

GUIA EXEMPLO PRÁTICO DE DESIGN INSEGURO

Para acesso à API, clone o repositório:

```
git clone https://github.com/jjgirotto/insecure-design.git
```

Requisitos para execução do projeto:

- Java JDK 21 ou superior;
- Maven;
- Postman (para requisições).

1. Abra o arquivo '*src/main/resources/application.yml*' e altere as configurações do banco de dados MySQL Workbench para utilização na sua máquina:

```
username: root (insira o usuário desejado)
password: 123456 (insira a senha desejada)
url: jdbc:mysql://localhost:3306/alunos?createDatabaseIfNotExist=true&serverTimezone=UTC
(altere a porta se necessário em 'localhost:3306')
```

2. Execute os seguintes comandos:

```
cd appalunos
mvn clean install
mvn spring-boot:run
```

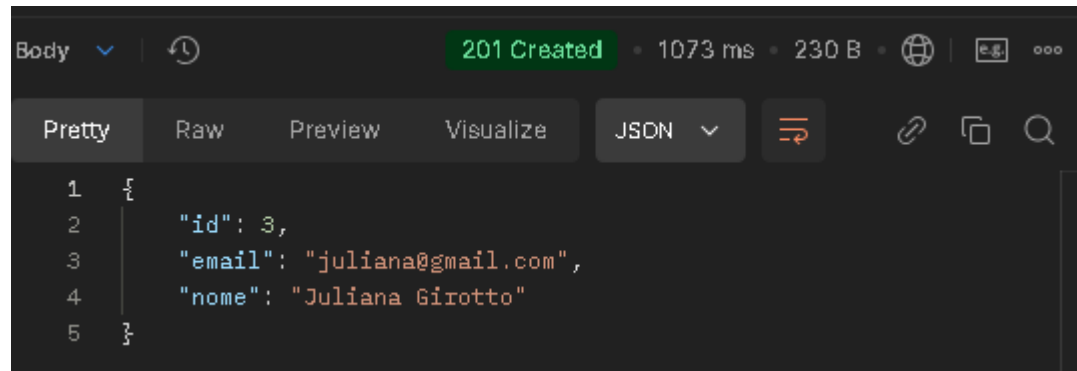
3. Consumo da API no Postman:

URL: <http://localhost:8000>

ENDPOINT	MÉTODO HTTP	URL	AÇÃO
Aluno	POST	/alunos	Cadastra um aluno
Nota	POST	/notas	Cadastra uma nota
Nota	GET	/alunos/notas	Busca notas
Nota	GET	/alunos/notas/{tipo}	Busca nota por tipo

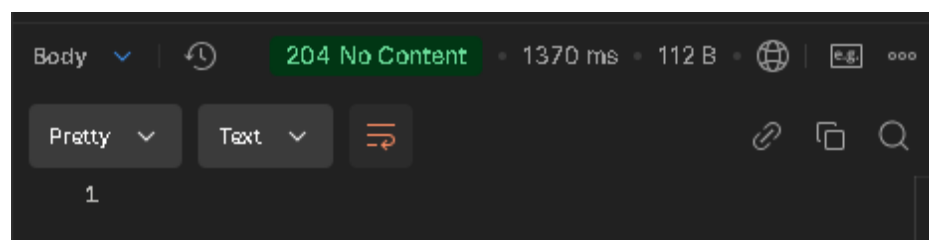
- Cadastrar um aluno:

```
1 curl --location 'http://localhost:8080/alunos' \  
2 --header 'Content-Type: application/json' \  
3 --data-raw '{  
4     "email": "juliana@gmail.com",  
5     "senha": "123456",  
6     "nome": "Juliana Girotto"  
7 }'
```



- Cadastrar uma nota:

```
1 curl --location 'http://localhost:8080/notas' \  
2 --header 'Content-Type: application/json' \  
3 --data '{  
4     "idAluno": 3,  
5     "tipo": "P1",  
6     "valor": 9.0  
7 }'
```



