Projeto final – Parte 2

Evidências de testes e descrição técnica do projeto.

André Alecsander A. Lopes – 18/11/2021

A aplicação a seguir trata-se de uma solução para um sistema de cadastro de cursos da cast. Inicia-se o projeto utilizando o Visual Studio 2019 para criar uma aplicação Web API do ASP.NET Core, utilizando Net 5.0. Primeiramente, criamos as Models para modelar o banco de dados através do método CodeFirst. Definimos os atributos das classes, que se tornaram as tabelas do banco de dados a serem populadas através do uso do Entity Framework Core, e validações apropriadas para o bom funcionamento do sistema.

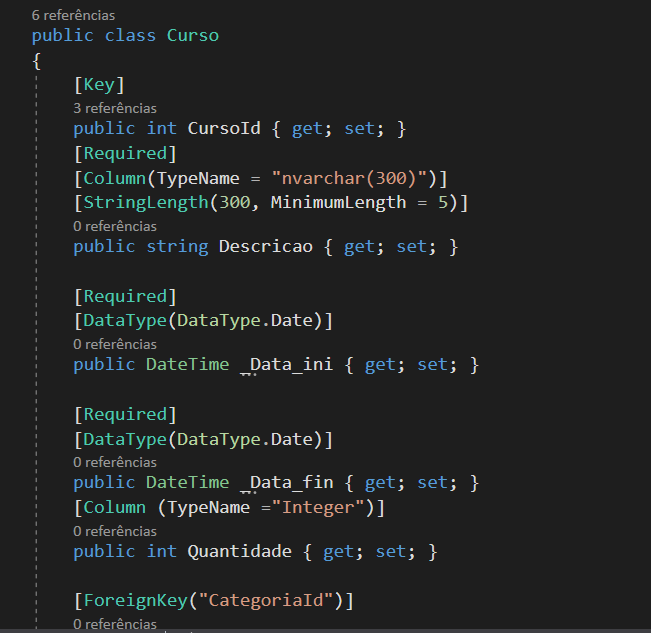


Figura 1 – Definindo a classe curso para popular o banco através do Entity Framework Core.

Após isso, criamos uma nova pasta no projeto de nome “Data” para podermos definir os DbSet, utilizando as models que foram criadas em forma de classes, para realizar a migração da classe para o banco de dados. Nisso, providenciamos o serviço de banco de dados a ser utilizado e a string de conexão para deste banco de dados.

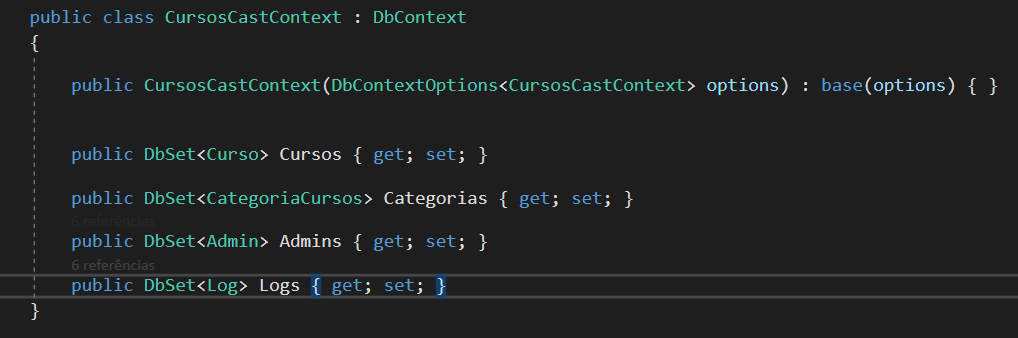


Figura 2 – Dbset para realização de migração de banco de dados.

Em seguida, vamos no Startup.cs para aplicarmos injeção de dependência através do ASP.NET Core, salvando a string de conexão no appsettings.json.

Desta forma, o sistema já está pronto para realização das migrações do programa para o banco de dados, utilizando os comandos “add-migration ‘<Mensagem>’” e update-database. É neste momento em que o banco de dados é definitivamente criado.

Especificamente neste projeto, foi utilizado o serviço de banco de dados Microsoft SQL Server Management Studio 18 para consulta e manipulação dos dados contidos.

À partir daí, seguimos para a necessidade de controle e manipulação dos dados na nossa API. Para isto, foram adicionados controllers no projeto a fim de realizar os métodos PUT, POST, GET e DELETE, através de adição de novo item Scaffoldin para controlar API com ações, usando Entity FrameWork.

