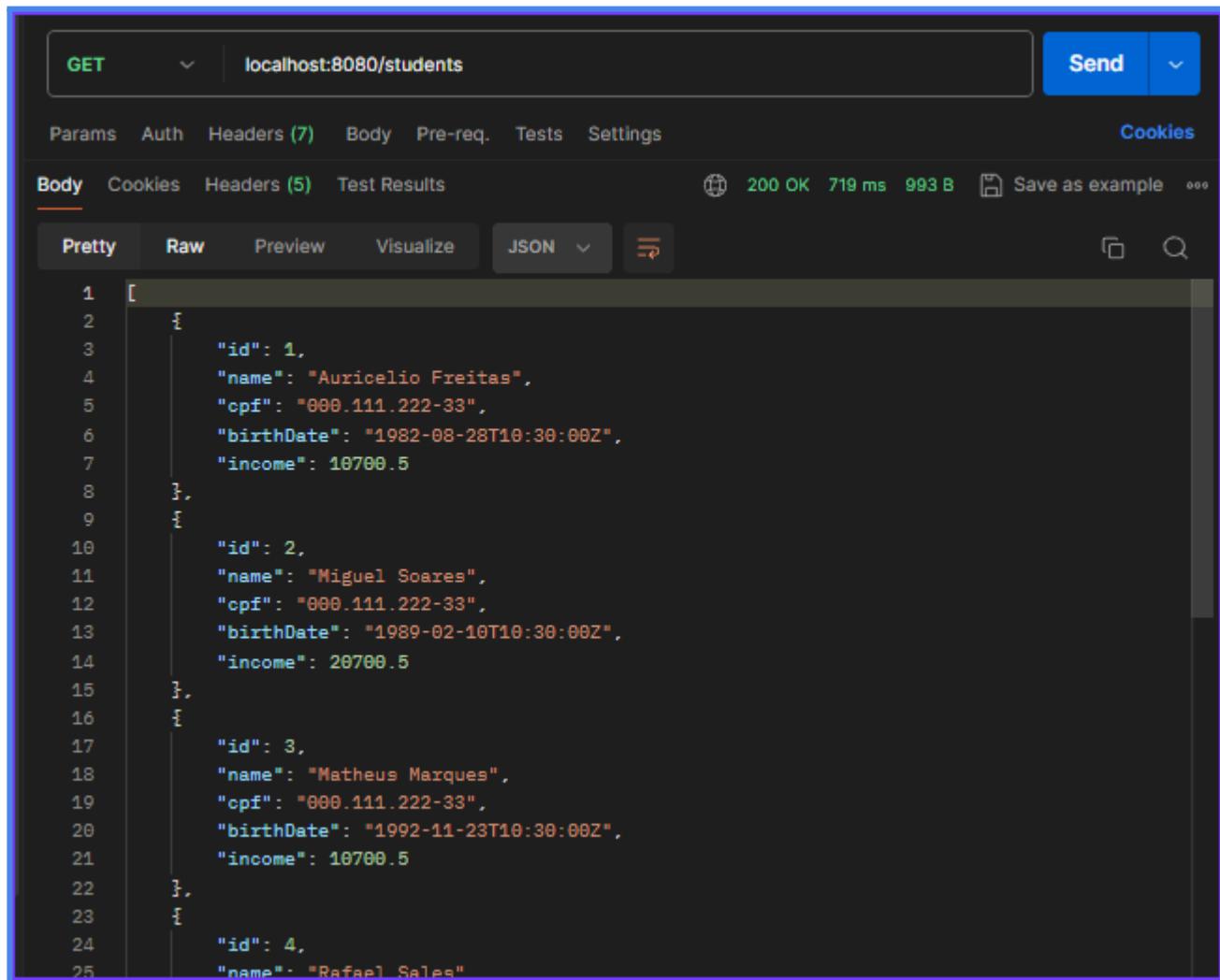
```
1 package pessoal.projcrud.controllers;
2
3 import java.util.List;
4
5
6 @RestController
7 @RequestMapping(value = "/students")
8 public class StudentController {
9
10
11     @Autowired
12     private StudentService service;
13
14
15     @GetMapping
16     public ResponseEntity<List<StudentDTO>> findAll() {
17
18         List<StudentDTO> list = service.findAll();
19
20         return ResponseEntity.ok().body(list);
21     }
22
23 }
24
25
26 }
```

Rodar o projeto



```
2024-06-27T16:03:33.017-03:00 INFO 5580 --- [projcrud] [main] j.:
2024-06-27T16:03:33.282-03:00 INFO 5580 --- [projcrud] [main] o.:
2024-06-27T16:03:33.287-03:00 INFO 5580 --- [projcrud] [main] pe:
2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.:
2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.:
2024-06-27T16:03:41.016-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.:
```

TESTAR COM O POSTMAN



The screenshot shows the Postman interface with a successful API call. The URL is `localhost:8080/students`. The response status is `200 OK` with `719 ms` latency and `993 B` size. The response body is displayed in Pretty JSON format:

```
1 [  
2   {  
3     "id": 1,  
4     "name": "Auricelio Freitas",  
5     "cpf": "000.111.222-33",  
6     "birthDate": "1982-08-28T10:30:00Z",  
7     "income": 10700.5  
8   },  
9   {  
10    "id": 2,  
11    "name": "Miguel Soares",  
12    "cpf": "000.111.222-33",  
13    "birthDate": "1989-02-10T10:30:00Z",  
14    "income": 20700.5  
15  },  
16  {  
17    "id": 3,  
18    "name": "Matheus Marques",  
19    "cpf": "000.111.222-33",  
20    "birthDate": "1992-11-23T10:30:00Z",  
21    "income": 10700.5  
22  },  
23  {  
24    "id": 4,  
25    "name": "Rafael Sales"
```

Github-8

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Created DTO”
 - git push

```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git add .

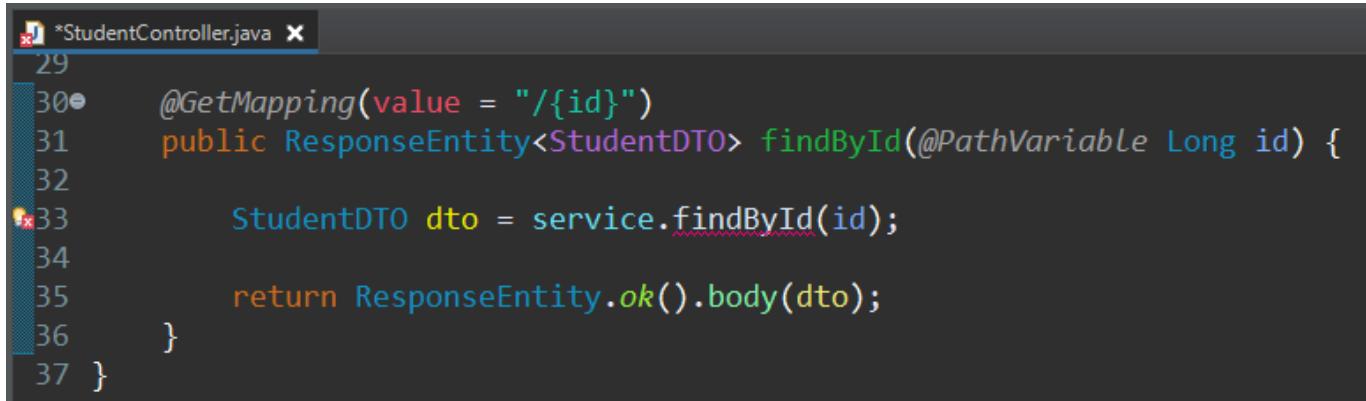
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Created DTO"
[main e195c15] Created DTO
 3 files changed, 97 insertions(+), 5 deletions(-)
 create mode 100644 backend/src/main/java/pessoal/projcrud/dto/StudentDTO.java

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 28 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (14/14), 1.52 KiB | 1.52 MiB/s, done.
Total 14 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
  959a22f..e195c15  main -> main
```

ENDPOINT: FIND_BY_ID

BUSCAR ALUNOS POR ID COM GET

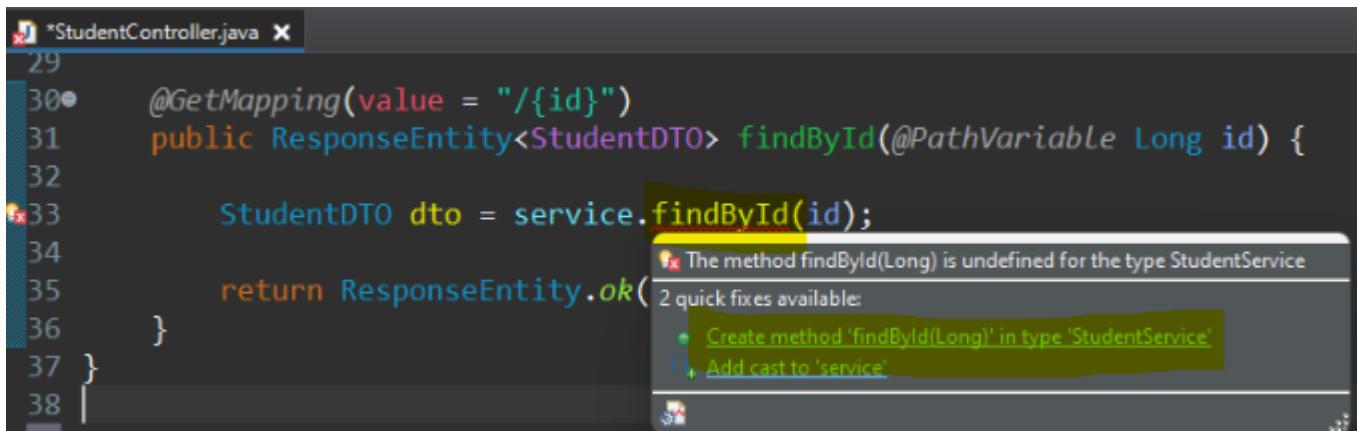
Implementar busca por Id, no StudentController



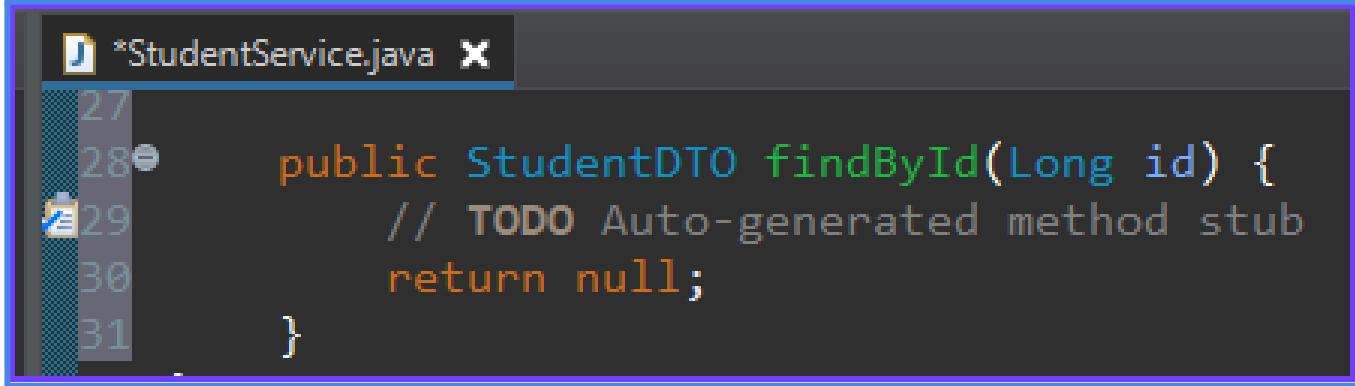
```
*StudentController.java
29
30     @GetMapping(value = "/{id}")
31     public ResponseEntity<StudentDTO> findById(@PathVariable Long id) {
32
33         StudentDTO dto = service.findById(id);
34
35         return ResponseEntity.ok().body(dto);
36     }
37 }
```

Implementar o método **findById**, no StudentService

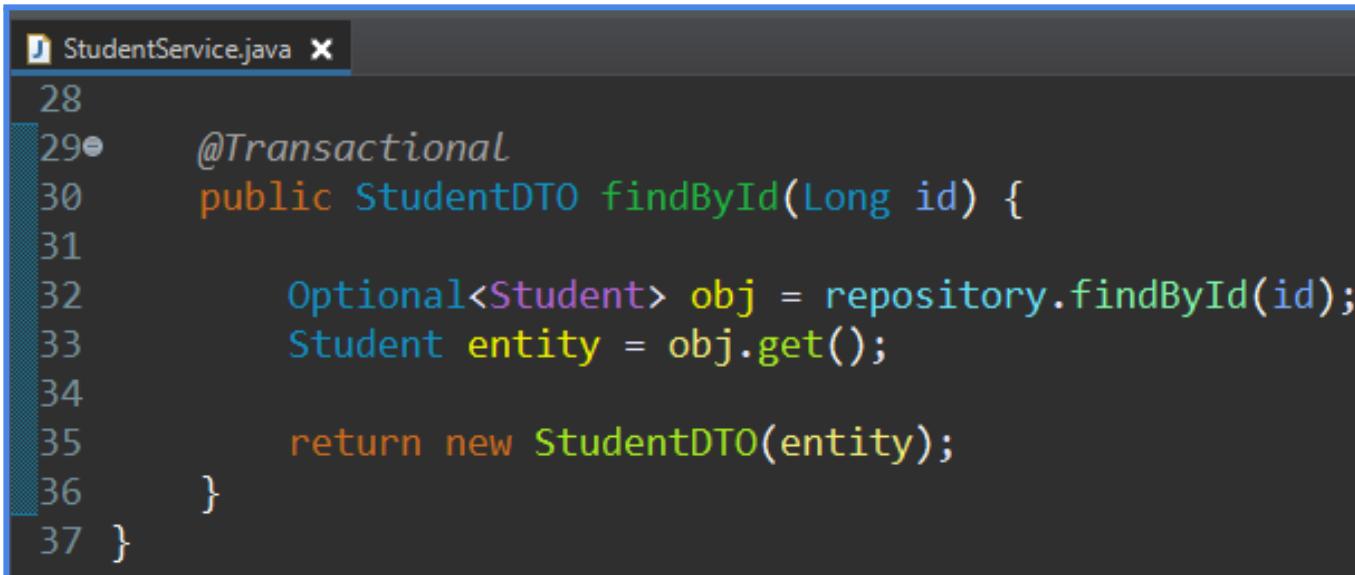
DICA: No StudentController, clicando sobre o método e em sobre o *Create method ...*, será criado um método default no StudentService, onde podemos implementá-lo em seguida.



```
*StudentController.java
29
30     @GetMapping(value = "/{id}")
31     public ResponseEntity<StudentDTO> findById(@PathVariable Long id) {
32
33         StudentDTO dto = service.findById(id);
34
35         return ResponseEntity.ok();
36     }
37 }
38 }
```

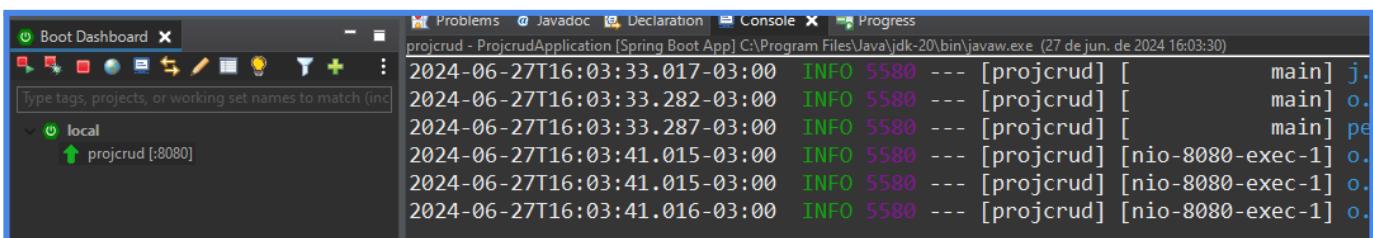


```
*StudentService.java
27
28     public StudentDTO findById(Long id) {
29         // TODO Auto-generated method stub
30         return null;
31     }
```



```
StudentService.java
28
29     @Transactional
30     public StudentDTO findById(Long id) {
31
32         Optional<Student> obj = repository.findById(id);
33         Student entity = obj.get();
34
35         return new StudentDTO(entity);
36     }
37 }
```

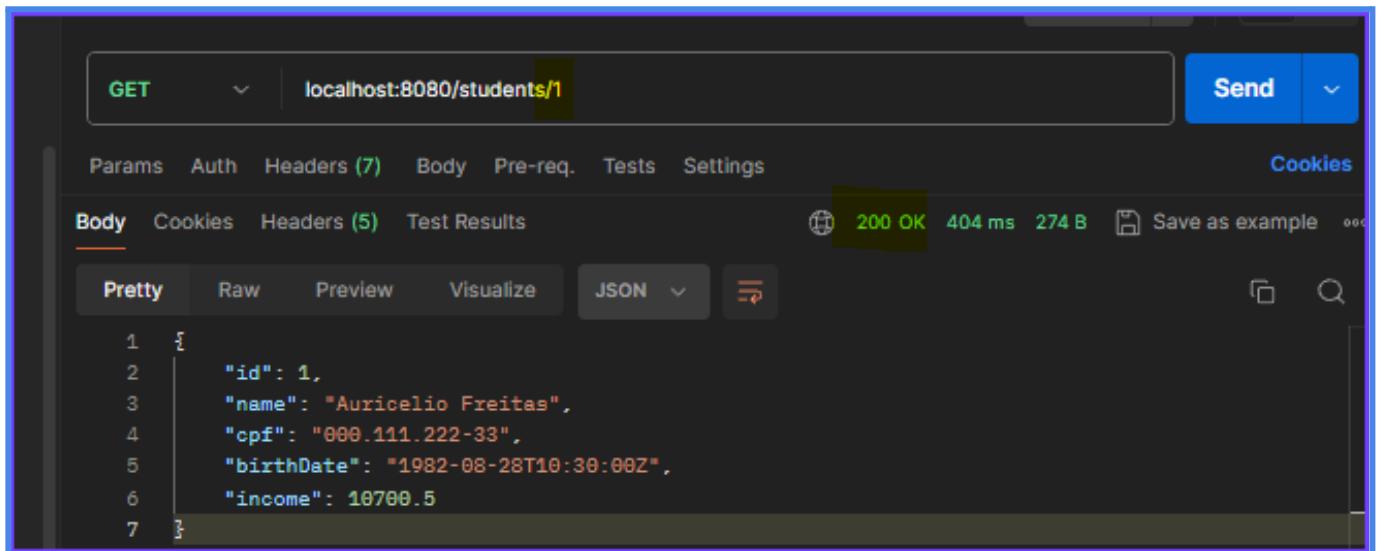
Rodar o projeto



The screenshot shows the Eclipse IDE interface. On the left, the 'Boot Dashboard' view displays a project named 'projcrud' under the 'local' workspace, with a port number ':8080' listed. On the right, the 'Console' tab is active, showing the output of a Spring Boot application. The log entries are as follows:

```
2024-06-27T16:03:33.017-03:00 INFO 5580 --- [projcrud] [main] j.
2024-06-27T16:03:33.282-03:00 INFO 5580 --- [projcrud] [main] o.
2024-06-27T16:03:33.287-03:00 INFO 5580 --- [projcrud] [main] pe
2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.016-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
```

