

Hackathon Stefanini 2019

(Número 5 - Aluno | Mochila)

César Lucas Júnior

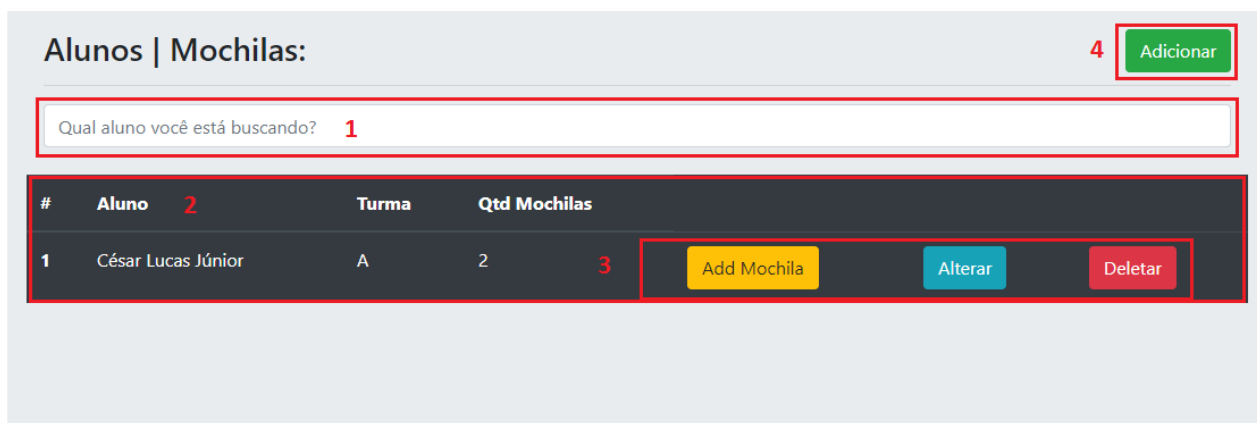
Front-end:

Seguindo o padrão inicial, criei uma aplicação que consumisse os endpoints disponibilizados pelo backend. Provi uma tela inicial(/home) com um menu de acesso a tela de consulta da entidade pai (Aluno).



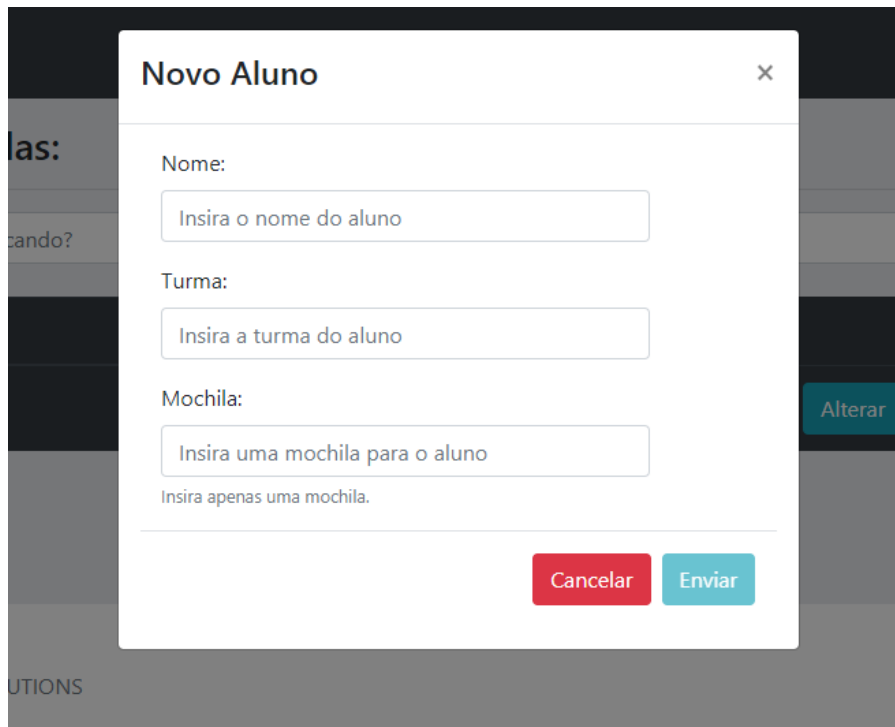
Na tela de consulta, disponibilizamos:

1. Um filtro que permite a busca pelo nome do aluno, mas que não é obrigatório.
2. O resultado da consulta – tabela com o nome do aluno, turma cadastrada e a quantidade de filhos (mochila) vinculados.
3. Em cada registro, disponibilizei três botões: uma para adicionar uma mochila (seguindo as restrições), um para alterá-lo e outro para deletá-lo.
4. Um botão para adicionar um aluno (pai) e apenas uma mochila (filha) conforme solicitado.