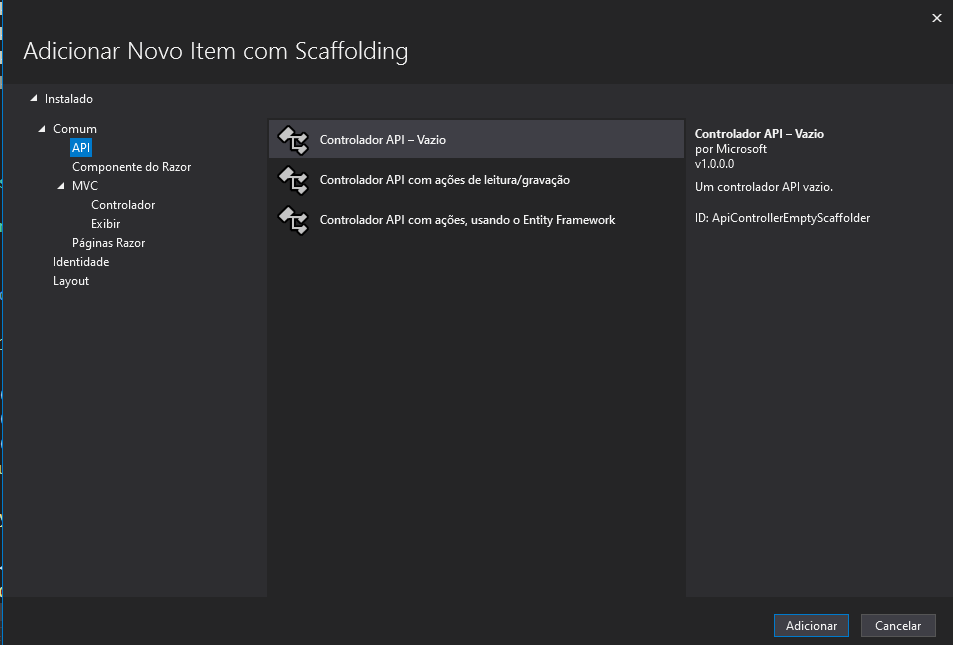


Figura 3 – Adicionar controlador na pasta controllers para criar uma API com ações.

Assim, nossa API está pronta para ser “consumida” pelo nosso frontEnd.

Partindo agora para o Visual Studio Code, criamos uma pasta qualquer e, dentro dela, chamamos o terminal do VisualCode e digitamos “ng new <nome\_do\_projeto>” para criarmos uma aplicação AngularJS.

Neste projeto, utilizamos o Angular 12 para construção da interface do projeto, usando bootstrap para estilização das páginas.

Assim, criamos as devidas pastas para cada operação, através dos comandos ng g c <nome\_do\_componente>.

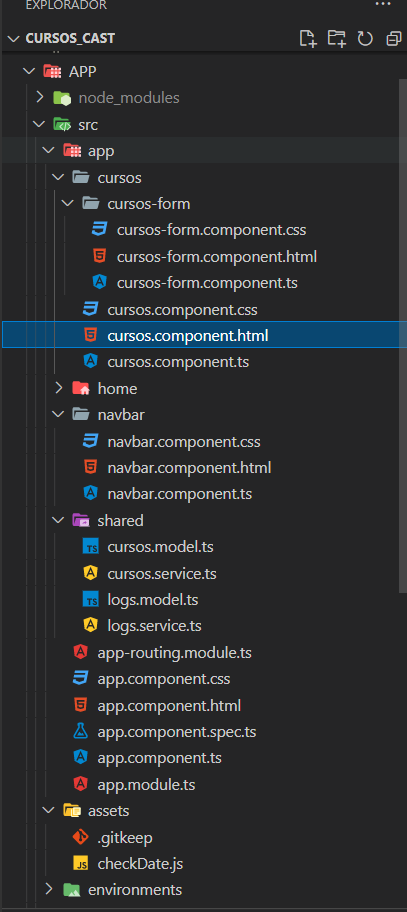


Figura 4 – Componentes do projeto Angular

Nesta etapa, criamos todas as aplicações, através de HTML, CSS, Javascript e TypeScript, do frontEnd, como as validações dos campos requisitados. Inicialmente, temos a homepage, onde teremos uma navbar para acesso aos cursos, utilizando a componente app-routing.module.ts para definir uma rota que conecta os componentes “home” e “navbar”.

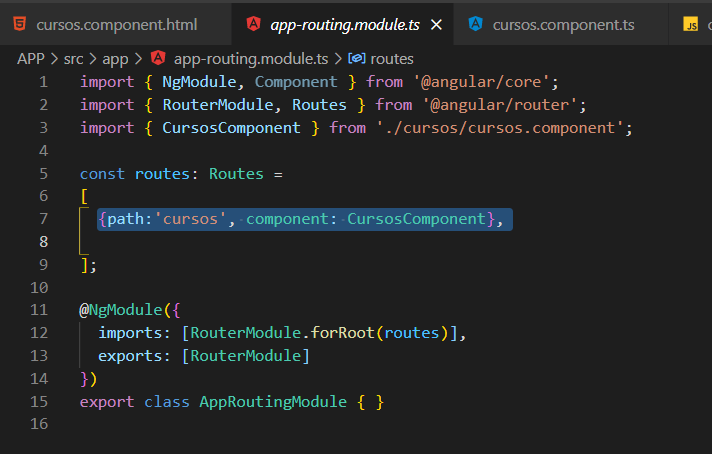


Figura 5 – Definição de rotas para acesso do navbar.