TESTAR NO POSTMAN



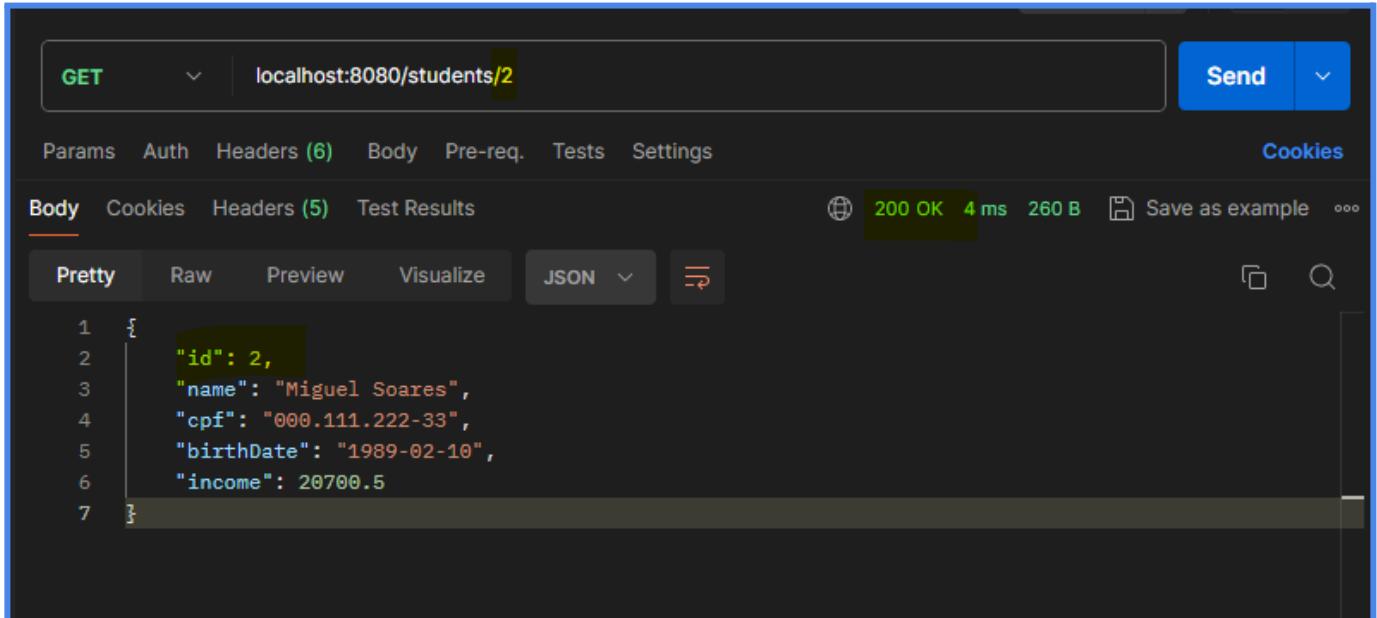
GET localhost:8080/students/1 Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 1,  
3   "name": "Auricelio Freitas",  
4   "cpf": "000.111.222-33",  
5   "birthDate": "1982-08-28T10:30:00Z",  
6   "income": 10700.5  
7 }
```



GET localhost:8080/students/2 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

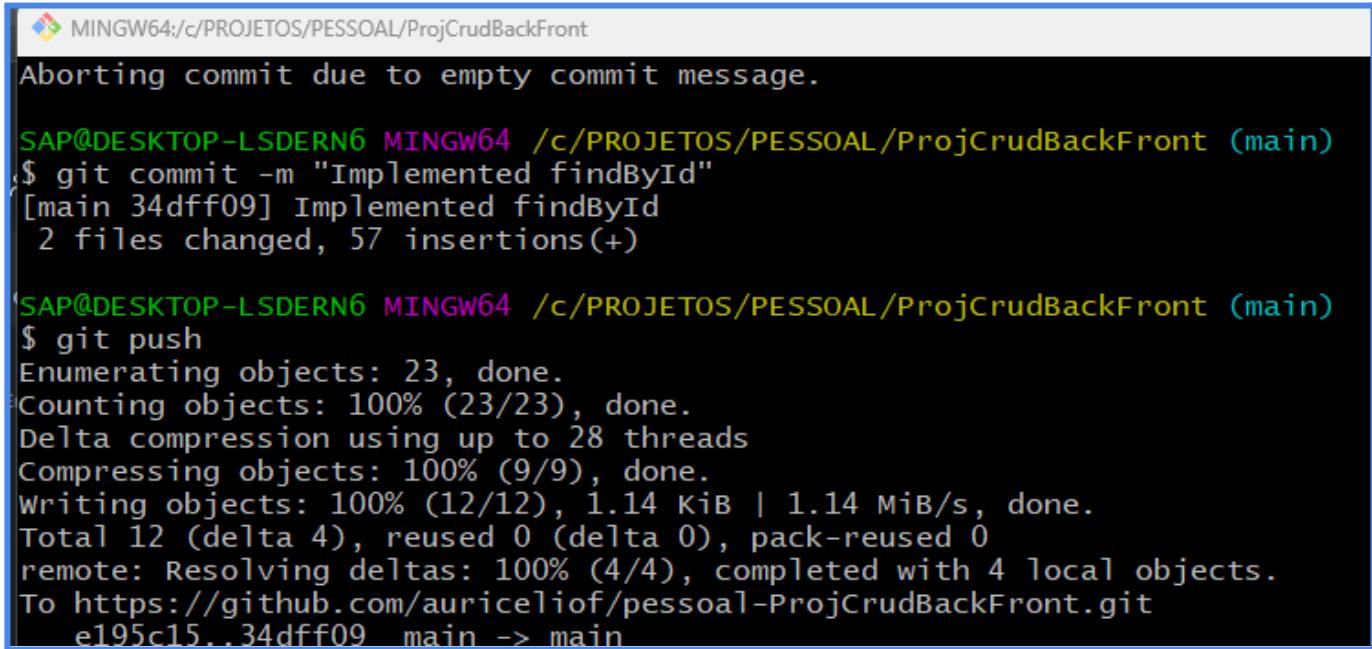
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 2,  
3   "name": "Miguel Soares",  
4   "cpf": "000.111.222-33",  
5   "birthDate": "1989-02-10",  
6   "income": 20700.5  
7 }
```

Github-9

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Implemented findById”
 - git push



MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront

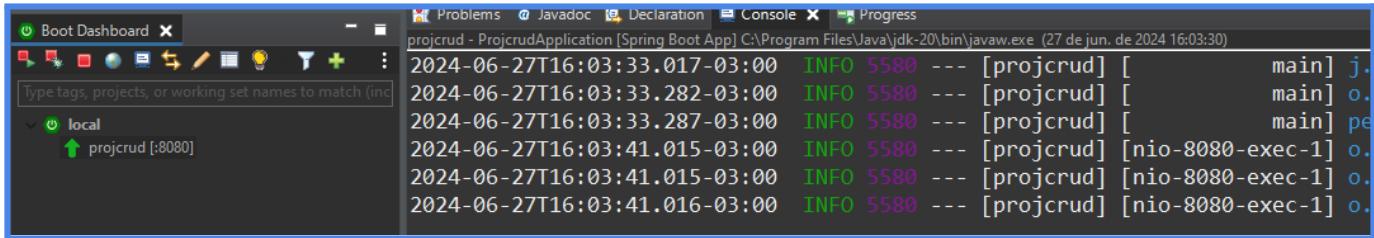
```
Aborting commit due to empty commit message.

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Implemented findById"
[main 34dff09] Implemented findById
 2 files changed, 57 insertions(+)

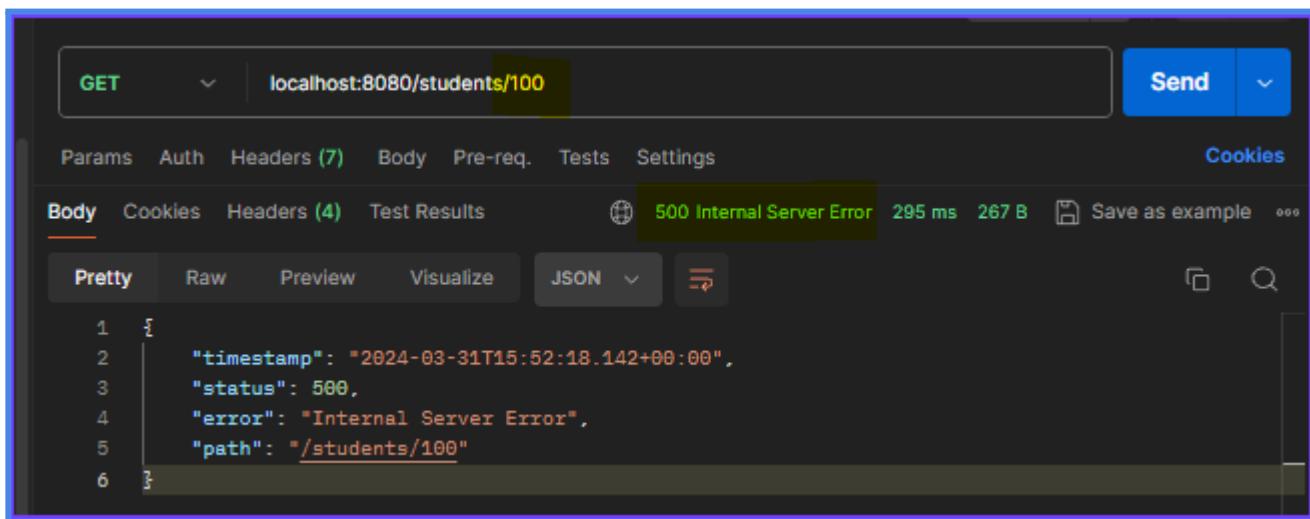
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 28 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.14 KiB | 1.14 MiB/s, done.
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
  e195c15..34dff09 main -> main
```

TRATAMENTO DE EXCEÇÕES PARA O FIND_BY_ID

Rodar o projeto

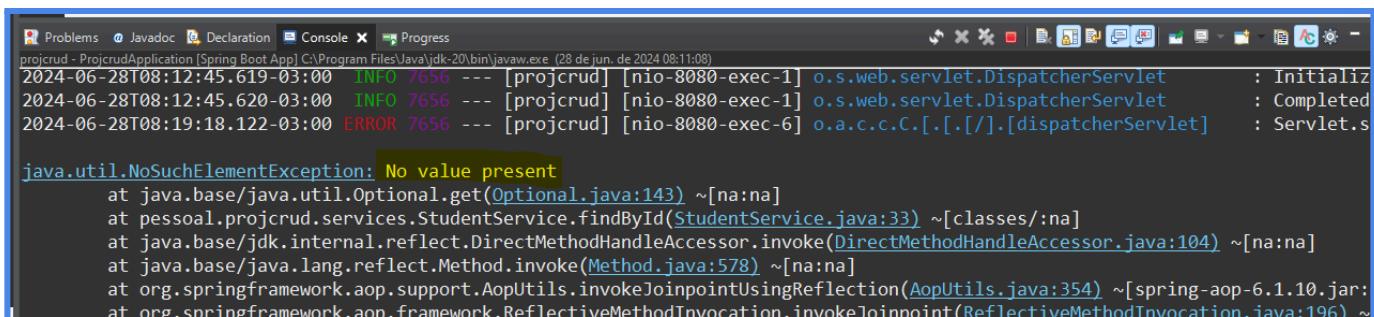


Simular erro no Postman

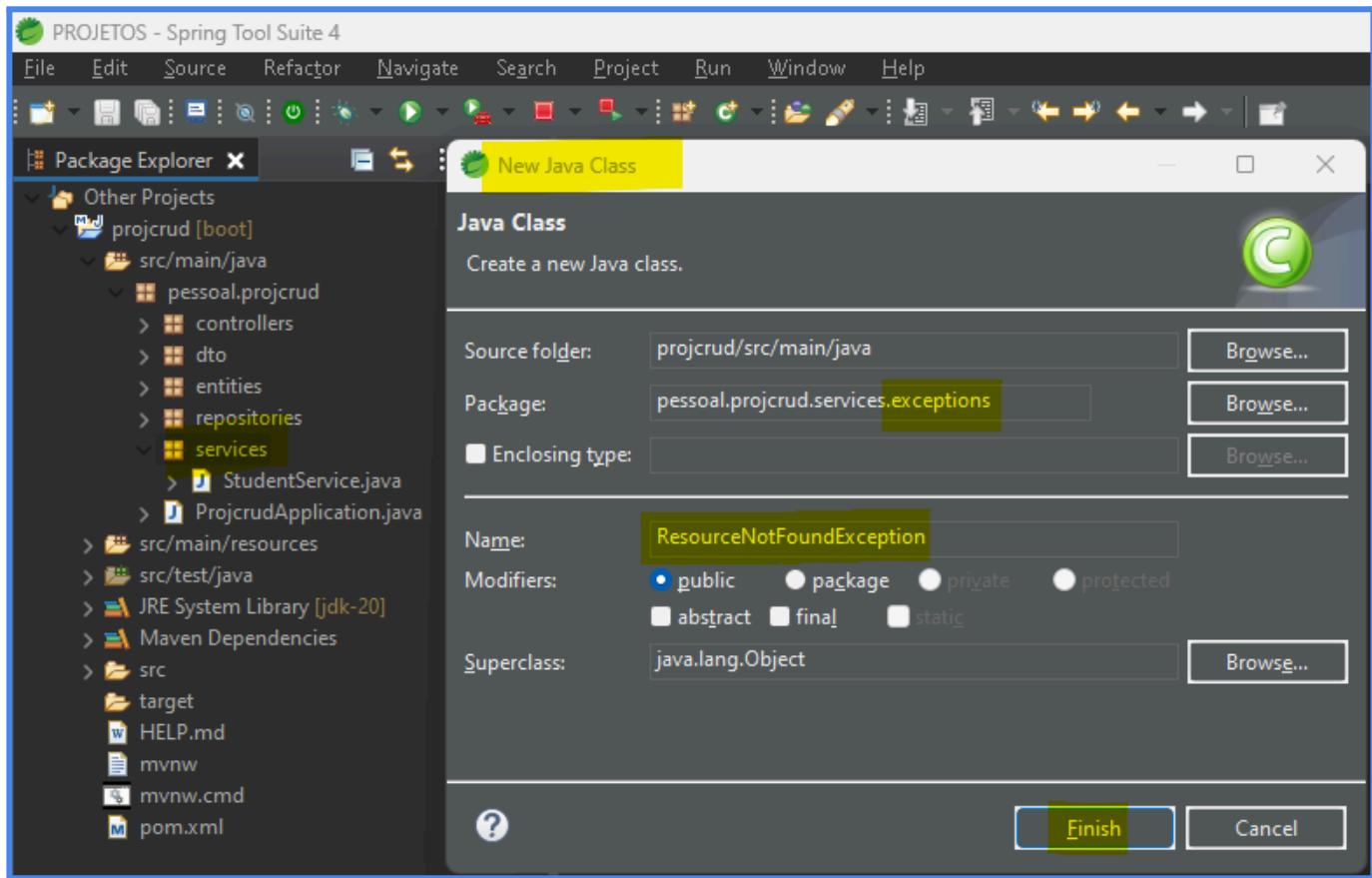


NOTA: Quando realizamos uma busca por um ID que não existe, é retornado o erro 500. Iremos tratá-lo a seguir.