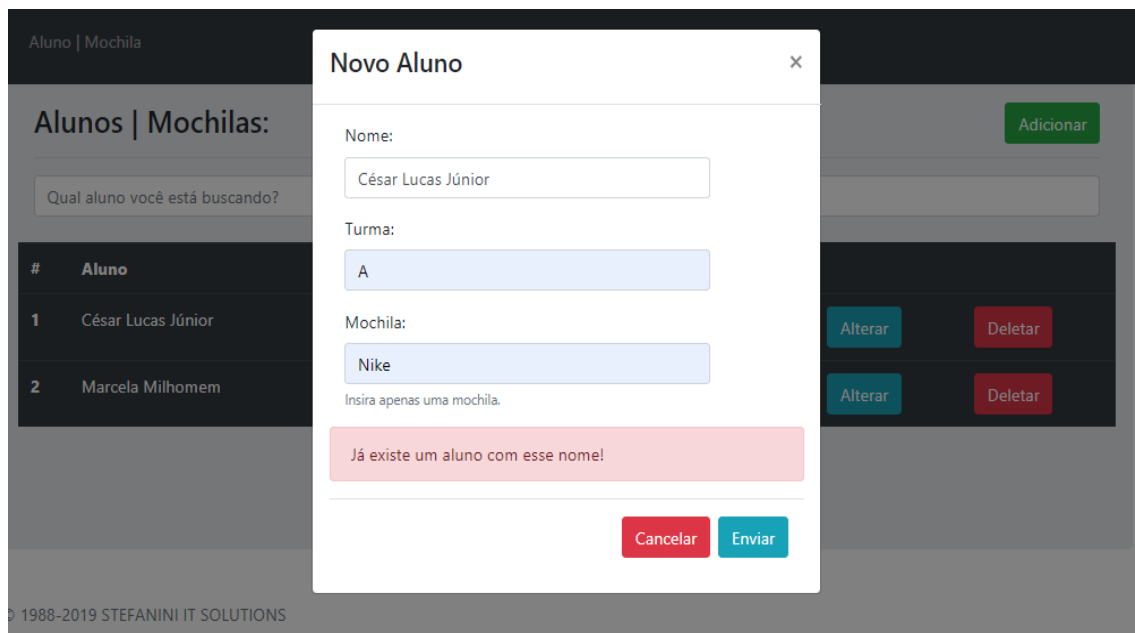
Na tela para registrar um novo aluno nos preocupamos em só se poder submeter quando todos os requisitos mínimos forem obedecidos:

- ✓ Campo nome e mochila não podem ter mais de 20 caracteres.
- ✓ O campo nome não pode ser repetido no registro.



A screenshot of a web application showing a modal window titled "Novo Aluno" (New Student). The modal has a close button (X) in the top right corner. It contains three input fields: "Nome:" (Name), "Turma:" (Class), and "Mochila:" (Backpack). Each field has a placeholder text: "Insira o nome do aluno", "Insira a turma do aluno", and "Insira uma mochila para o aluno" respectively. Below the "Mochila:" field, there is a note: "Insira apenas uma mochila." (Insert only one backpack). At the bottom of the modal, there are two buttons: "Cancelar" (Cancel) in red and "Enviar" (Send) in teal. The background of the application is dark gray, and there are some partially visible elements like "las:", "cando?", "Alterar", and "UTIONS".

Caso um desses requisitos não sejam estabelecidos uma mensagem de erro aparecerá na modal e o botão “Enviar” será bloqueado.