Neste componente angular, definimos as aplicações do nosso projeto, utilizando as componentes “<nome\_da\_componente>.model.ts” e ”<nome\_da\_componente>.services.ts”, além das padrões do projeto.

Utilizamos estas duas componentes em especial para definirmos a conexão do frontEnd para nossa API, através de operações CRUD.

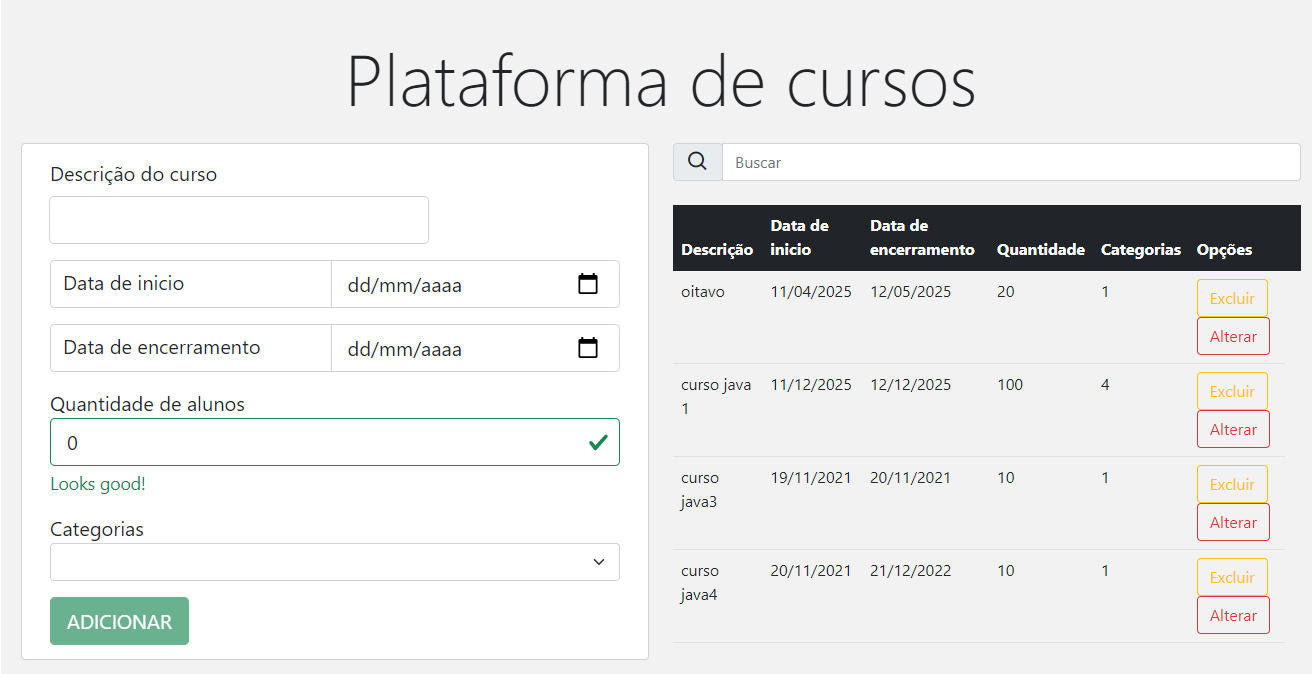


Figura 6 – Página de criação de cursos da cast.

Na qual temos duas aplicações web rodando na mesma página. Sendo um o formulário de cadastro de novos cursos e o outro sendo um formulário de consulta dos cursos já criados.

Evidências de Teste:

1 – Criação de um novo curso:

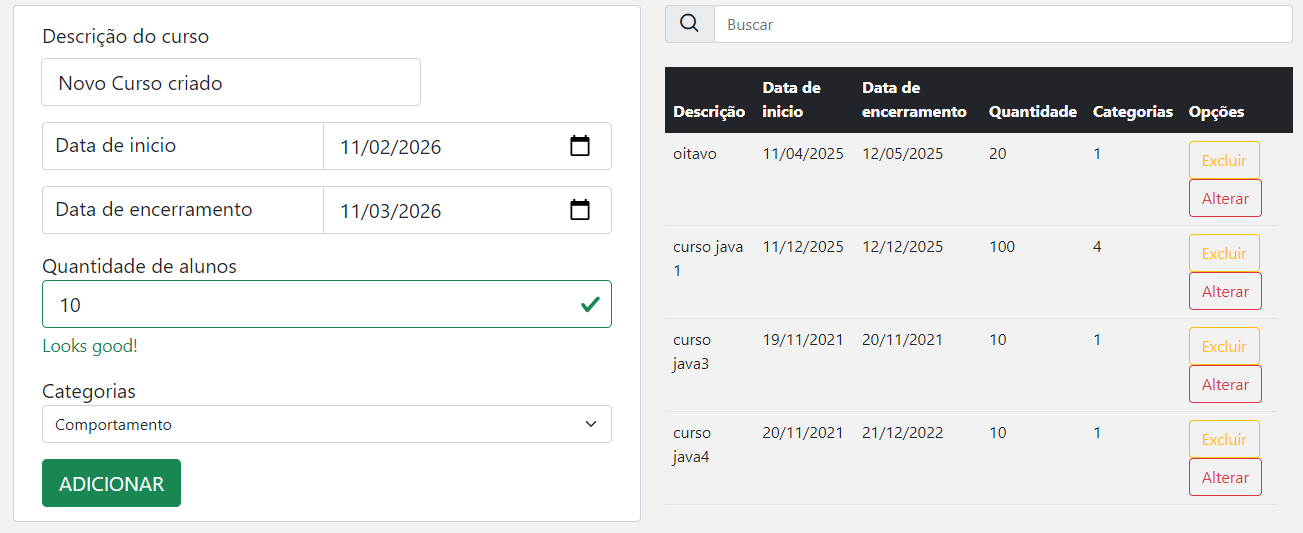


Figura 7 – Criação de curso antes

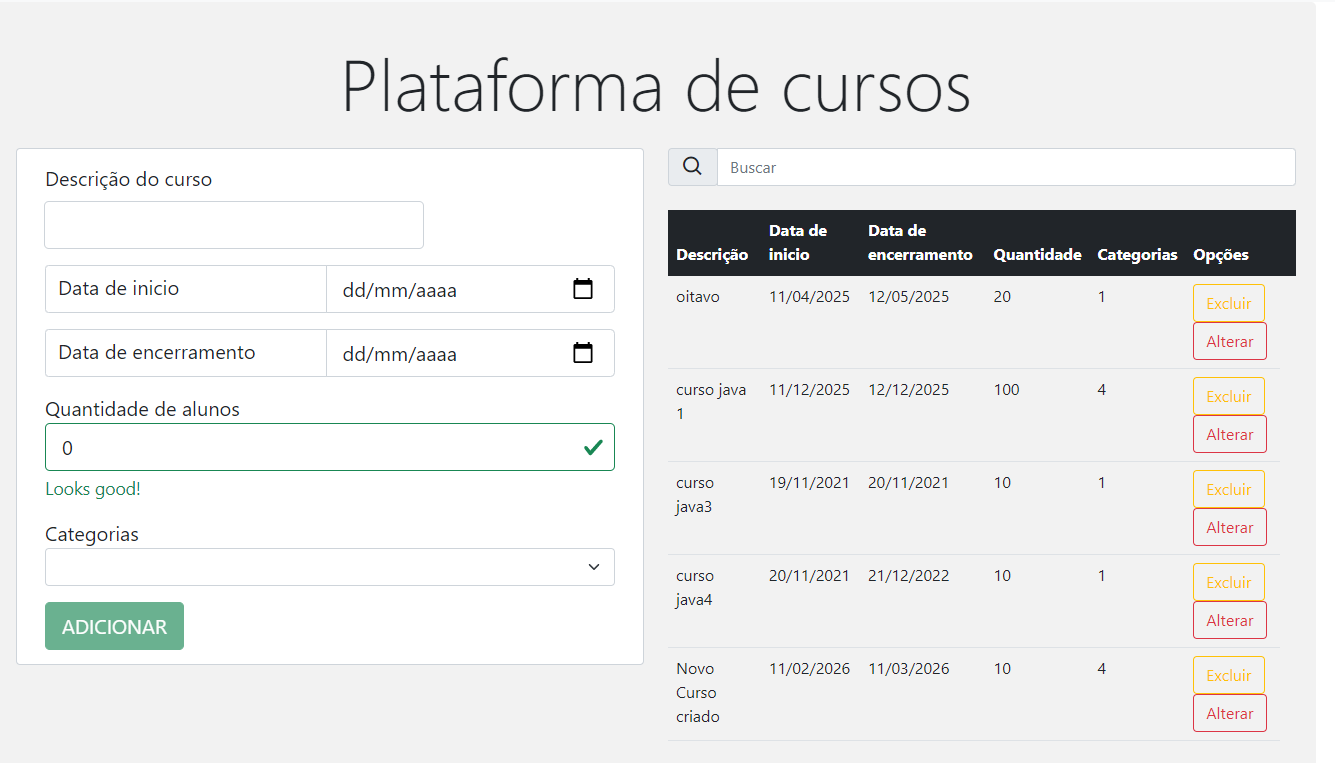


Figura 8 – Criação de curso depois

Como pode ser observado acima, em caso de campos sem preenchimento, o botão de adicionar um novo curso não é habilitado, sendo impossível adicionar um novo curso.