Verificar o erro no console



Criar a estrutura de exceções no service

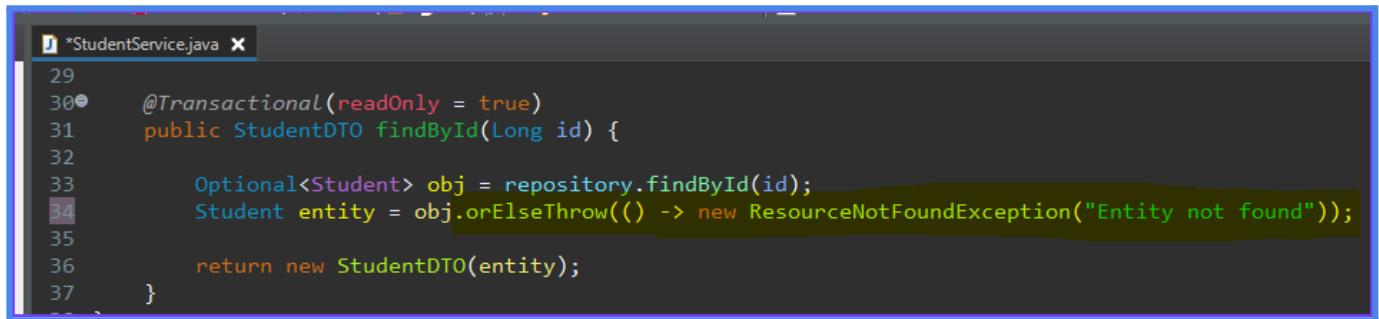


Implementar o ResourceNotFoundException

The screenshot shows the code editor with the file 'ResourceNotFoundException.java' open. The code implements the `ResourceNotFoundException` class, which extends `RuntimeException`. The code is as follows:

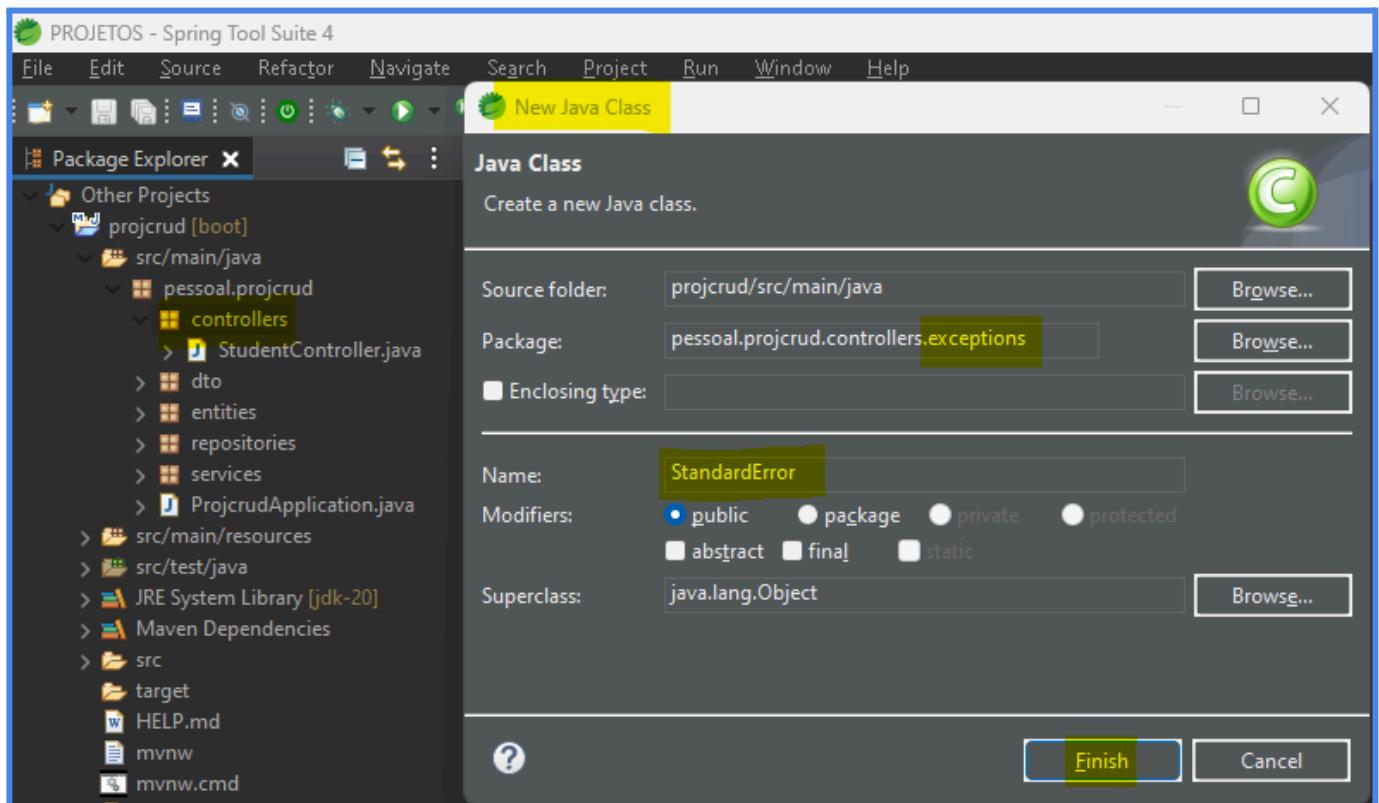
```
1 package pessoal.projcrud.services.exceptions;
2
3 public class ResourceNotFoundException extends RuntimeException{
4     private static final long serialVersionUID = 1L;
5
6     public ResourceNotFoundException(String msg) {
7         super(msg);
8     }
9 }
```

Implementar a exceção no findById, do StudentService



```
29
30     @Transactional(readOnly = true)
31     public StudentDTO findById(Long id) {
32
33         Optional<Student> obj = repository.findById(id);
34         Student entity = obj.orElseThrow(() -> new ResourceNotFoundException("Entity not found"));
35
36         return new StudentDTO(entity);
37     }
```

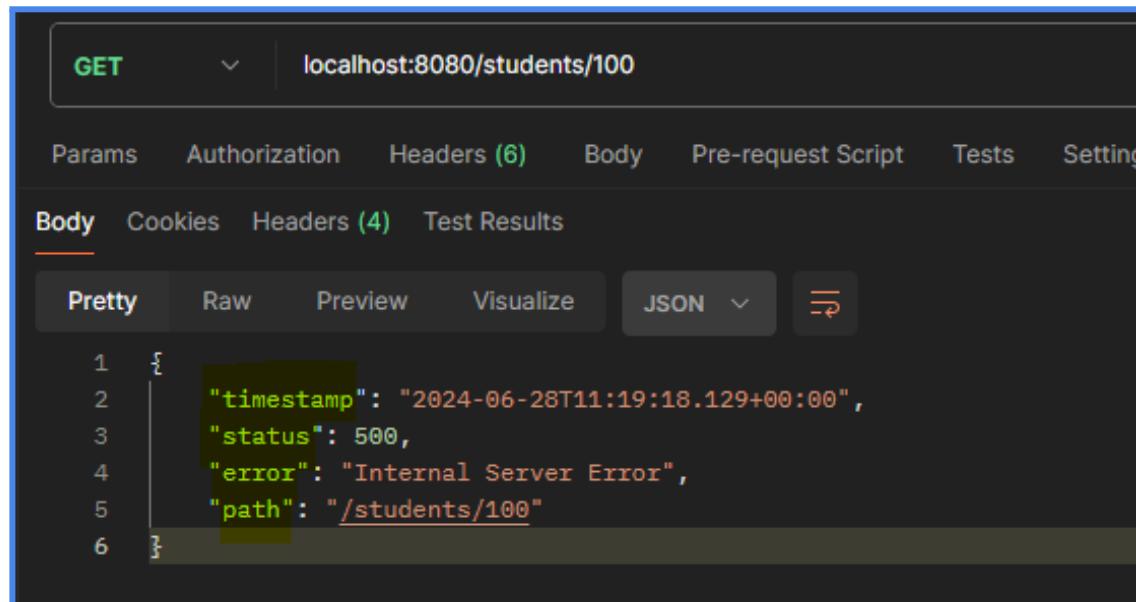
Criar a estrutura de exceções e uma classe personalizada no controller



Implementar o Serializable e os atributos do StandardError conforme erro visto anteriormente

```
StandardError.java ✘
1 package pessoal.projcrud.controllers.exceptions;
2
3 import java.io.Serializable;
4 import java.time.Instant;
5
6 public class StandardError implements Serializable {
7     private static final long serialVersionUID = 1L;
8
9     private Instant timestamp;
10    private Integer status;
11    private String error;
12    private String message;
13    private String path;
14 }
```

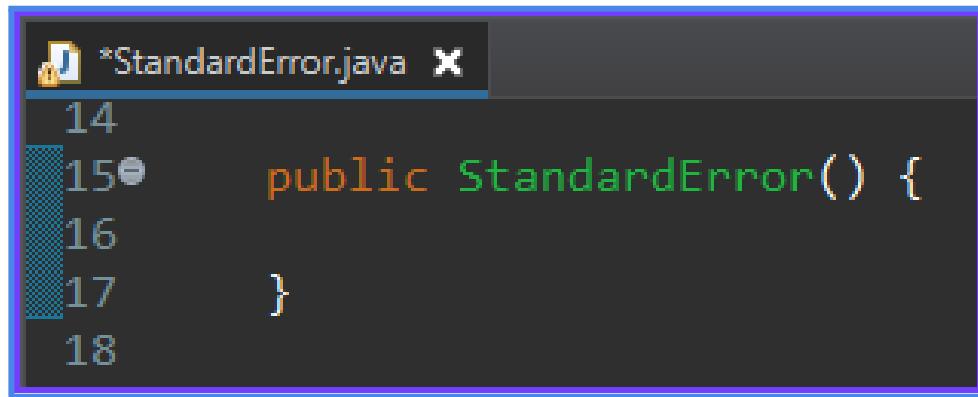
NOTA: Devemos definir os mesmos atributos mostrados no teste de erro do postman.



The screenshot shows a Postman interface with a GET request to `localhost:8080/students/100`. The response body is displayed in JSON format, showing the following structure:

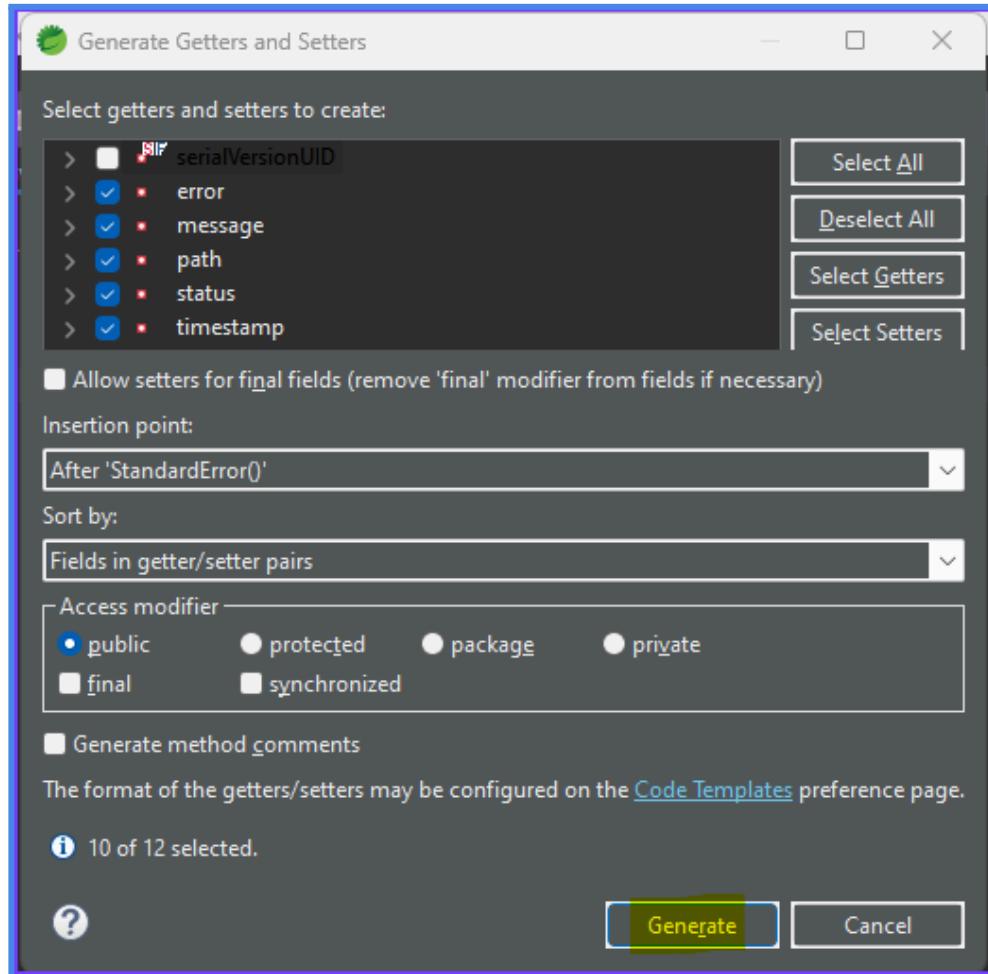
```
1 {
2     "timestamp": "2024-06-28T11:19:18.129+00:00",
3     "status": 500,
4     "error": "Internal Server Error",
5     "path": "/students/100"
6 }
```

Implementar um construtor vazio



```
14
15     public StandardError() {
16
17 }
18
```

Implementar os Getters and Setters



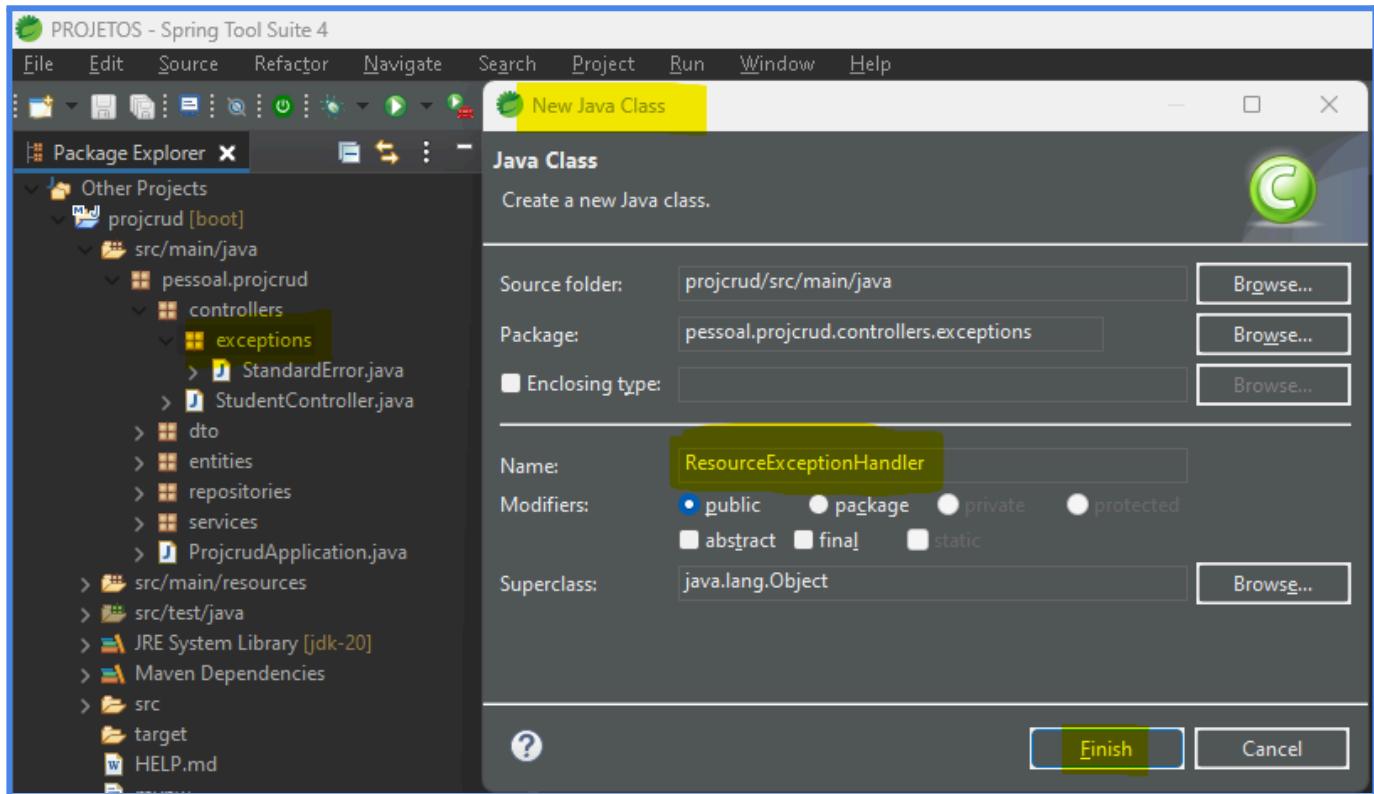
The image shows a Java code editor with two tabs open. The top tab is titled "StandardError.java" and contains the following code:

```
18     public Instant getTimestamp() {  
19         return timestamp;  
20     }  
21  
22     public void setTimestamp(Instant timestamp) {  
23         this.timestamp = timestamp;  
24     }  
25  
26     public Integer getStatus() {  
27         return status;  
28     }  
29  
30     public void setStatus(Integer status) {  
31         this.status = status;  
32     }  
33  
34     public String getError() {  
35         return error;  
36     }  
37  
38     public void setError(String error) {  
39         this.error = error;  
40     }  
41  
42
```

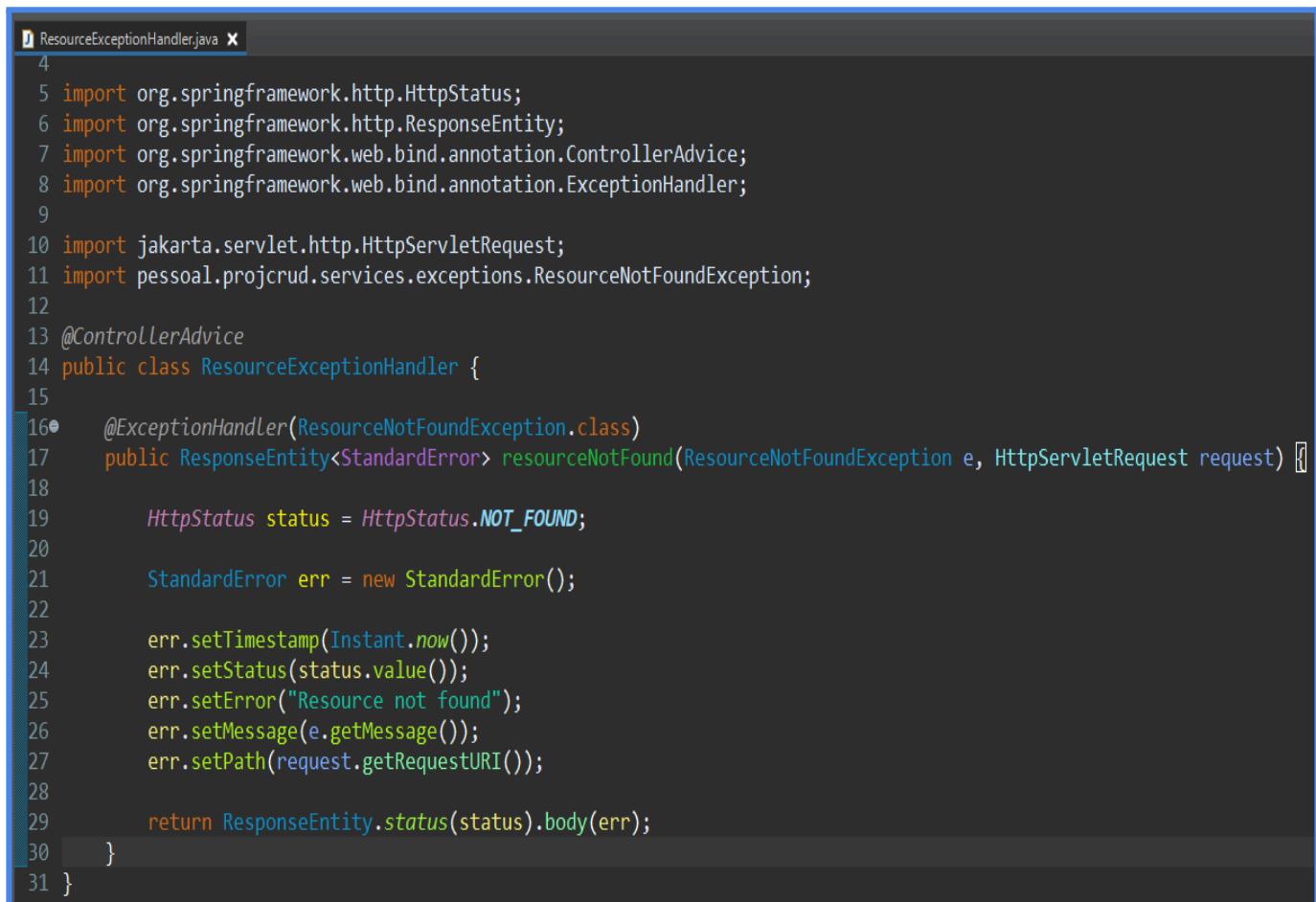
The bottom tab is titled "PathMessage.java" and contains the following code:

```
42  
43     public String getMessage() {  
44         return message;  
45     }  
46  
47     public void setMessage(String message) {  
48         this.message = message;  
49     }  
50  
51     public String getPath() {  
52         return path;  
53     }  
54  
55     public void setPath(String path) {  
56         this.path = path;  
57     }  
58  
59 }
```