A screenshot of the same web application showing the "Novo Aluno" modal window. The modal is now filled with data: "Nome:" is "César Lucas Júnior", "Turma:" is "A", and "Mochila:" is "Nike". Below the "Mochila:" field, the note "Insira apenas uma mochila." is still present. At the bottom of the modal, there is a red error message box that says "Já existe um aluno com esse nome!" (There is already a student with this name!). The "Enviar" button is now disabled (grayed out). The background of the application is dark gray, and there are some partially visible elements like "Aluno | Mochila", "Alunos | Mochilas:", "Qual aluno você está buscando?", a table with student data, "Adicionar", "Alterar", "Deletar", and "© 1988-2019 STEFANINI IT SOLUTIONS".

#	Aluno
1	César Lucas Júnior
2	Marcela Milhomem

as:

ando?

Alterar

TIONS

Novo Aluno

×

Nome:

Insira o nome do aluno

Turma:

A

Mochila:

Insira uma mochila para o aluno

Insira apenas uma mochila.

Por favor, preencha o nome do aluno

Por favor, preencha o nome da mochila

Cancelar

Enviar

Na tela para adicionar uma mochila permite a adição do filho sem a alteração de qualquer outro dado.

Incluir Mochila

×

Nome:

César Lucas Júnior

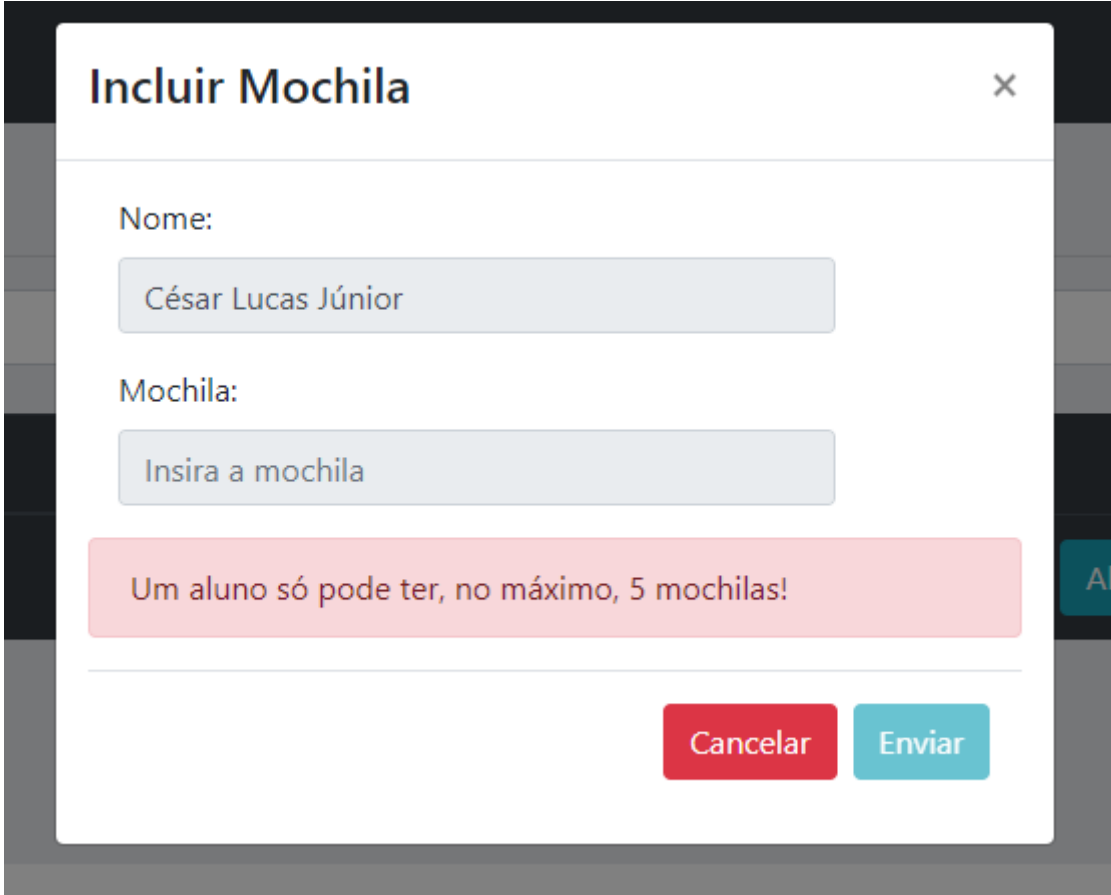
Mochila:

Insira a mochila

Cancelar

Enviar

Caso um aluno já tenha 5 mochilas associadas mandamos uma mensagem de erro e bloqueamos o botão “Enviar”.