2 – Tentativa de cadastramento num período já cadastrado com outro curso:

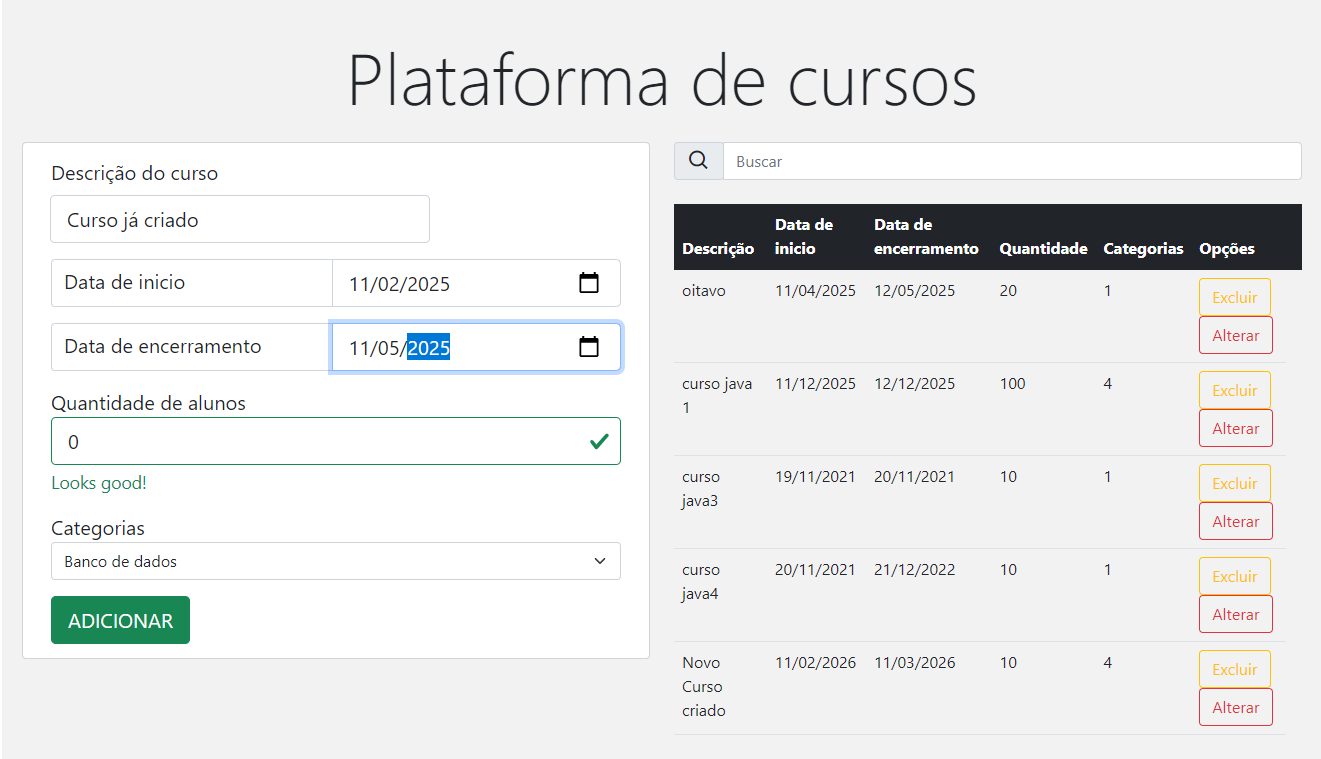


Figura 9 – Realização do teste 2, antes.