Criar um controller advice para manipular a exceção

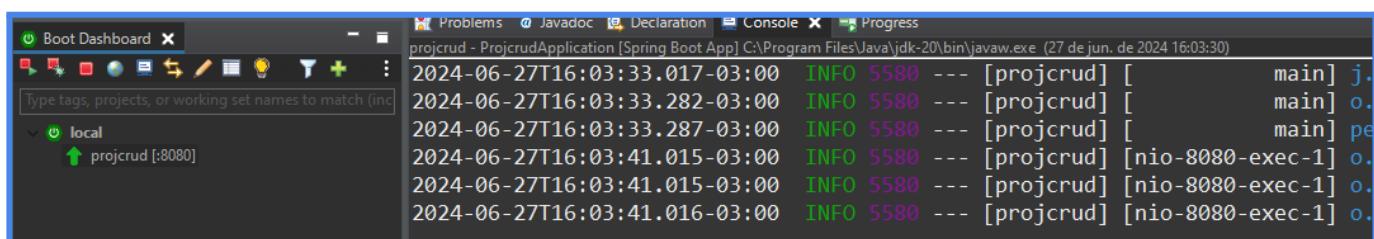


Implementar o ResourceExceptionHandler



```
ResourceExceptionHandler.java
4
5 import org.springframework.http.HttpStatus;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.ControllerAdvice;
8 import org.springframework.web.bind.annotation.ExceptionHandler;
9
10 import jakarta.servlet.http.HttpServletRequest;
11 import pessoal.projcrud.services.exceptions.ResourceNotFoundException;
12
13 @ControllerAdvice
14 public class ResourceExceptionHandler {
15
16     @ExceptionHandler(ResourceNotFoundException.class)
17     public ResponseEntity<StandardError> resourceNotFound(ResourceNotFoundException e, HttpServletRequest request) {
18
19         HttpStatus status = HttpStatus.NOT_FOUND;
20
21         StandardError err = new StandardError();
22
23         err.setTimestamp(Instant.now());
24         err.setStatus(status.value());
25         err.setError("Resource not found");
26         err.setMessage(e.getMessage());
27         err.setPath(request.getRequestURI());
28
29         return ResponseEntity.status(status).body(err);
30     }
31 }
```

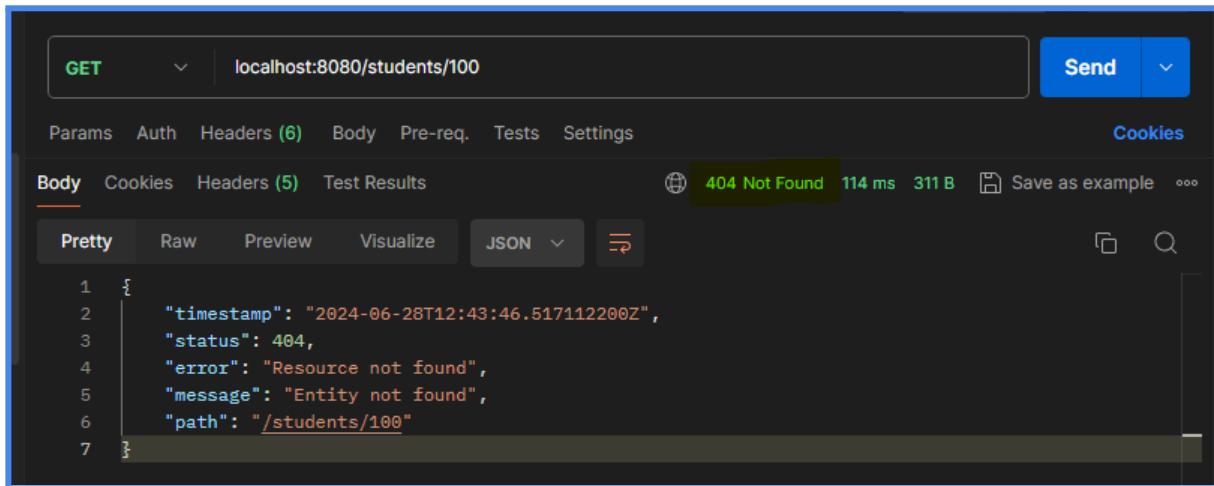
Rodar o projeto



The screenshot shows the Eclipse IDE's Console view with the following log entries:

```
2024-06-27T16:03:33.017-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:33.282-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:33.287-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.016-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
```

TESTAR NO POSTMAN

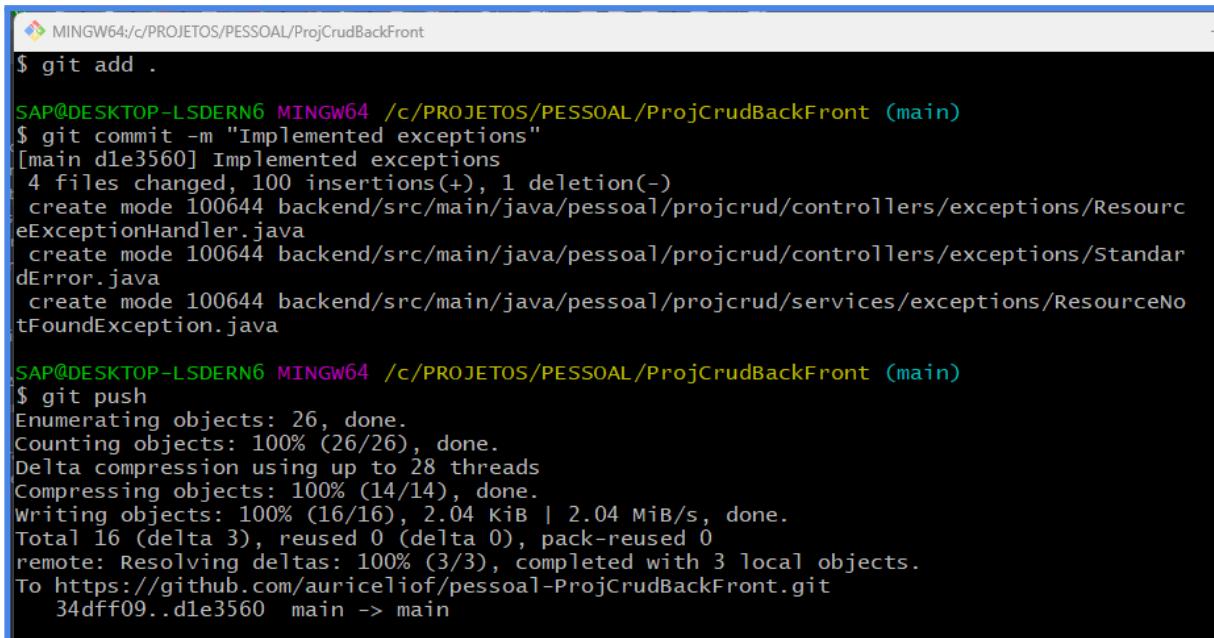


The screenshot shows a Postman request for a GET operation to the URL `localhost:8080/students/100`. The response status is 404 Not Found, with a timestamp of `2024-06-28T12:43:46.517112Z`, a duration of 114 ms, and a body size of 311 B. The response body is a JSON object:

```
1  {
2   "timestamp": "2024-06-28T12:43:46.517112Z",
3   "status": 404,
4   "error": "Resource not found",
5   "message": "Entity not found",
6   "path": "/students/100"
7 }
```

Github-10

- “Git bash here” no diretório do projeto
 - `git add backend`
 - `git commit -m “Implemented exception for findById”`
 - `git push`



```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront
$ git add .

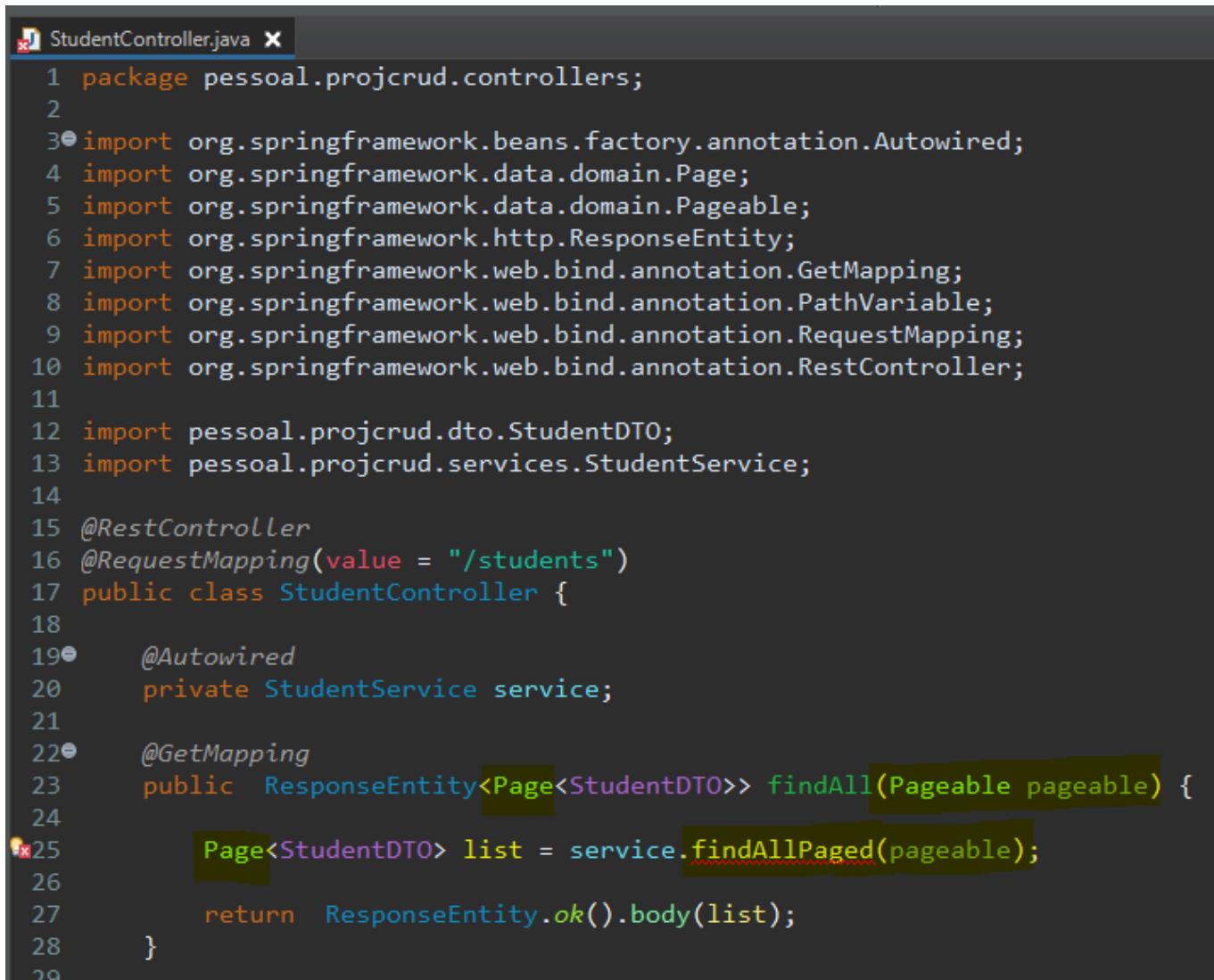
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Implemented exceptions"
[main d1e3560] Implemented exceptions
 4 files changed, 100 insertions(+), 1 deletion(-)
  create mode 100644 backend/src/main/java/pessoal/projcrud/controllers/exceptions/ResourceExceptionHandler.java
  create mode 100644 backend/src/main/java/pessoal/projcrud/controllers/exceptions/StandarError.java
  create mode 100644 backend/src/main/java/pessoal/projcrud/services/exceptions/ResourceNotFoundException.java

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 26, done.
Counting objects: 100% (26/26), done.
Delta compression using up to 28 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (16/16), 2.04 KiB | 2.04 MiB/s, done.
Total 16 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
 34dff09..d1e3560  main -> main
```

PAGINAÇÃO

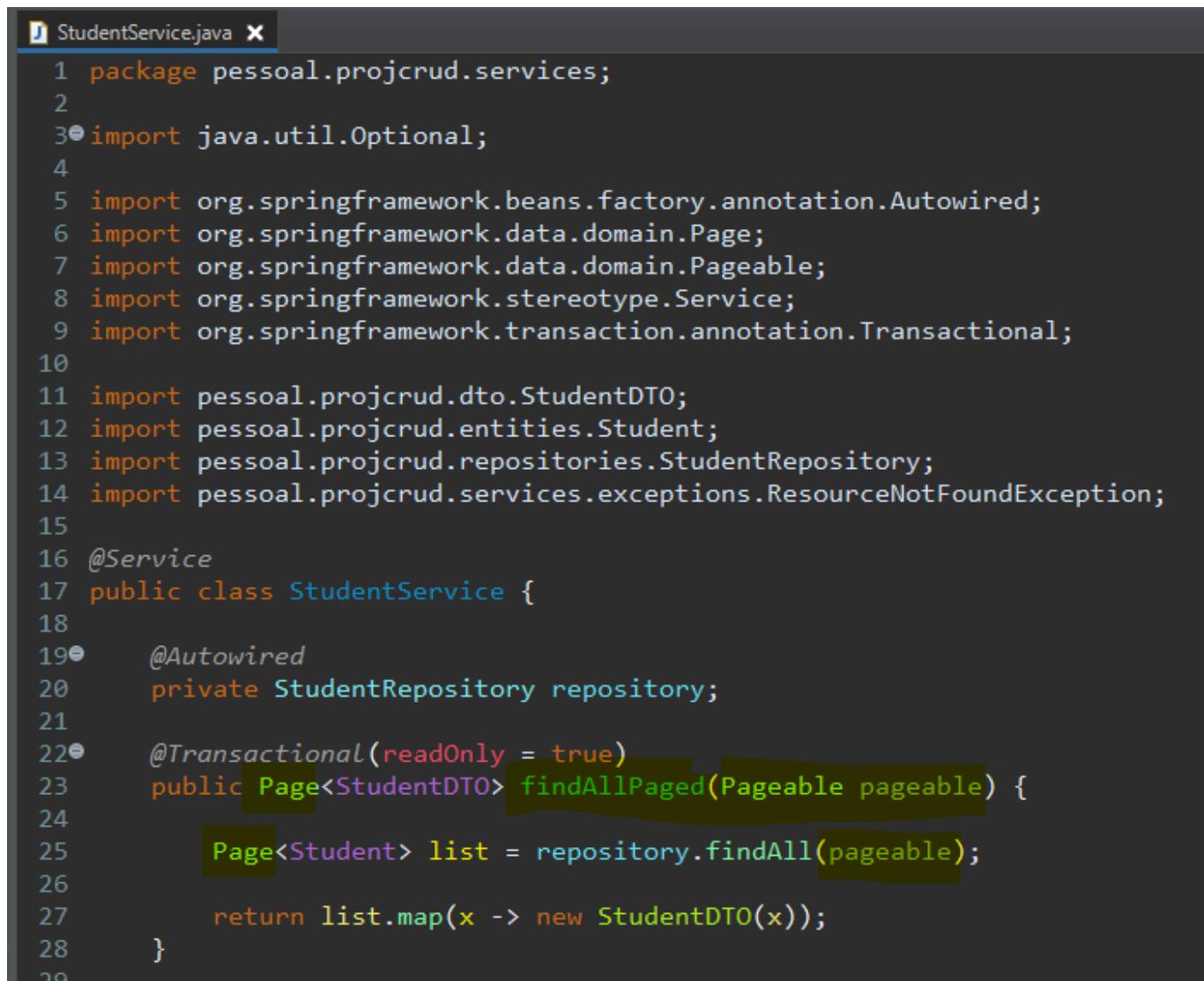
AJUSTAR O FIND_ALL PARA BUSCA PAGINADA

Implementar a busca paginada, no StudentController



```
StudentController.java
1 package pessoal.projcrud.controllers;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.data.domain.Page;
5 import org.springframework.data.domain.Pageable;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.PathVariable;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import pessoal.projcrud.dto.StudentDTO;
13 import pessoal.projcrud.services.StudentService;
14
15 @RestController
16 @RequestMapping(value = "/students")
17 public class StudentController {
18
19     @Autowired
20     private StudentService service;
21
22     @GetMapping
23     public ResponseEntity<Page<StudentDTO>> findAll(Pageable pageable) {
24
25         Page<StudentDTO> list = service.findAllPaged(pageable);
26
27         return ResponseEntity.ok().body(list);
28     }
29 }
```

Ajustar a busca paginada, no StudentService



```
1 package pessoal.projcrud.services;
2
3 import java.util.Optional;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.data.domain.Page;
7 import org.springframework.data.domain.Pageable;
8 import org.springframework.stereotype.Service;
9 import org.springframework.transaction.annotation.Transactional;
10
11 import pessoal.projcrud.dto.StudentDTO;
12 import pessoal.projcrud.entities.Student;
13 import pessoal.projcrud.repositories.StudentRepository;
14 import pessoal.projcrud.services.exceptions.ResourceNotFoundException;
15
16 @Service
17 public class StudentService {
18
19     @Autowired
20     private StudentRepository repository;
21
22     @Transactional(readOnly = true)
23     public Page<StudentDTO> findAllPaged(Pageable pageable) {
24
25         Page<Student> list = repository.findAll(pageable);
26
27         return list.map(x -> new StudentDTO(x));
28     }
29 }
```

Expandir o seed do banco para teste de paginação

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure for 'projcrud [boot]'. It includes 'src/main/java' with 'pessoal.projcrud' and 'src/main/resources' containing 'static', 'templates', 'application.properties', 'application-test.properties', and 'import.sql'. On the right, the editor view shows the content of the 'import.sql' file, which contains 24 INSERT statements for the 'tb_student' table, adding student records with names like Auricelio Freitas, Miguel Soares, Matheus Marques, Rafael Sales, Bruno Freitas, Romulo Araujo, Ryan Sousa, and test1, each with a random CPF and birth date.

```

1 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Auricelio Freitas', '000.111.222-22')
2 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Miguel Soares', '000.111.222-33')
3 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Matheus Marques', '000.111.222-3')
4 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Rafael Sales', '000.111.222-33')
5 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Bruno Freitas', '000.111.222-33')
6 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Romulo Araujo', '000.111.222-33')
7 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Ryan Sousa', '000.111.222-33', T
8 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('test1', '000.111.222-33', null,
9 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Auricelio Freitas', '000.111.222
10 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Miguel Soares', '000.111.222-33'
11 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Matheus Marques', '000.111.222-3
12 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Rafael Sales', '000.111.222-33'
13 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Bruno Freitas', '000.111.222-33'
14 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Romulo Araujo', '000.111.222-33'
15 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Ryan Sousa', '000.111.222-33', T
16 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('test1', '000.111.222-33', null,
17 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Auricelio Freitas', '000.111.222
18 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Miguel Soares', '000.111.222-33'
19 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Matheus Marques', '000.111.222-3
20 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Rafael Sales', '000.111.222-33'
21 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Bruno Freitas', '000.111.222-33'
22 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Romulo Araujo', '000.111.222-33'
23 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('Ryan Sousa', '000.111.222-33', T
24 INSERT INTO tb_student (name, cpf, birth_date, income) VALUES ('test1', '000.111.222-33', null,

```

NOTA: Neste caso, apenas replicar os existentes (copiar, colar).