The image shows a modal window titled "Incluir Mochila" with a close button (X) in the top right corner. It contains two input fields: "Nome:" with the value "César Lucas Júnior" and "Mochila:" with the placeholder text "Insira a mochila". Below these fields is a red error message box that reads "Um aluno só pode ter, no máximo, 5 mochilas!". At the bottom right, there are two buttons: "Cancelar" (red) and "Enviar" (teal).

Incluir Mochila ✕

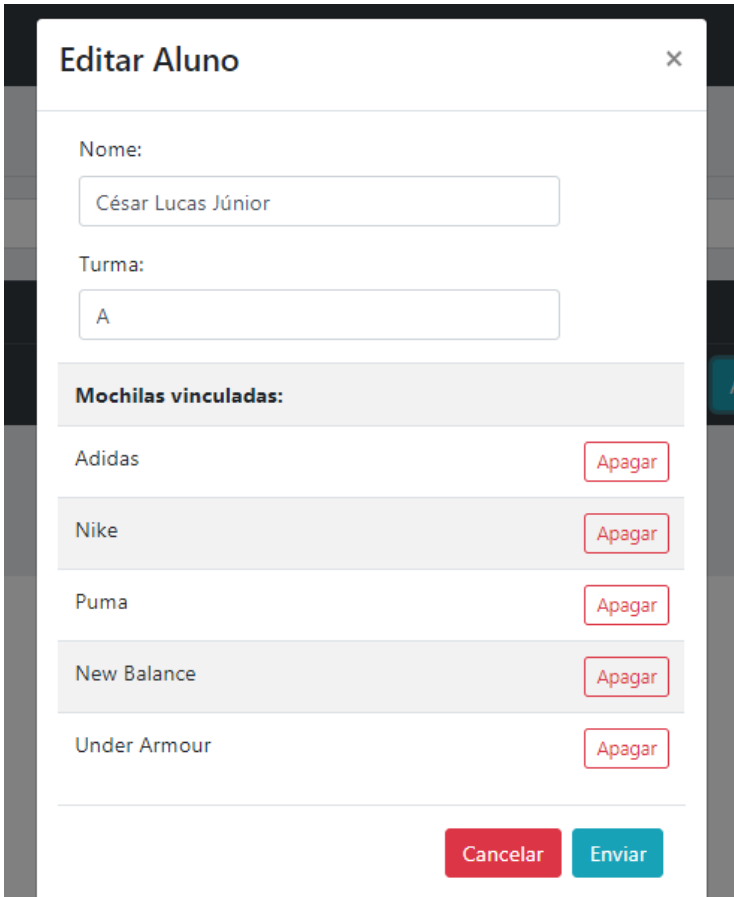
Nome:
César Lucas Júnior

Mochila:
Insira a mochila

Um aluno só pode ter, no máximo, 5 mochilas!

Cancelar Enviar

Na tela Editar Aluno permite-se a alteração no Nome e da Turma assim como a deleção individual das mochilas vinculadas.



The image shows a modal window titled "Editar Aluno" with a close button (X) in the top right corner. It contains two input fields: "Nome:" with the value "César Lucas Júnior" and "Turma:" with the value "A". Below these fields is a section titled "Mochilas vinculadas:" which lists five backpacks: Adidas, Nike, Puma, New Balance, and Under Armour. Each backpack name is followed by a red "Apagar" button. At the bottom right, there are two buttons: "Cancelar" (red) and "Enviar" (teal).