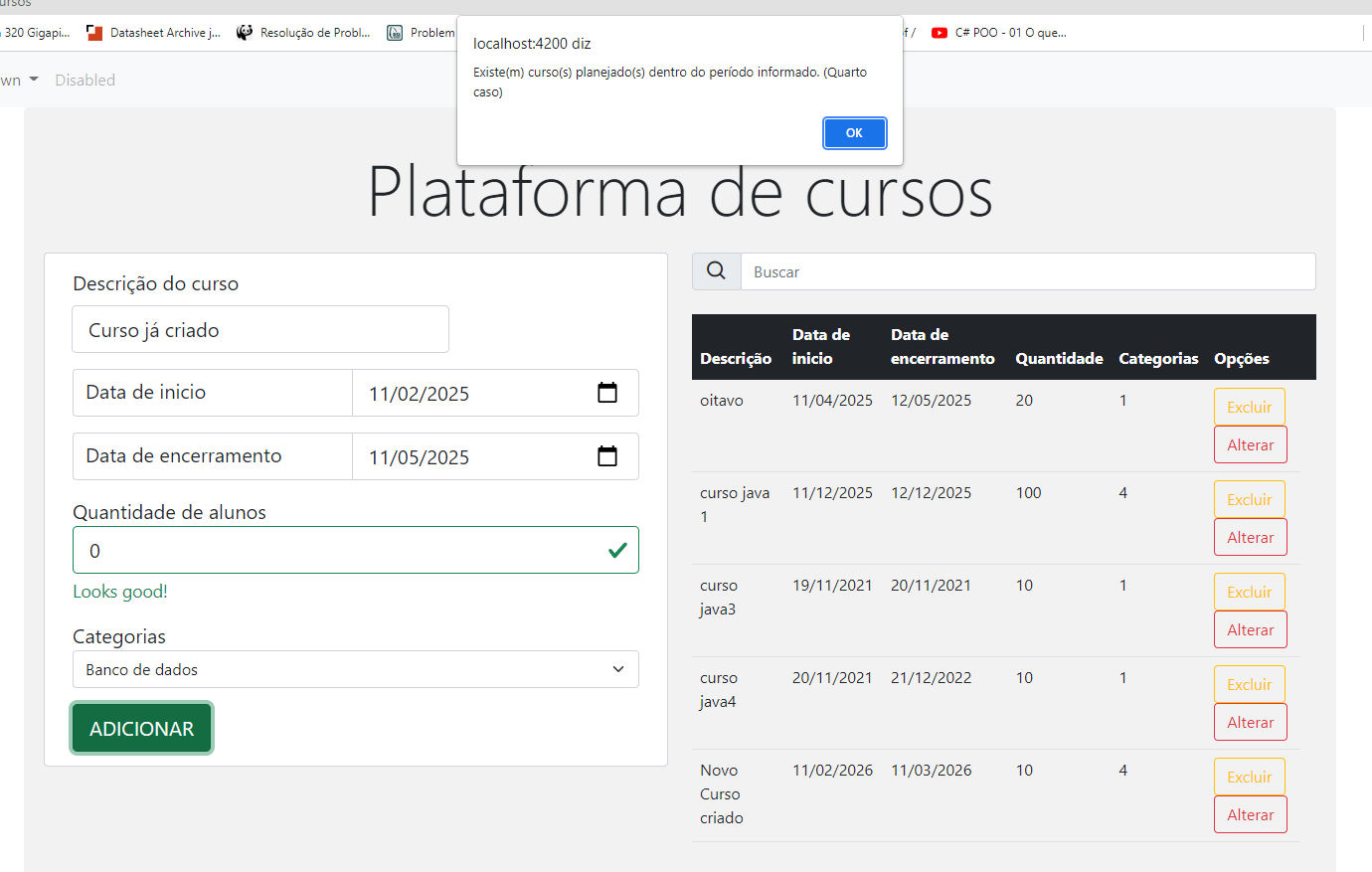


Figura 10 – Realização do teste 2, depois.

3 – Curso com data inválida:

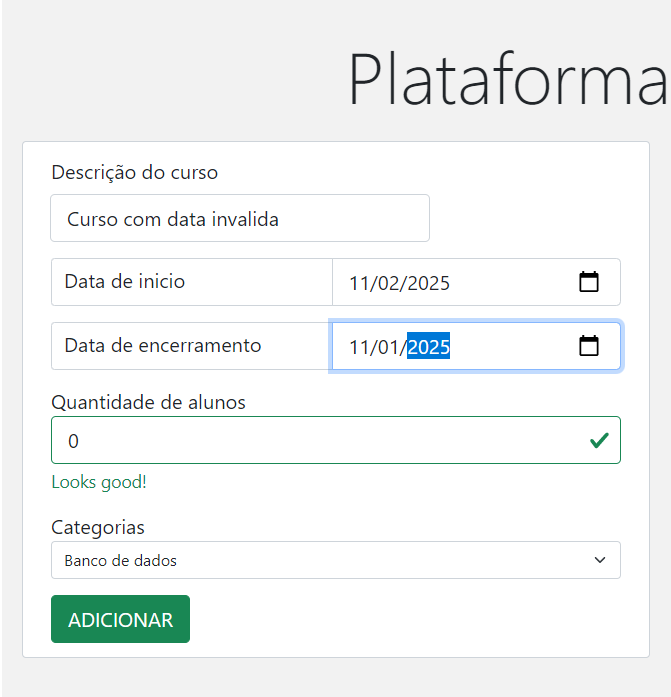


Figura 10 – Teste da terceira condição, antes

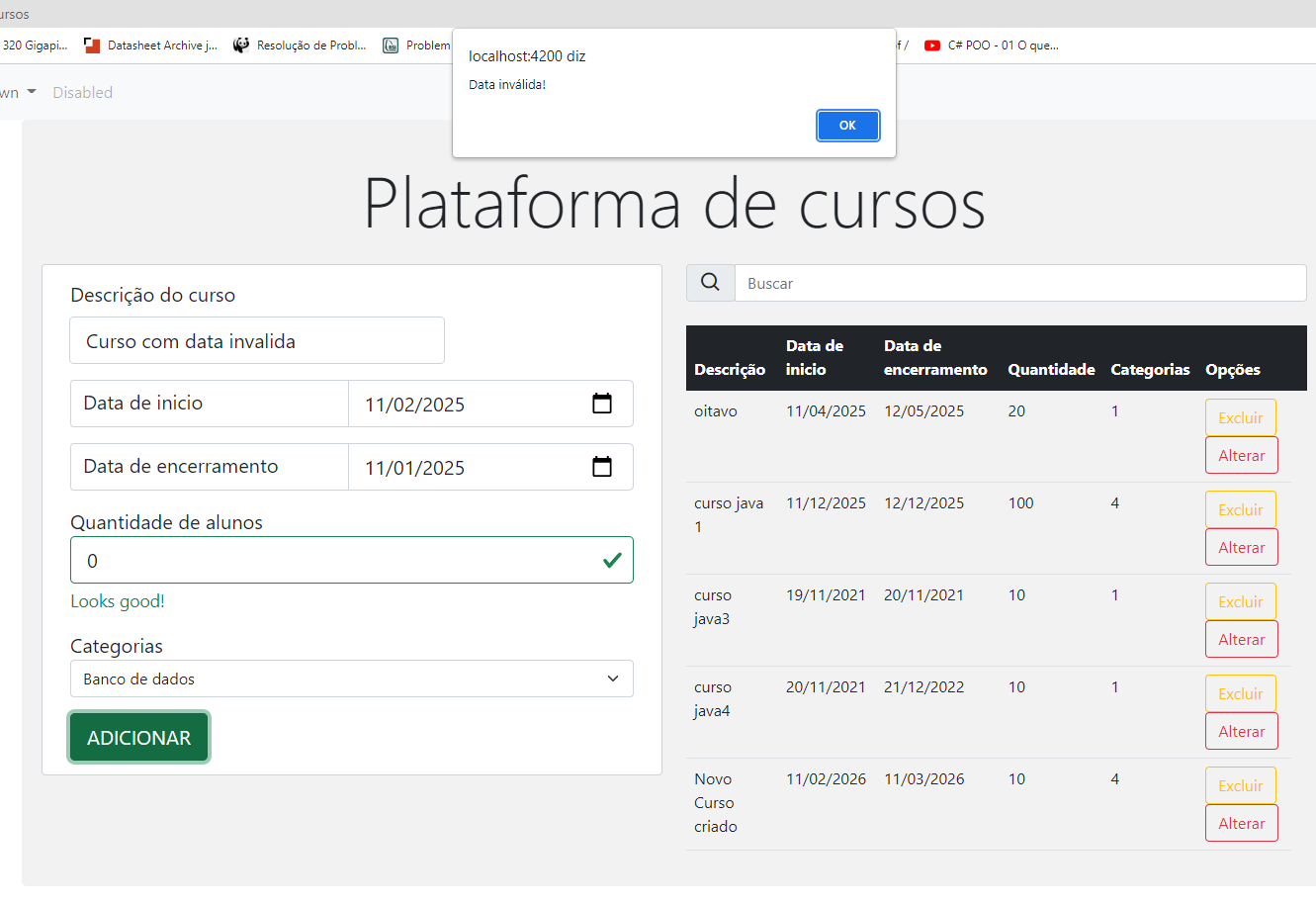


Figura 11 – Teste da terceira condição, depois.

4 – Mensagem de alerta em caso de curso já criado.

Neste teste, partimos do pressuposto de que um curso repetido terá a mesma descrição que outro curso.