Rodar o projeto

The screenshot shows the Eclipse IDE's Run View. The 'Boot Dashboard' tab is active, displaying the application 'projcrud - ProjcrudApplication [Spring Boot App]' running on port 8080. The 'Console' tab shows the application logs, which include several INFO messages from the 'projcrud' package indicating successful startup and initialization of various components like main, o.s, and pe.

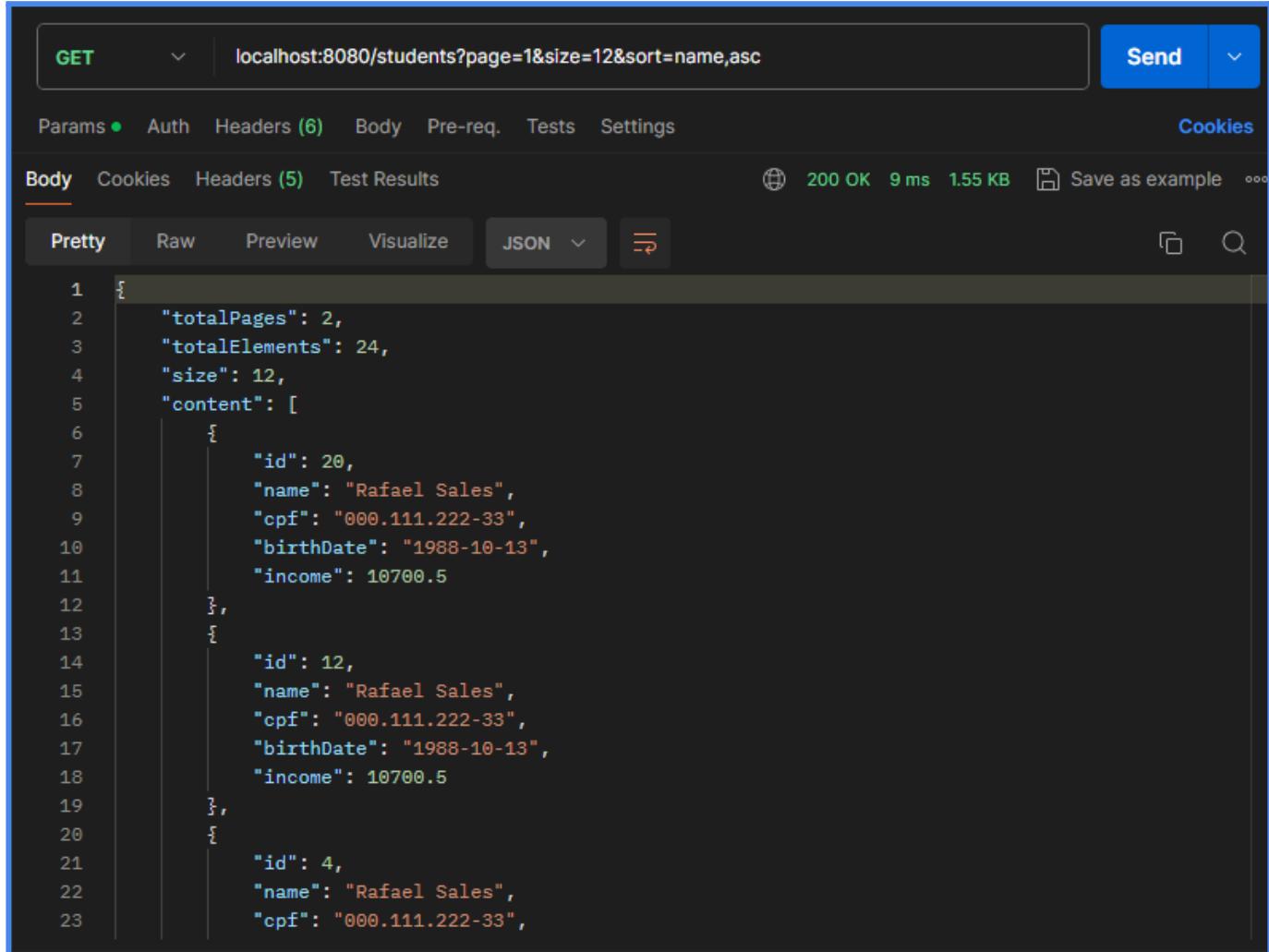
```

2024-06-27T16:03:33.017-03:00  INFO 5580 --- [projcrud] [main] j.
2024-06-27T16:03:33.282-03:00  INFO 5580 --- [projcrud] [main] o.
2024-06-27T16:03:33.287-03:00  INFO 5580 --- [projcrud] [main] pe
2024-06-27T16:03:41.015-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.015-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.016-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.

```

TESTAR NO POSTMAN

- GET: localhost:8080/students?page=0&size=12&sort=name,asc



GET localhost:8080/students?page=0&size=12&sort=name,asc Send

Params • Auth Headers (6) Body Pre-req. Tests Settings Cookies

Body Cookies Headers (5) Test Results 200 OK 9 ms 1.55 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 {  
2     "totalPages": 2,  
3     "totalElements": 24,  
4     "size": 12,  
5     "content": [  
6         {  
7             "id": 20,  
8             "name": "Rafael Sales",  
9             "cpf": "000.111.222-33",  
10            "birthDate": "1988-10-13",  
11            "income": 10700.5  
12        },  
13        {  
14            "id": 12,  
15            "name": "Rafael Sales",  
16            "cpf": "000.111.222-33",  
17            "birthDate": "1988-10-13",  
18            "income": 10700.5  
19        },  
20        {  
21            "id": 4,  
22            "name": "Rafael Sales",  
23            "cpf": "000.111.222-33",  
24        }  
25    ]  
26}
```

Github-11

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Implemented FindAllPaged”
 - git push

```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git add .

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Implemented FindAllPaged"
[main ea296b6] Implemented FindAllPaged
 3 files changed, 25 insertions(+), 9 deletions(-)

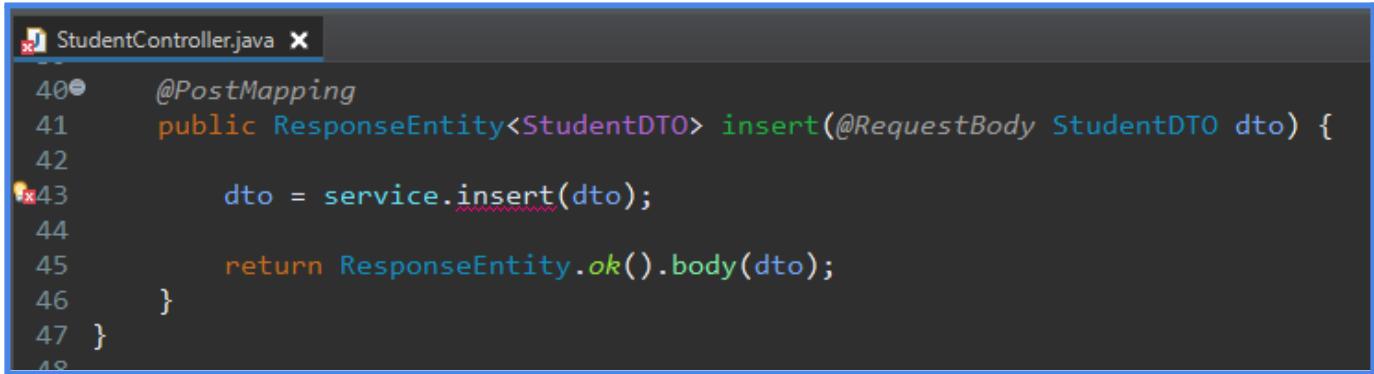
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 28 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (14/14), 1.48 KiB | 1.48 MiB/s, done.
Total 14 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
  d1e3560..ea296b6  main -> main

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$
```

ENDPOINT - INSERT

INSERIR NOVO ALUNO COM POST

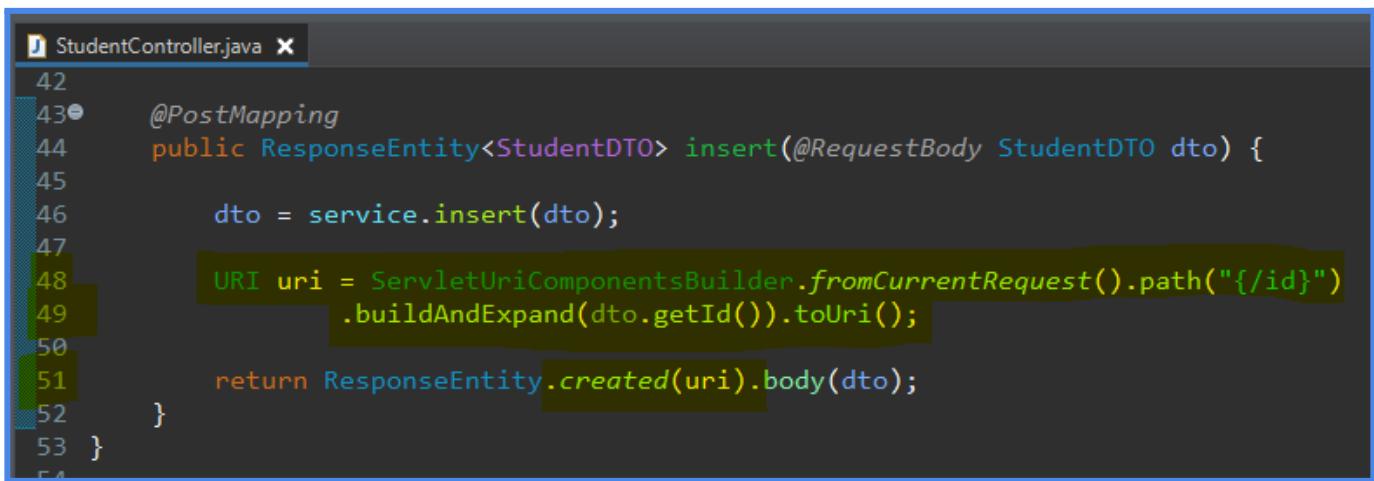
Implementar o insert, no StudentController



```
40     @PostMapping
41     public ResponseEntity<StudentDTO> insert(@RequestBody StudentDTO dto) {
42
43         dto = service.insert(dto);
44
45         return ResponseEntity.ok().body(dto);
46     }
47 }
```

Implementar a metodologia REST ao método

URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(dto.getId()).toUri();

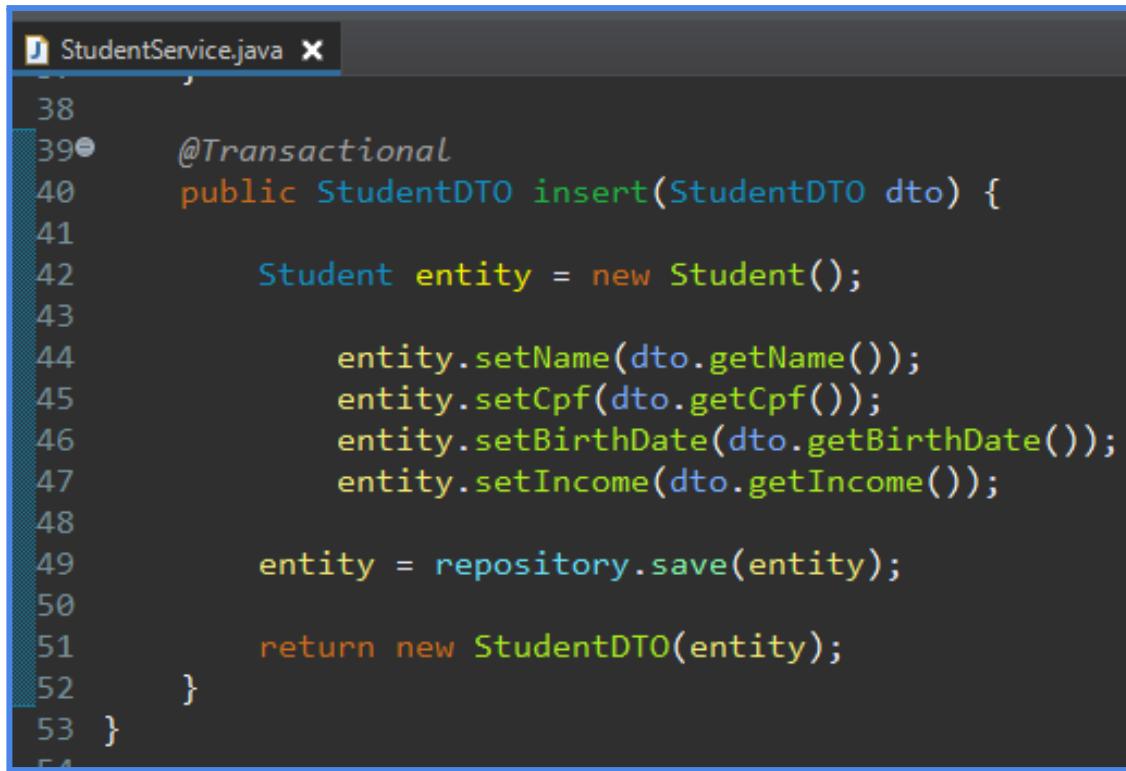


```
42
43     @PostMapping
44     public ResponseEntity<StudentDTO> insert(@RequestBody StudentDTO dto) {
45
46         dto = service.insert(dto);
47
48         URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}")
49             .buildAndExpand(dto.getId()).toUri();
50
51         return ResponseEntity.created(uri).body(dto);
52     }
53 }
```

OBS: Importar o URI do “java.net”

NOTA: Implantamos o caminho no Header da requisição e corrigimos o retorno de 200 (padrão) para 201 (recomendação REST), com o created.

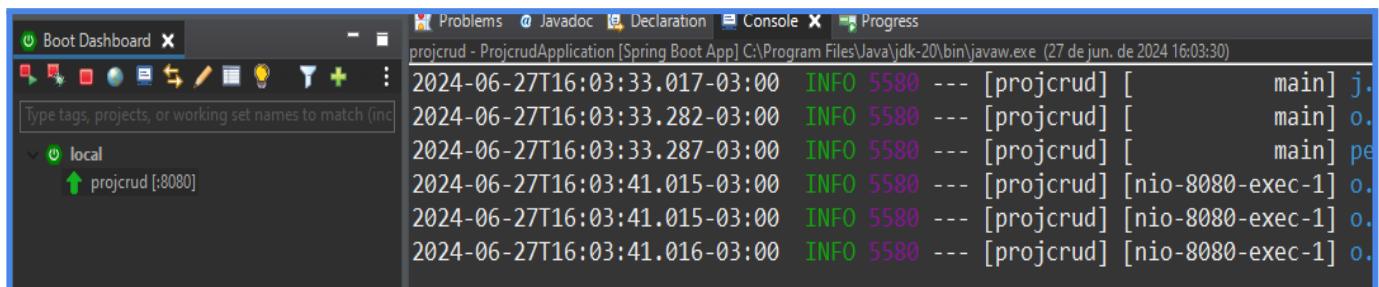
Implementar o método insert, convertendo o DTO para uma entidade, no StudentService



```
38
39     @Transactional
40     public StudentDTO insert(StudentDTO dto) {
41
42         Student entity = new Student();
43
44         entity.setName(dto.getName());
45         entity.setCpf(dto.getCpf());
46         entity.setBirthDate(dto.getBirthDate());
47         entity.setIncome(dto.getIncome());
48
49         entity = repository.save(entity);
50
51         return new StudentDTO(entity);
52     }
53 }
```

NOTA: Não colocar o atributo ID, pois é o banco que irá autoincrementar.

Rodar o projeto



Time	Level	Source	Message
2024-06-27T16:03:33.017-03:00	INFO	5580 --- [projcrud]	[main] j.
2024-06-27T16:03:33.282-03:00	INFO	5580 --- [projcrud]	[main] o.
2024-06-27T16:03:33.287-03:00	INFO	5580 --- [projcrud]	[main] pe
2024-06-27T16:03:41.015-03:00	INFO	5580 --- [projcrud]	[nio-8080-exec-1] o.
2024-06-27T16:03:41.015-03:00	INFO	5580 --- [projcrud]	[nio-8080-exec-1] o.
2024-06-27T16:03:41.016-03:00	INFO	5580 --- [projcrud]	[nio-8080-exec-1] o.

TESTAR NO POSTMAN

- *POST: localhost:8080/students*

- *Body:*