Editar Aluno ✕

Nome:
César Lucas Júnior

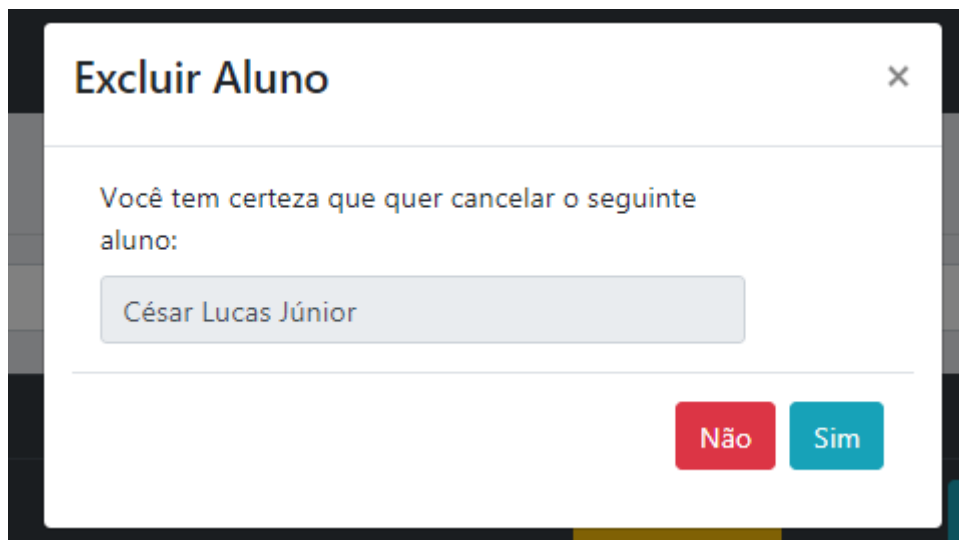
Turma:
A

Mochilas vinculadas:

Adidas	Apagar
Nike	Apagar
Puma	Apagar
New Balance	Apagar
Under Armour	Apagar

Cancelar Enviar

Na deleção do registro aparecerá uma tela de confirmação:



Back-end:

Escolhi o projeto 5 para desenvolver a aplicação com Spring MVC e arquitetura Restful. Desse modo, disponibilizei os endpoints cabíveis conforme requisitos na <http://localhost:8080/alunos> variando conforme métodos buscados.

Utilizei o banco H2 para persistir em memória os dados necessários. Incluí, na inicialização da aplicação um Aluno com duas Mochilas para facilitar visualização no front-end conforme imagem a seguir:

```
public void run(String... args) throws Exception {

    Mochila mochila1 = new Mochila("Adidas");
    Mochila mochila2 = new Mochila("Nike");
    Aluno aluno = new Aluno("César Lucas Júnior", "A");

    aluno.setMochilas(Arrays.asList(mochila1, mochila2));

    mochila1.setAluno(aluno);
    mochila2.setAluno(aluno);

    // Persistindo no banco de dados:

    System.out.println("Persistindo dados...");

    alunoRepository.saveAll(Arrays.asList(aluno));
    mochilaRepository.saveAll(Arrays.asList(mochila1, mochila2));

}
```

Obedecendo a arquitetura Resource-Service-Repository, disponibilizei os seguintes métodos da Resource juntamente com seus respectivos métodos services e, consequentemente, na repository.

```
public class AlunoResource {  
  
    private AlunoService alunoService;   
  
    public ResponseEntity<Aluno> insert(@RequestBody Aluno aluno) {  
  
    public ResponseEntity<List<Aluno>> findAll() {  
  
    public ResponseEntity<List<Aluno>> findByName(@PathVariable String nome) {  
  
    public ResponseEntity<Aluno> update(@RequestBody Aluno aluno) {  
  
    public ResponseEntity<Aluno> delete(@PathVariable Long id) {  
  
}
```

Vale ressaltar que busquei tratar todos os possíveis fluxos de exceção com classes personalizadas como:

- ✓ DataIntegrityException,
- ✓ InstanceNumberExceed,
- ✓ ObjectNotFoundException

Todas essas classes foram utilizadas pela RestExceptionHandler para personalizar os erros assim como ser captura e tratada pelo front quando necessário.

Quanto aos testes unitários, realizei as validações pela service assim como pela repository e testes de integração com o AlunoResource. Desse modo, conseguimos validar os métodos em suas menores partes assim como o fluxo de trabalho até o endpoint.

Garanti, conforme solicitado a exclusão dos registros da entidade filha quando a entidade pai fosse excluída mais ainda, evitando ciclos infinitos entre as classes através do @JsonIgnore.