- Consultar notas:

```
1 curl --location --request GET 'http://localhost:8080/  
alunos/notas' \  
2 --header 'Content-Type: application/json' \  
3 --data-raw '{  
4     "email": "juliana@gmail.com",  
5     "senha": "123456"  
6 }'
```

```
Body 200 OK • 41 ms • 280 B
Pretty JSON
1 {
2   "email": "juliana@gmail.com",
3   "nome": "Juliana Girotto",
4   "notas": [
5     {
6       "tipo": "P1",
7       "valor": 9.0
8     },
9     {
10      "tipo": "P2",
11      "valor": 8.0
12    }
13  ]
14 }
```

- Consultar notas por tipo:

```
1 curl --location --request GET 'http://localhost:8080/
  alunos/notas/P2' \
2 --header 'Content-Type: application/json' \
3 --data-raw '{
4   "email": "juliana@gmail.com",
5   "senha": "123456"
6 }'
```

```
Body 200 OK • 436 ms • 254 B
Pretty JSON
1 {
2   "email": "juliana@gmail.com",
3   "nome": "Juliana Girotto",
4   "notas": [
5     {
6       "tipo": "P2",
7       "valor": 8.0
8     }
9   ]
10 }
```

4. Em `'application.yml'`, altere o item `ddl-auto`:

```
ddl-auto: none
```

5. Agora, no MySQL Workbench, execute a seguinte query para testar se a API foi implementada pensando em um design seguro:

```
ALTER TABLE notas DROP COLUMN tipo;
```

6. Execute a aplicação novamente com as alterações realizadas. Volte ao passo 4 e realize o método *GET* de buscar notas por tipo e utilize o mesmo corpo de requisição utilizado anteriormente. A resposta será a seguinte:

```
1 {
2   "timestamp": "2024-11-22T21:55:37.414+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "trace": "org.springframework.dao.InvalidDataAccessResourceUsageException: JDBC exception executing SQL
[select n1_0.id,n1_0.id_aluno,n1_0.tipo,n1_0.valor from notas n1_0 where n1_0.id_aluno is null]
[Unknown column 'n1_0.tipo' in 'field list'] [n/a]; SQL [n/a]\r\n\tat org.springframework.orm.jpa.
vendor.HibernateJpaDialect.convertHibernateAccessException(HibernateJpaDialect.java:277)\r\n\tat org.
springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.
java:241)\r\n\tat org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.
translateExceptionIfPossible(AbstractEntityManagerFactoryBean.java:550)\r\n\tat org.springframework.
dao.support.ChainedPersistenceExceptionTranslator.translateExceptionIfPossible
(ChainedPersistenceExceptionTranslator.java:61)\r\n\tat org.springframework.dao.support.
DataAccessUtils.translateIfNecessary(DataAccessUtils.java:335)\r\n\tat org.springframework.dao.support.
PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.java:160)
\r\n\tat org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:184)\r\n\tat org.springframework.data.jpa.repository.support.
CrudMethodMetadataPostProcessor$CrudMethodMetadataPopulatingMethodInterceptor.invoke
```

7. Para correção da exceção com dados sensíveis do banco de dados, recomenda-se a implementação de uma classe global para tratamento de exceções. Portanto, no arquivo `'src/main/java/com/juliana/appalunos/application/exception/GlobalExceptionHandler.java'` e desfaça o comentário que contém o tratamento previamente realizado da exceção em estudo: *DataAccessException*.

8. Execute a aplicação novamente com as alterações realizadas. Volte ao passo 4 e realize o método *GET* de buscar notas por tipo e utilize o mesmo corpo de requisição utilizado anteriormente. A resposta será a seguinte:



The screenshot shows a web browser's developer console with the 'Body' tab selected. At the top right, a red banner displays '500 Internal Server Error'. Below the banner, there are tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize', with 'Pretty' being the active tab. The JSON response is displayed in a dark-themed editor with line numbers 1 through 7. The response is an object with the following properties: 'path' (a URL), 'method' (GET), 'status' (500), 'statusText' (Internal Server Error), and 'message' (a Portuguese error message).

```
1 {
2   "path": "/alunos/notas/P2",
3   "method": "GET",
4   "status": 500,
5   "statusText": "Internal Server Error",
6   "message": "Erro no banco de dados. Tente novamente mais tarde."
7 }
```