```
{  
    "name" : "Novo aluno",  
    "cpf" : "123.456.789-00",  
    "birthDate" : "2005-01-12T10:28:00Z",  
    "income" : 1920.0  
}
```

Inserir

The screenshot shows the Postman interface with a successful API call. The request method is POST, the URL is localhost:8080/students, and the body is a JSON object representing a new student. The response status is 201 Created, with a response body showing the newly created student with an id of 25.

POST localhost:8080/students

Params Auth Headers (8) Body **Pre-req.** Tests Settings Cookies Beautify

raw JSON

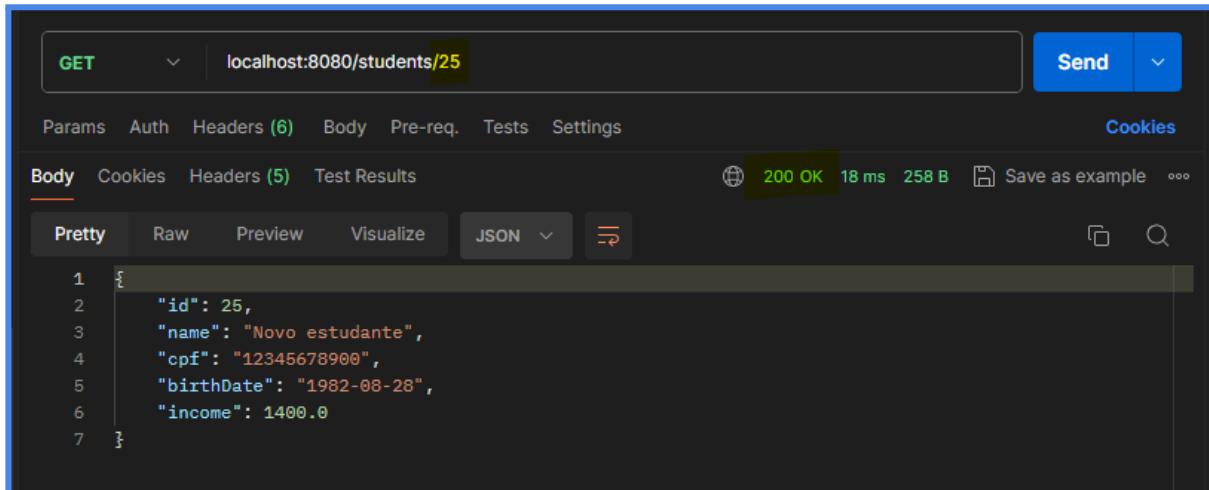
```
1 {  
2     "name" : "Novo estudante",  
3     "cpf" : "12345678900",  
4     "birthDate" : "1982-08-28T10:30:00Z",  
5     "income" : 1400.0  
6 }
```

Body Cookies Headers (6) Test Results 201 Created 150 ms 314 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {  
2     "id": 25,  
3     "name": "Novo estudante",  
4     "cpf": "12345678900",  
5     "birthDate": "1982-08-28",  
6     "income": 1400.0  
7 }
```

Buscar por Id



GET localhost:8080/students/25 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

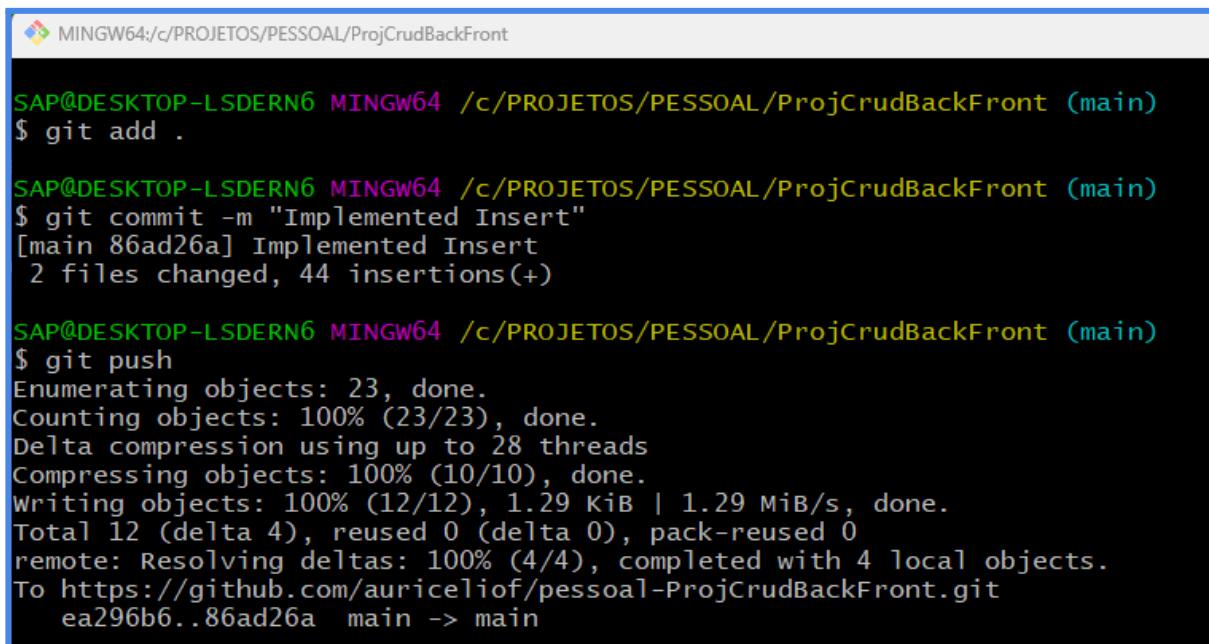
Body Cookies Headers (5) Test Results 200 OK 18 ms 258 B Save as example ...

Pretty Raw Preview Visualize JSON ↻ ⌂ ⌂

```
1 {  
2   "id": 25,  
3   "name": "Novo estudante",  
4   "cpf": "12345678900",  
5   "birthDate": "1982-08-28",  
6   "income": 1400.0  
7 }
```

Github-12

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Implemented Insert”
 - git push

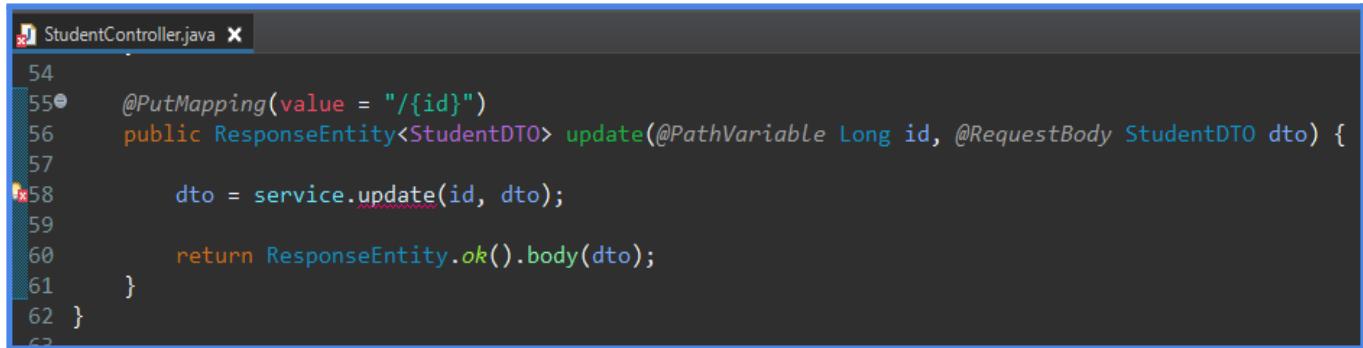


```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront  
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)  
$ git add .  
  
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)  
$ git commit -m "Implemented Insert"  
[main 86ad26a] Implemented Insert  
2 files changed, 44 insertions(+)  
  
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)  
$ git push  
Enumerating objects: 23, done.  
Counting objects: 100% (23/23), done.  
Delta compression using up to 28 threads  
Compressing objects: 100% (10/10), done.  
Writing objects: 100% (12/12), 1.29 KiB | 1.29 MiB/s, done.  
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.  
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git  
 ea296b6..86ad26a main -> main
```

ENDPOINT - UPDATE

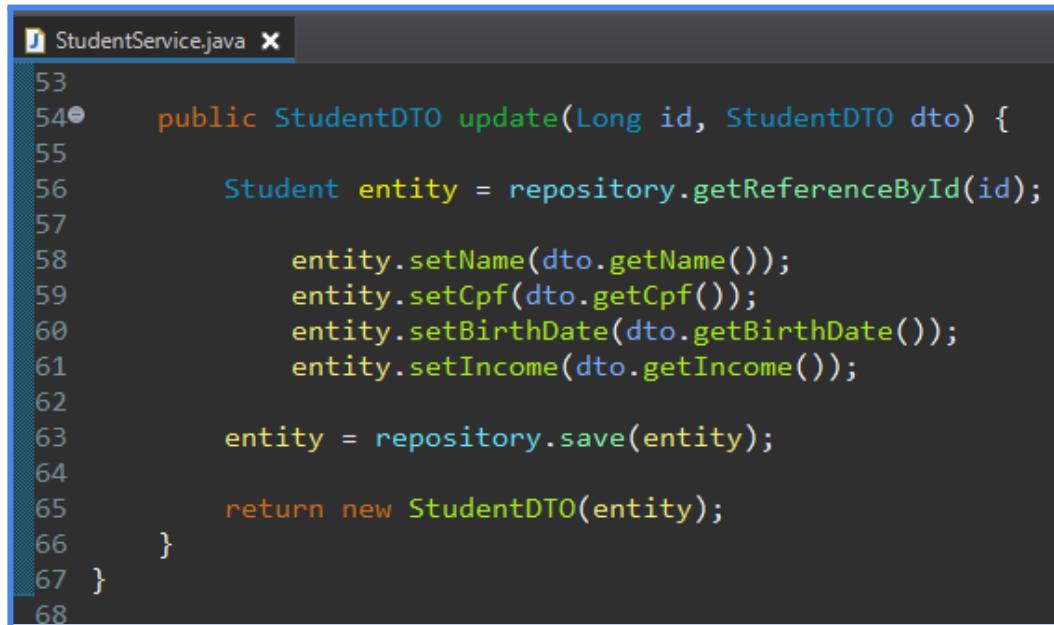
ATUALIZAR ALUNO COM PUT

Implementar o update, no StudentController



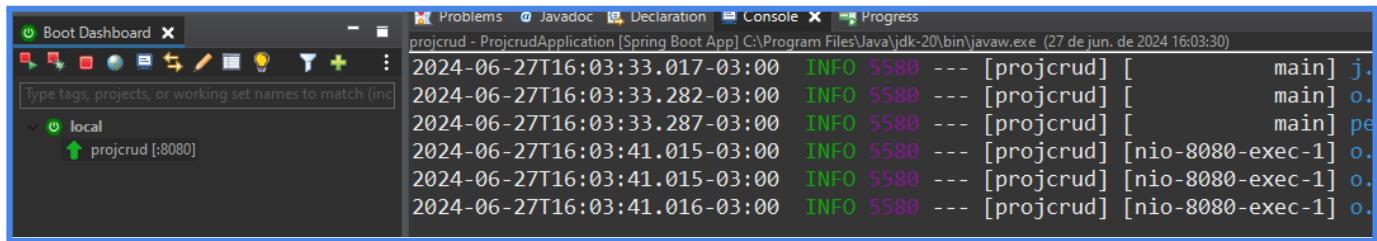
```
StudentController.java
54
55     @PutMapping(value = "/{id}")
56     public ResponseEntity<StudentDTO> update(@PathVariable Long id, @RequestBody StudentDTO dto) {
57
58         dto = service.update(id, dto);
59
60         return ResponseEntity.ok().body(dto);
61     }
62 }
```

Implementar o método update, no StudentService



```
StudentService.java
53
54     public StudentDTO update(Long id, StudentDTO dto) {
55
56         Student entity = repository.getReferenceById(id);
57
58         entity.setName(dto.getName());
59         entity.setCpf(dto.getCpf());
60         entity.setBirthDate(dto.getBirthDate());
61         entity.setIncome(dto.getIncome());
62
63         entity = repository.save(entity);
64
65         return new StudentDTO(entity);
66     }
67 }
```

Rodar o projeto



Boot Dashboard

Problems Javadoc Declaration Console Progress

projcrud - ProjcrudApplication [Spring Boot App] C:\Program Files\Java\jdk-20\bin\javaw.exe (27 de jun. de 2024 16:03:30)

```
2024-06-27T16:03:33.017-03:00 INFO 5580 --- [projcrud] [main] j. 2024-06-27T16:03:33.282-03:00 INFO 5580 --- [projcrud] [main] o. 2024-06-27T16:03:33.287-03:00 INFO 5580 --- [projcrud] [main] p. 2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o. 2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o. 2024-06-27T16:03:41.016-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
```

TESTAR NO POSTMAN

- *PUT: localhost:8080/students/1*

- *Body:*

```
{  
    "name" : "Aluno atualizado",  
    "cpf" : "123.456.789-00",  
    "birthDate" : "2005-01-12T10:28:00Z",  
    "income" : 1920.0  
}
```

Update

The screenshot shows the Postman interface for a PUT request. The URL is set to `localhost:8080/students/1`. The **Body** tab is selected, and the **JSON** option is chosen. The request body contains the following JSON data:

```
1 {  
2     "name" : "Estudante atualizado",  
3     "cpf" : "123.456.789-00",  
4     "birthDate" : "2005-01-12T10:28:00Z",  
5     "income" : 1920.0  
6 }
```

The response status is **200 OK** with **289 ms** latency and **276 B** size.

Busca por ID

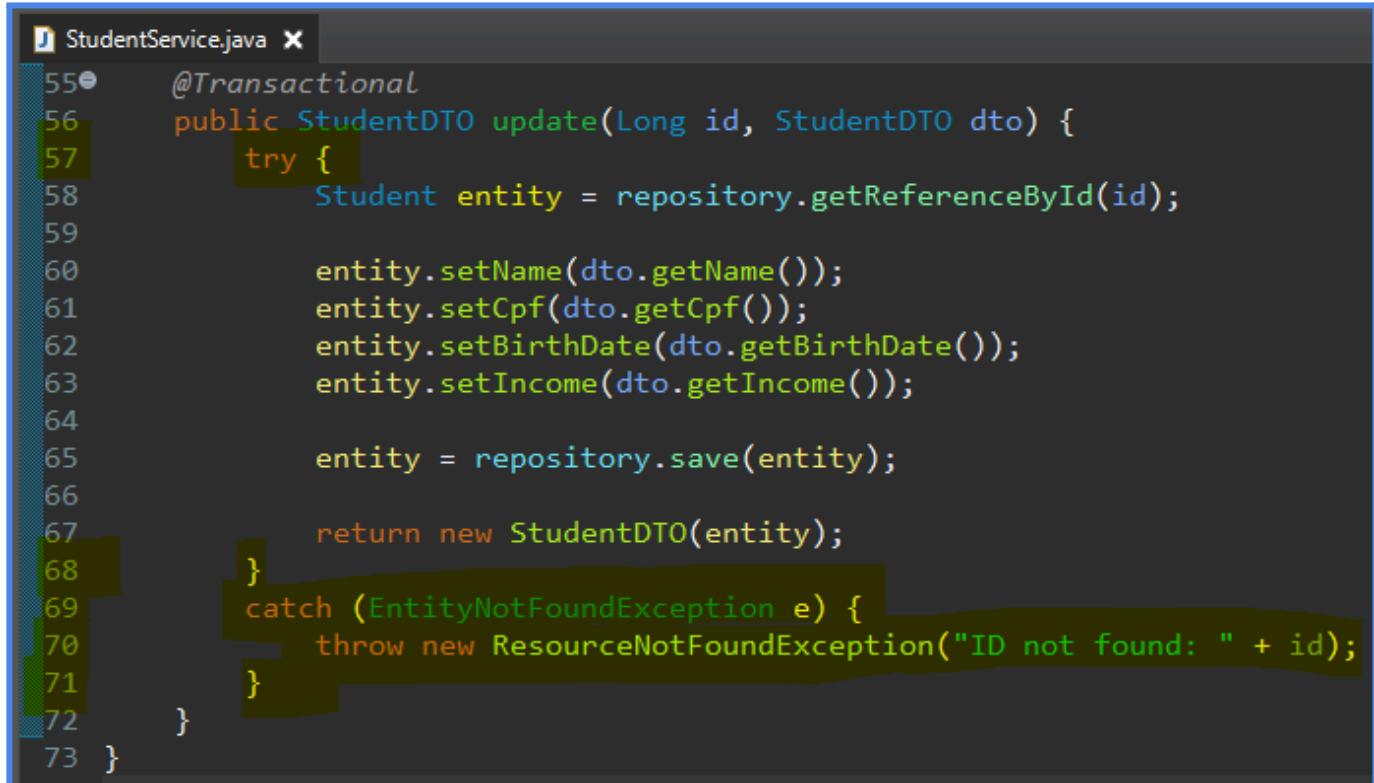
The screenshot shows the Postman interface for a GET request. The URL is set to `localhost:8080/students/1`. The **Body** tab is selected, and the **JSON** option is chosen. The response body contains the following JSON data:

```
1 {  
2     "id": 1,  
3     "name": "Estudante atualizado",  
4     "cpf": "123.456.789-00",  
5     "birthDate": "2005-01-12T10:28:00Z",  
6     "income": 1920.0  
7 }
```

The response status is **200 OK** with **11 ms** latency and **276 B** size.

TRATAMENTO DE ERRO PARA O UPDATE

Implementar o tratamento para ID Não encontrado

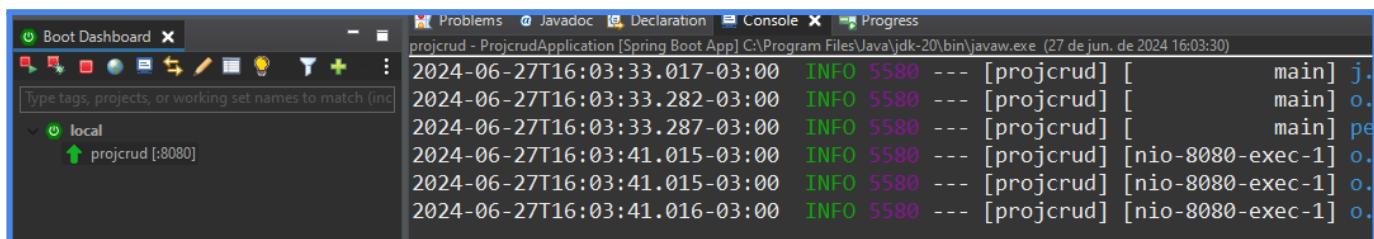


```

55     @Transactional
56     public StudentDTO update(Long id, StudentDTO dto) {
57         try {
58             Student entity = repository.getReferenceById(id);
59
60             entity.setName(dto.getName());
61             entity.setCpf(dto.getCpf());
62             entity.setBirthDate(dto.getBirthDate());
63             entity.setIncome(dto.getIncome());
64
65             entity = repository.save(entity);
66
67             return new StudentDTO(entity);
68         } catch (EntityNotFoundException e) {
69             throw new ResourceNotFoundException("ID not found: " + id);
70         }
71     }
72 }
73 }
```

NOTA: Devemos colocar o método update num bloco “Try Catch”, pois ao atualizar um ID, este pode não existir.