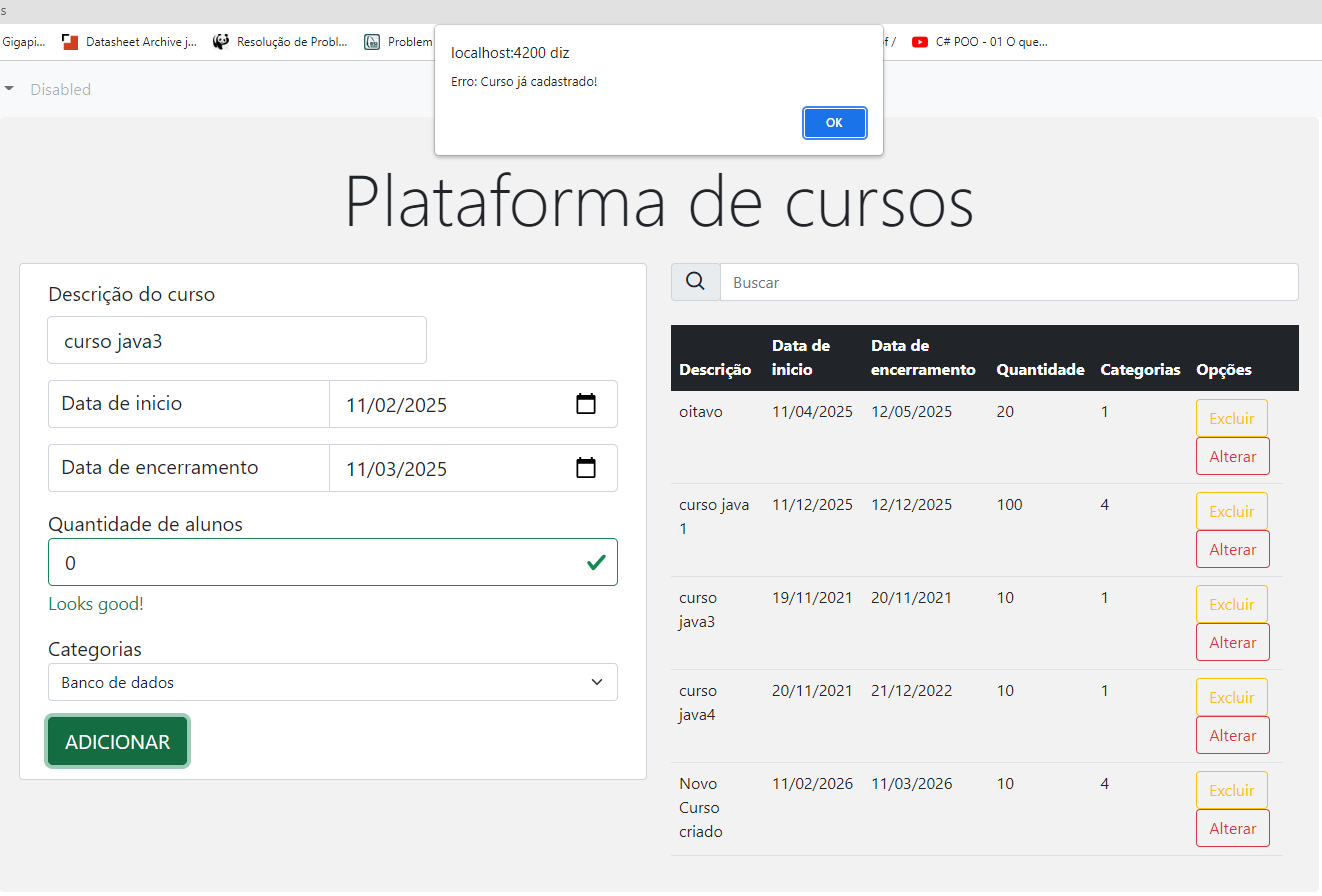


Figura 12 – Tentativa de cadastramento de um curso repetido.

5 – Pesquisa por nome e período do curso:

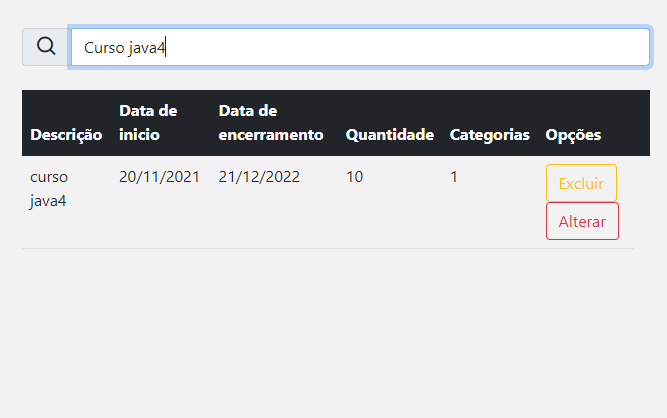


Figura 13 – Realização de pesquisa pela descrição do curso

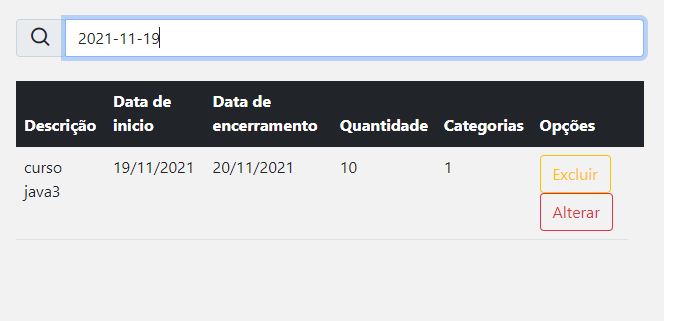


Figura 14 – Realização de pesquisa pelo período do curso

No caso das condições listadas abaixo, não foi possível implementar uma solução nesta fase de desenvolvimento.

* Não deve permitir a exclusão de cursos já realizados;
* Deve registrar em uma tabela de log a data da inclusão, data da última atualização e usuário responsável;