Rodar o projeto



```

2024-06-27T16:03:33.017-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] j.
2024-06-27T16:03:33.282-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:33.287-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] pe
2024-06-27T16:03:41.015-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.015-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.016-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
```

TESTAR NO POSTMAN

- *PUT: localhost:8080/students/100*

Update

The screenshot shows the Postman interface with a 'PUT' method selected. The URL is 'localhost:8080/students/100'. The response status is '404 Not Found' with a timestamp of '2024-06-28T18:39:27.152491700Z', status '404', error 'Resource not found', message 'ID not found: 100', and path '/students/100'.

Github-13

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Implemented Update with exceptions”
 - git push

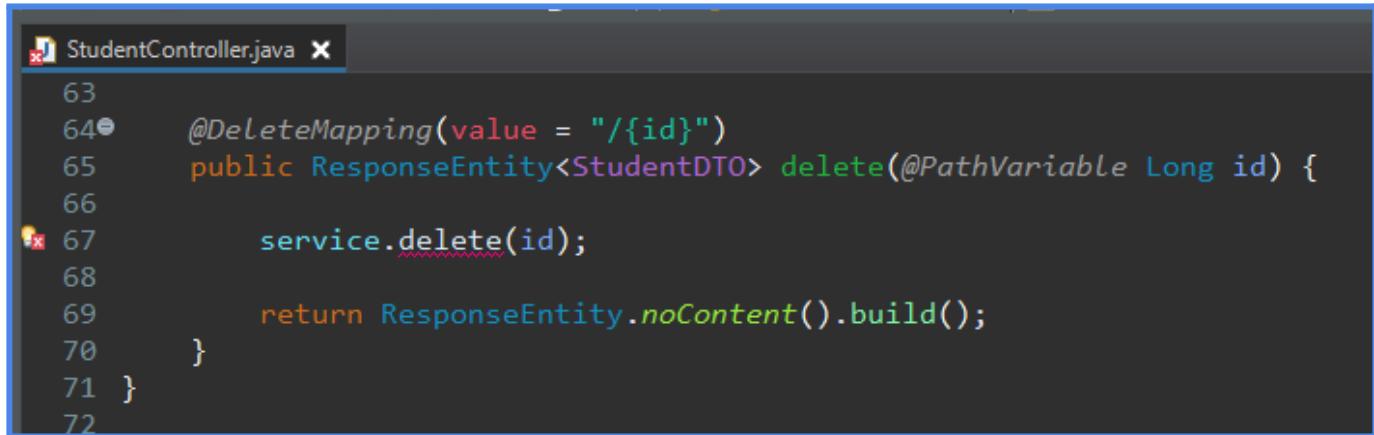
```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront
$ git add .
$ git commit -m "Implemented Update with exceptions"
[main f3862e6] Implemented Update with exceptions
 2 files changed, 30 insertions(+), 1 deletion(-)

MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 28 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 1.16 KiB | 1.16 MiB/s, done.
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
 86ad26a..f3862e6 main -> main
```

ENDPOINT - DELETE

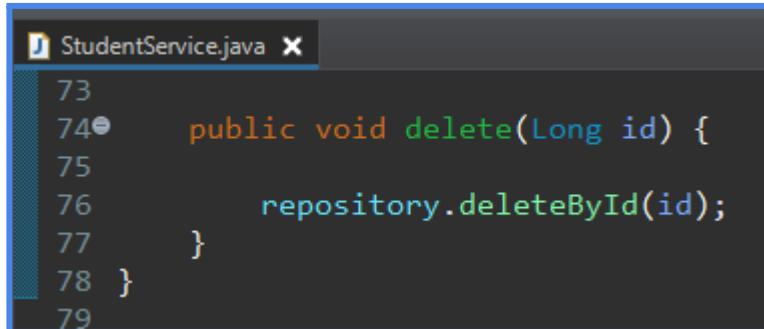
DELETAR UM ALUNO COM O MÉTODO REST DELETE

Implementar o update, no StudentController



```
StudentController.java
63
64  @DeleteMapping(value = "/{id}")
65  public ResponseEntity<StudentDTO> delete(@PathVariable Long id) {
66
67      service.delete(id);
68
69      return ResponseEntity.noContent().build();
70  }
71 }
72 }
```

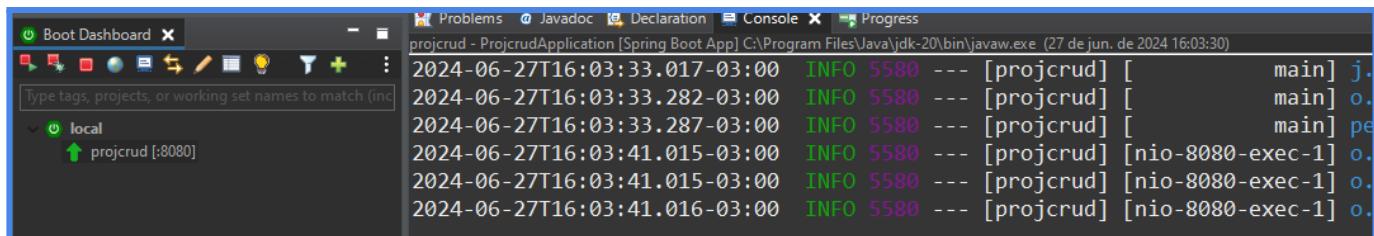
Implementar o método update, no StudentService



```
StudentService.java
73
74  public void delete(Long id) {
75
76      repository.deleteById(id);
77  }
78 }
79 }
```

NOTA: No método delete não colocamos o @Transactional.

Rodar o projeto



```
Boot Dashboard
Type tags, projects, or working set names to match (inc
local
projcrud [:8080]

Console
2024-06-27T16:03:33.017-03:00  INFO 5580 --- [projcrud] [main] j.
2024-06-27T16:03:33.282-03:00  INFO 5580 --- [projcrud] [main] o.
2024-06-27T16:03:33.287-03:00  INFO 5580 --- [projcrud] [main] pe
2024-06-27T16:03:41.015-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.015-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.016-03:00  INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
```

TESTAR NO POSTMAN

The screenshot shows the Postman interface. The URL in the header is `localhost:8080/students/2`. The status bar at the bottom indicates a `204 No Content` response with `117 ms` and `112 B`. The body section shows the number `1`.

TRATAMENTO DE ERRO DO DELETE

Implementar o tratamento para ID Não encontrado

```
StudentService.java
74
75    public void delete(Long id) {
76
77        try {
78            repository.deleteById(id);
79        }
80        catch (EmptyResultDataAccessException e) {
81            throw new ResourceNotFoundException("ID not found: " + id);
82        }
83    }
84 }
```

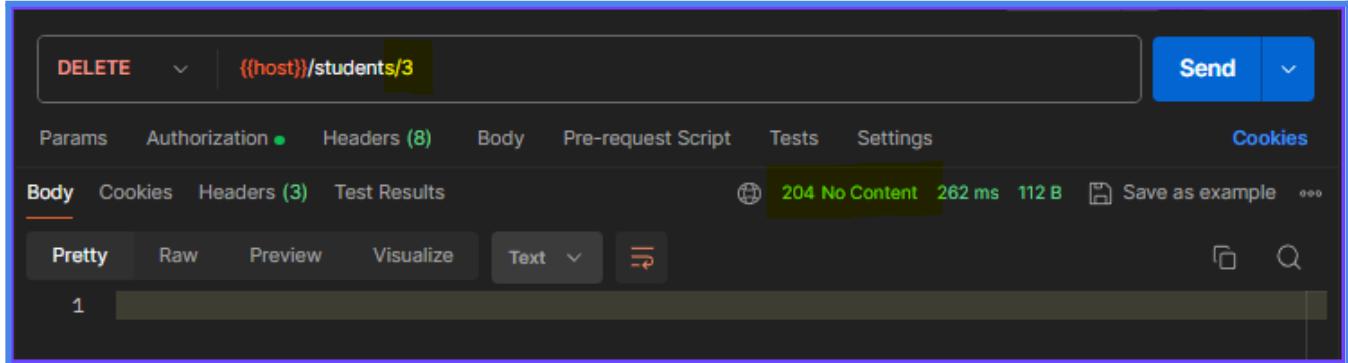
NOTA: Devemos colocar o método `delete` num bloco “Try Catch”, pois ao deletar um `ID`, este pode não existir.

OBS: Neste projeto, não iremos implementar o tratamento de erro para “integridade referencial”, pois a classe `student` não se relaciona com nenhuma outra, ao menos por enquanto.

Rodar o projeto

```
2024-06-27T16:03:33.017-03:00 INFO 5580 --- [projcrud] [main] j. 2024-06-27T16:03:33.282-03:00 INFO 5580 --- [projcrud] [main] o. 2024-06-27T16:03:33.287-03:00 INFO 5580 --- [projcrud] [main] pe 2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o. 2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o. 2024-06-27T16:03:41.016-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
```

TESTAR NO POSTMAN



Github-14

- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Implemented Delete”
 - git push

```
MINGW64:/c/PROJETOS/PESSOAL/ProjCrudBackFront
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git add .

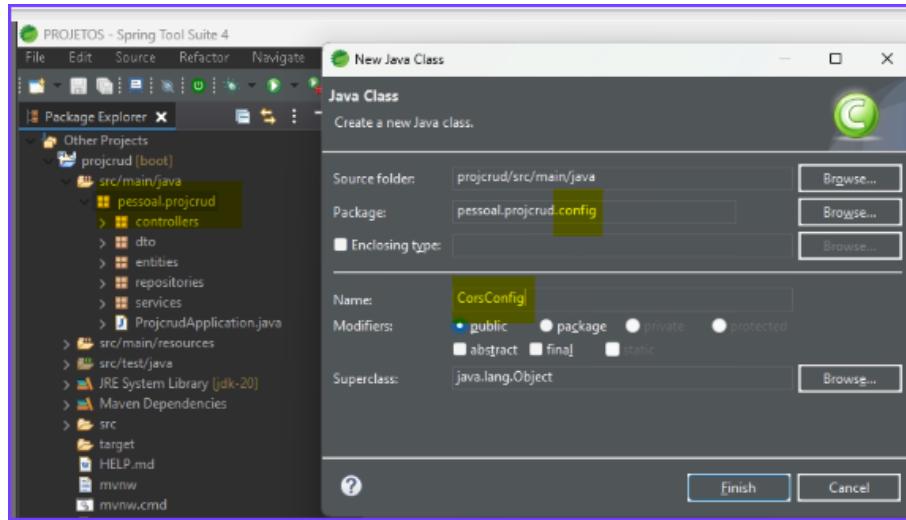
SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git commit -m "Implemented Delete"
[main 1d8929a] Implemented Delete
 2 files changed, 27 insertions(+)

SAP@DESKTOP-LSDERN6 MINGW64 /c/PROJETOS/PESSOAL/ProjCrudBackFront (main)
$ git push
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 28 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 1.11 KiB | 1.11 MiB/s, done.
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/auriceliof/pessoal-ProjCrudBackFront.git
 f3862e6..1d8929a main -> main
```

LIBERAÇÃO CORS PARA FRONTEND

IMPLEMENTAR O CONFIG

Criar o CorsConfig



Implementar o CorsConfig

```
1 package pessoal.projcrud.config;
2 import org.springframework.context.annotation.Bean;
3 import org.springframework.context.annotation.Configuration;
4 import org.springframework.web.cors.CorsConfiguration;
5 import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
6 import org.springframework.web.filter.CorsFilter;
7
8 @Configuration
9 public class CorsConfig {
10
11     @Bean
12     public CorsFilter corsFilter() {
13         UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
14         CorsConfiguration config = new CorsConfiguration();
15
16         // Permitir solicitações de qualquer origem (use * para permitir de qualquer origem)
17         config.addAllowedOrigin("*");
18
19         // Permitir solicitações GET, POST, PUT, DELETE e OPTIONS
20         config.addAllowedMethod("*");
21
22         // Permitir os cabeçalhos específicos que seu aplicativo React envia
23         config.addAllowedHeader("*");
24
25         source.registerCorsConfiguration("/**", config);
26         return new CorsFilter(source);
27     }
28 }
```

The code editor window shows the 'CorsConfig.java' file. The class implements the 'CorsConfiguration' interface and registers it with a 'CorsFilter'. The configuration allows all origins, methods, and headers.

Github-15

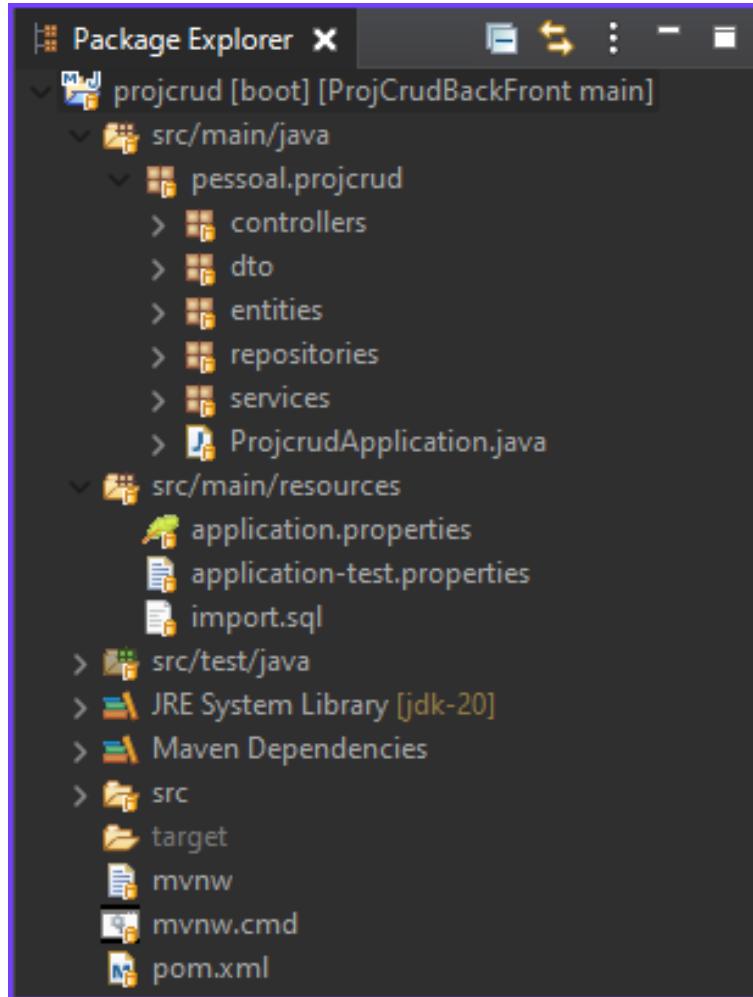
- “Git bash here” no diretório do projeto
 - git add backend
 - git commit -m “Implemented CORS”
 - git push

IMPLEMENTAR O SWAGGER

O objetivo deste capítulo é implementar o swagger no Spring Tool Suite 4.

O Swagger é uma ferramenta amplamente utilizada para criar, visualizar e manter a documentação de APIs de maneira automatizada.

PROJETO

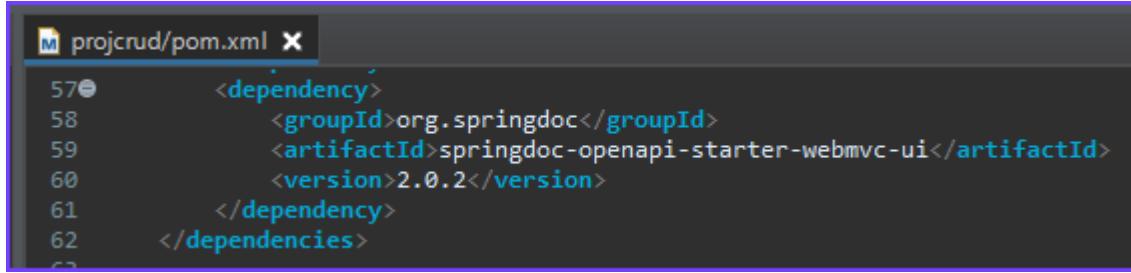


NOTA: Neste momento, o projeto já deve estar em funcionamento, com todos os endpoints respondendo.

IMPLEMENTAÇÃO DO SWAGGER

DEPENDÊNCIA MAVEN

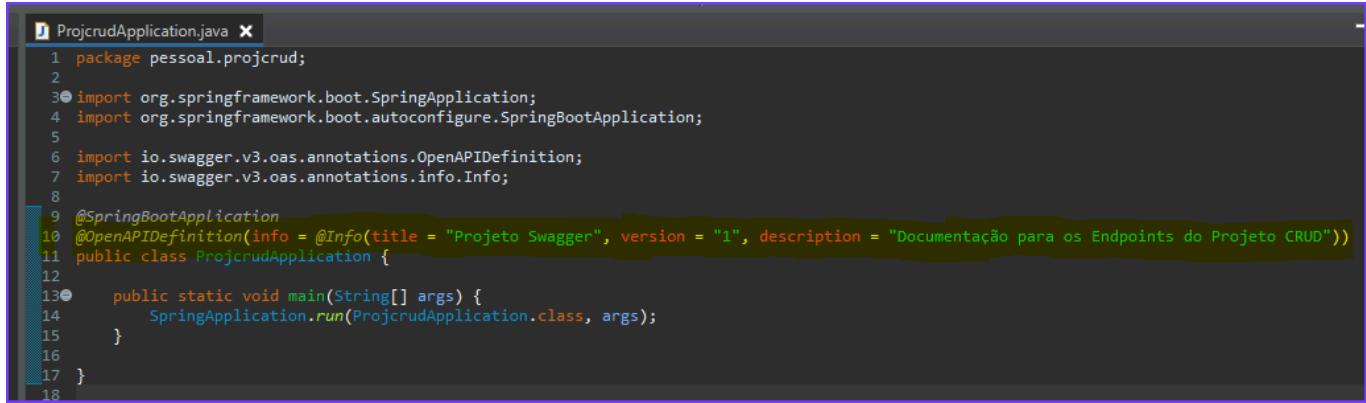
```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.0.2</version>
</dependency>
```



```
projcrud/pom.xml
57     <dependency>
58         <groupId>org.springdoc</groupId>
59         <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
60         <version>2.0.2</version>
61     </dependency>
62 </dependencies>
```

IMPLEMENTAR O MAIN PRINCIPAL

```
@OpenAPIDefinition(info = @Info(title = "Projeto Swagger", version = "1", description = "Documentação para os Endpoints do Projeto CRUD"))
```

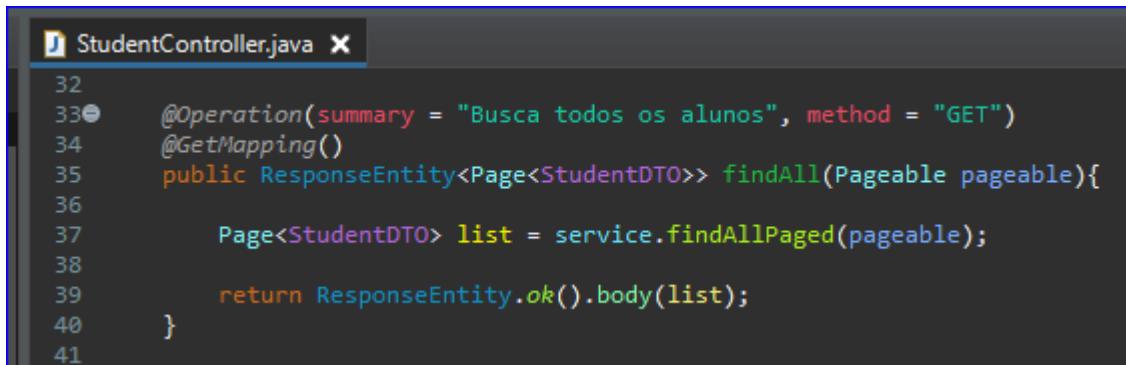


```
ProjrudApplication.java
1 package pessoal.projrud;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 import io.swagger.v3.oas.annotations.OpenAPIDefinition;
7 import io.swagger.v3.oas.annotations.info.Info;
8
9 @SpringBootApplication
10 @OpenAPIDefinition(info = @Info(title = "Projeto Swagger", version = "1", description = "Documentação para os Endpoints do Projeto CRUD"))
11 public class ProjrudApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(ProjrudApplication.class, args);
15     }
16
17 }
```

IMPLEMENTAR O CONTROLLER

FindAll

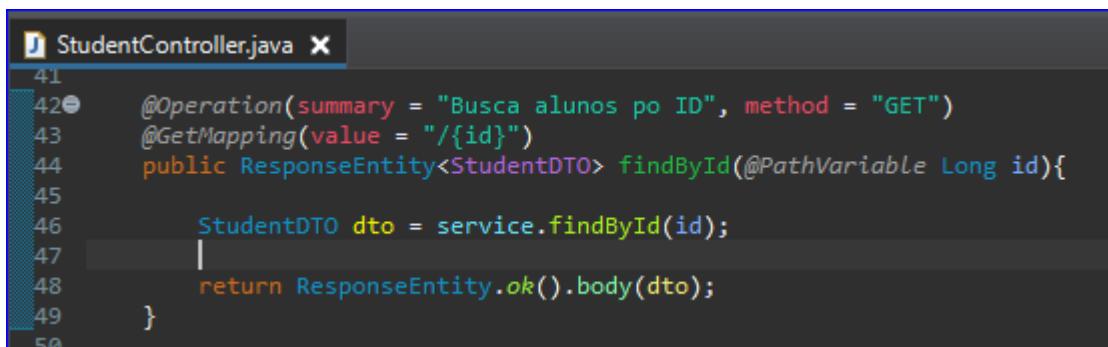
@Operation(summary = "Busca todos os alunos", method = "GET")



```
StudentController.java
32
33     @Operation(summary = "Busca todos os alunos", method = "GET")
34     @GetMapping()
35     public ResponseEntity<Page<StudentDTO>> findAll(Pageable pageable){
36
37         Page<StudentDTO> list = service.findAllPaged(pageable);
38
39         return ResponseEntity.ok().body(list);
40     }
41
```

FindByld

@Operation(summary = "Busca alunos po ID", method = "GET")



```
StudentController.java
41
42     @Operation(summary = "Busca alunos po ID", method = "GET")
43     @GetMapping(value = "/{id}")
44     public ResponseEntity<StudentDTO> findById(@PathVariable Long id){
45
46         StudentDTO dto = service.findById(id);
47
48         return ResponseEntity.ok().body(dto);
49     }
50
```

Insert

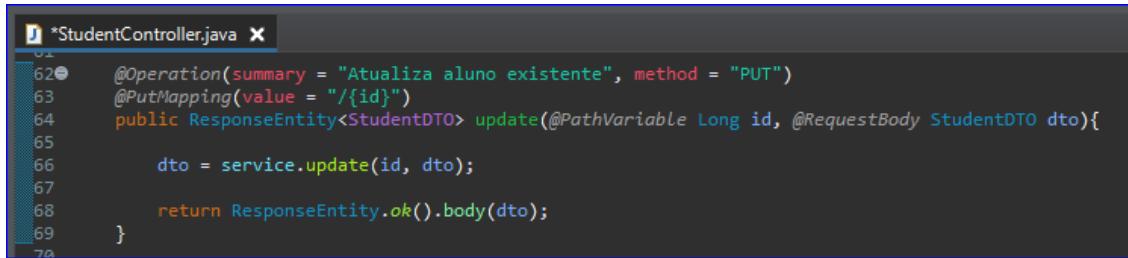
@Operation(summary = "Cadastra novos alunos", method = "POST")



```
*StudentController.java
50
51     @Operation(summary = "Cadastra novos alunos", method = "POST")
52     @PostMapping
53     public ResponseEntity<StudentDTO> insert(@RequestBody StudentDTO dto){
54
55         dto = service.insert(dto);
56
57         URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("{/id}").buildAndExpand(dto.getId()).toUri();
58
59         return ResponseEntity.created(uri).body(dto);
60     }
61
```

Update

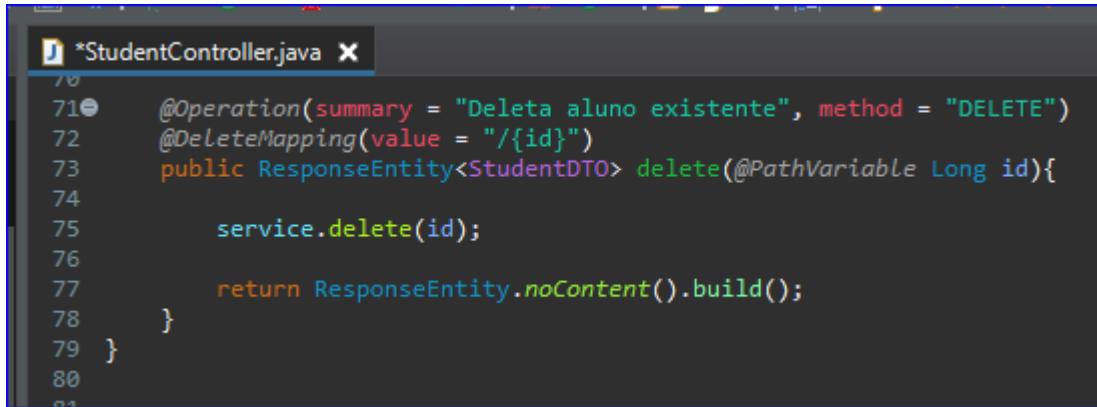
@Operation(summary = "Atualiza aluno existente", method = "PUT")



```
 61
 62     @Operation(summary = "Atualiza aluno existente", method = "PUT")
 63     @PutMapping(value = "/{id}")
 64     public ResponseEntity<StudentDTO> update(@PathVariable Long id, @RequestBody StudentDTO dto){
 65
 66         dto = service.update(id, dto);
 67
 68         return ResponseEntity.ok().body(dto);
 69     }
 70
 71
```

Delete

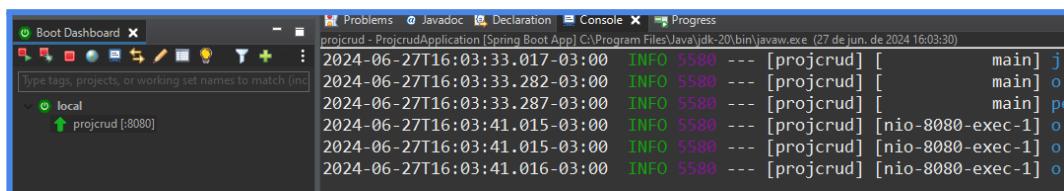
@Operation(summary = "Delete aluno existente", method = "DELETE")



```
 71     @Operation(summary = "Delete aluno existente", method = "DELETE")
 72     @DeleteMapping(value = "/{id}")
 73     public ResponseEntity<StudentDTO> delete(@PathVariable Long id){
 74
 75         service.delete(id);
 76
 77         return ResponseEntity.noContent().build();
 78     }
 79
 80
 81
```

ACESSAR O SWAGGER

Rodar o projeto



The screenshot shows the Eclipse IDE interface. On the left, the 'Project Explorer' view displays a local workspace with a project named 'projcrud'. In the center, the 'Java Console' view shows the application's log output:

```
2024-06-27T16:03:33.017-03:00 INFO 5580 --- [projcrud] [main] j.
2024-06-27T16:03:33.282-03:00 INFO 5580 --- [projcrud] [main] o.
2024-06-27T16:03:33.287-03:00 INFO 5580 --- [projcrud] [main] pe
2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.015-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
2024-06-27T16:03:41.016-03:00 INFO 5580 --- [projcrud] [nio-8080-exec-1] o.
```

Acesso online

- <http://localhost:8080/swagger-ui/index.html#>

The screenshot shows the Swagger UI interface for a RESTful API. At the top, there's a header with the URL "localhost:8080/swagger-ui/index.html#", a zoom level of "60%", and a star icon. Below the header, the title "Projeto Swagger" is displayed, along with a "Servers" dropdown set to "http://localhost:8080 - Generated server url".

The main content area is titled "student-controller". It lists several API endpoints:

- GET /students/{id} Busca alunos po ID
- PUT /students/{id} Atualiza aluno existente
- DELETE /students/{id} Deleta aluno existente
- GET /students Busca todos os alunos
- POST /students Cadastra novos alunos

Below the endpoints, there's a section titled "Schemas" which lists the following schema definitions:

- StudentDTO >
- Pageable >
- PageStudentDTO >
- PageableObject >
- SortObject >

Testar o Swagger

Buscar todos

The screenshot shows the Swagger UI interface for a 'students' endpoint. At the top, it says 'GET /students Busca todos os alunos'. Below this, there's a 'Parameters' section with a table. One parameter is listed: 'pageable * required object (query)'. The schema for 'pageable' is shown as:

```
{
  "page": 0,
  "size": 1,
  "sort": [
    {
      "string"
    }
  ]
}
```

At the bottom of the screen are two buttons: 'Execute' (highlighted in yellow) and 'Clear'.

The screenshot shows the response for the 'Buscar todos' endpoint. It indicates a '200 OK' status. Under 'Media type', there is a dropdown set to '*/*'. Below it, there is a note: 'Controls Accept header.' and 'Example Value | Schema'. The 'Example Value' section displays a JSON object representing a pageable object:

```
{
  "totalElements": 0,
  " totalPages": 0,
  "size": 0,
  "content": [
    {
      "id": 0,
      "name": "string",
      "cpf": "string",
      "birthdate": "2024-06-30",
      "income": 0
    }
  ],
  "number": 0,
  "sort": {
    "unsorted": true,
    "empty": true,
    "sorted": true
  },
  "pageable": {
    "paged": true,
    "pageSize": 0,
    "offset": 0,
    "sort": {
      "unsorted": true,
      "empty": true,
      "sorted": true
    },
    "unpaged": true
  }
}
```

Busca por ID

The screenshot shows a REST API client interface. At the top, a blue bar indicates a **GET** request to the endpoint **/students/{id}**. Below this, under the **Parameters** section, there is a field for the parameter **id**, which is marked as required and has the value **2** entered. To the right of the input field is a yellow button labeled **Execute**. In the bottom right corner of the execute button, there is a small yellow box highlighting the word "Execute". After executing the request, the **Responses** section is displayed. It includes a **Curl** command, a **Request URL** (http://localhost:8080/students/2), and a **Server response**. The server response shows a **Code** of **200** and a **Details** section containing a JSON object representing a student. The JSON object is:

```
{ "id": 2, "name": "Miguel Soares", "cpf": "000.111.222-33", "birthdate": "1988-02-10", "income": 10700.5 }
```

Below the JSON object, there is a **Response headers** section showing standard HTTP headers.

Atualizar

The screenshot shows a REST API testing interface with the following details:

- Path:** /students/{id} - Atualiza aluno estudante
- Parameters:**
 - Name:** id **Description:** integer (lista) (path)
 - Value: 5
- Request body (required):**

```
{
  "id": 0,
  "name": "string",
  "cpf": "string",
  "birthdate": "2024-06-08",
  "income": 0
}
```
- Content-Type:** application/json
- Buttons:** Execute (highlighted in yellow), Clear
- Responses:**
 - Curl:**

```
curl -X 'PUT' \
  'http://localhost:8000/students/5' \
  -H 'Accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "name": "string",
    "cpf": "string",
    "birthdate": "2024-06-08",
    "income": 0
  }'
```
 - Request URL:** http://localhost:8000/students/5
 - Server response:**

Code	Details
200	Response body: <pre>[{ "id": 0, "name": "string", "cpf": "string", "birthdate": "2024-06-08", "income": 0 }]</pre> Download

Inserir

The screenshot shows a REST API client interface with a purple border. At the top, a green header bar displays "POST /students Cadastra novos alunos". Below this, a "Parameters" section indicates "No parameters". The "Request body" section is labeled "required" and contains a JSON schema:

```
{
  "id": 0,
  "name": "string",
  "cpf": "string",
  "birthdate": "2024-06-30",
  "income": 0
}
```

On the right side of the "Request body" section is a dropdown menu set to "application/json". Below the request body is a blue button labeled "Execute" and a "Clear" button.

Under the "Responses" section, there is a "Curl" code block:

```
curl -X 'POST' \
  'http://localhost:8080/students' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "name": "string",
    "cpf": "string",
    "birthdate": "2024-06-30",
    "income": 0
}'
```

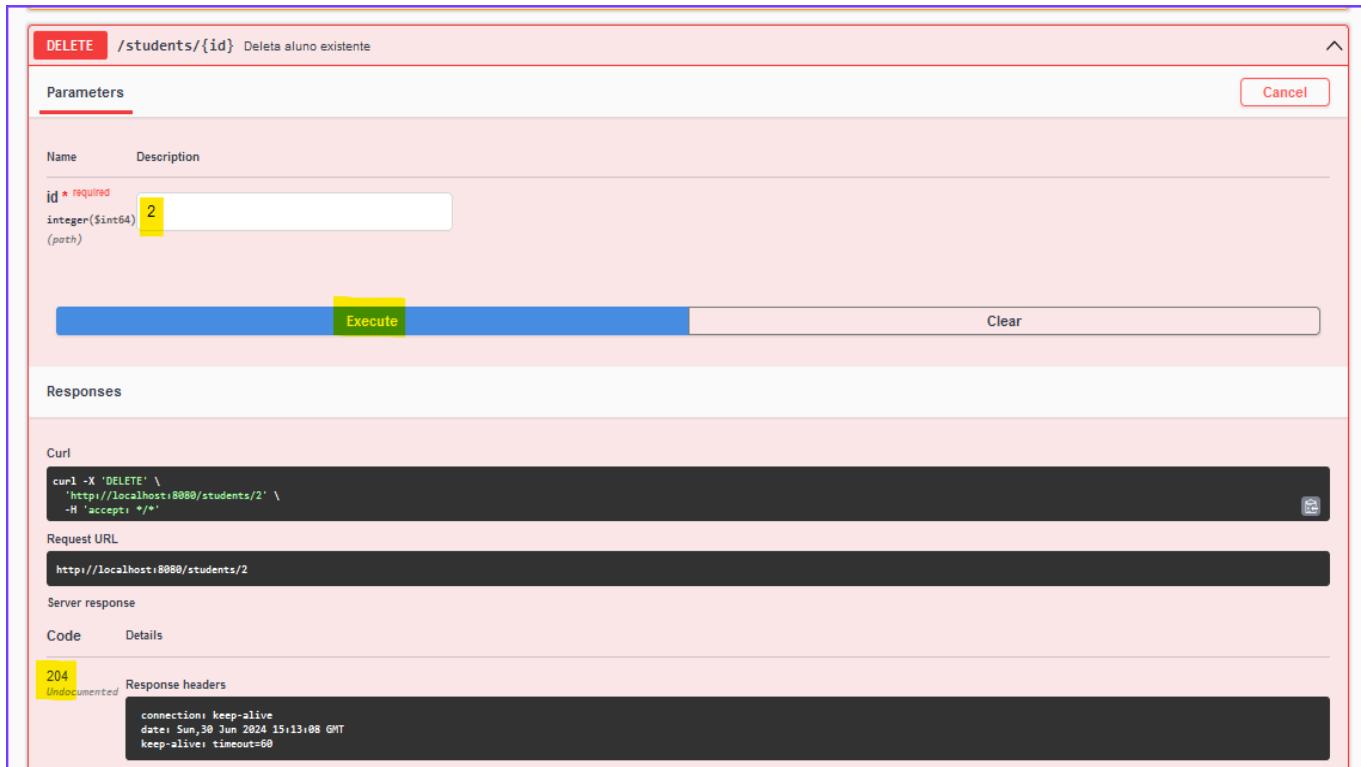
Below the curl command is a "Request URL" field containing "http://localhost:8080/students".

The "Server response" section shows a table with two columns: "Code" and "Details". A yellow box highlights the "201 Undocumented" row. Under "Response body", there is another JSON schema:

```
[
  {
    "id": 25,
    "name": "string",
    "cpf": "string",
    "birthdate": "2024-06-30",
    "income": 0
}
```

On the right side of this section are "Copy" and "Download" buttons.

Deletar



The screenshot shows a Swagger UI interface for a DELETE request to the endpoint `/students/{id}`. The endpoint is described as "Delete aluno existente". The `id` parameter is required and has a type of `integer($int64)`. The value `2` is entered in the input field. Below the parameters, there is an `Execute` button, which is highlighted with a yellow background. To the right of the `Execute` button is a `Cancel` button. The interface also includes sections for Responses, Curl, Request URL, and Server response.

Github-16

- “Git bash here” no diretório do projeto
 - `git add backend`
 - `git commit -m “Implemented SWAGGER”`
 - `git push`

Fim