

Documentação

Roman

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Alameda Barão de Limeira, 539 – São Paulo/SP |  |  |  | sp.senai.br |

Sumário

1. [Resumo 2](#_Toc3470412)

[Objetivos](#_Toc3470413)

1. [Descrição do projeto 2](#_Toc3470414)

[Resumo do projeto](#_Toc3470415)

1. [Modelagem de Software 3](#_Toc3470416)

[Modelo Lógico](#_Toc3470417)

[Modelo Físico](#_Toc3470418)

[Modelo Conceitual](#_Toc3470419)

[Trello](#_Toc3470420)

[GitHub](#_Toc3470421)

1. [Funcionalidades 8](#_Toc3470422)

[Back-end – Passo a Passo](#_Toc3470423)

[Web](#_Toc3470424)

[Mobile](#_Toc3470425)

1. [Protótipos 11](#_Toc3470426)

[Web](#_Toc3470427)

[Mobile](#_Toc3470428)

1. [Front-End 17](#_Toc3470429)
2. [Mobile 21](#_Toc3470430)
3. [Arquitetura do Projeto 22](#_Toc3470431)
4. [Referências 23](#_Toc3470432)

[Links](#_Toc3470433)

[Livros](#_Toc3470434)

Resumo

Objetivos

Este documento tem por finalidade registrar o final do sistema Roman Mobile, mostrando suas especificações técnicas e descrevendo seus recursos e capacidades.

Descrição do projeto

O projeto se refere à um sistema mobile para os professores e alunos do Senai de Informática, onde os alunos poderão enviar dicas ou sugestões para temas de aulas, projetos e atividades..

Resumo do projeto

O projeto foi realizado por meio de metodologias ágeis, como Scrum e foi separado nas seguintes fases: Design Thinking, Banco de dados, Back-end e Mobile. Cada fase com suas especificações e objetivos.

Modelagem de Software

Modelo Lógico

Chaves primárias e estrangeiras de acordo com as entidades/tabelas apresentadas.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Projetos | |  |
| ID | NOME | ID\_TEMA | ID\_USUARIO |
| 1 | Controle de Estoque | 1 | 3 |
| 2 | Listagem de Personagem | 2 | 2 |

|  |  |
| --- | --- |
| Equipes | |
| ID | NOME |
| 1 | Desenvolvimento |
| 2 | Redes |
| 3 | Multimídia |

|  |  |
| --- | --- |
| Tipo\_Usuario | |
| ID | NOME |
| 1 | Administrador |
| 2 | Professor |

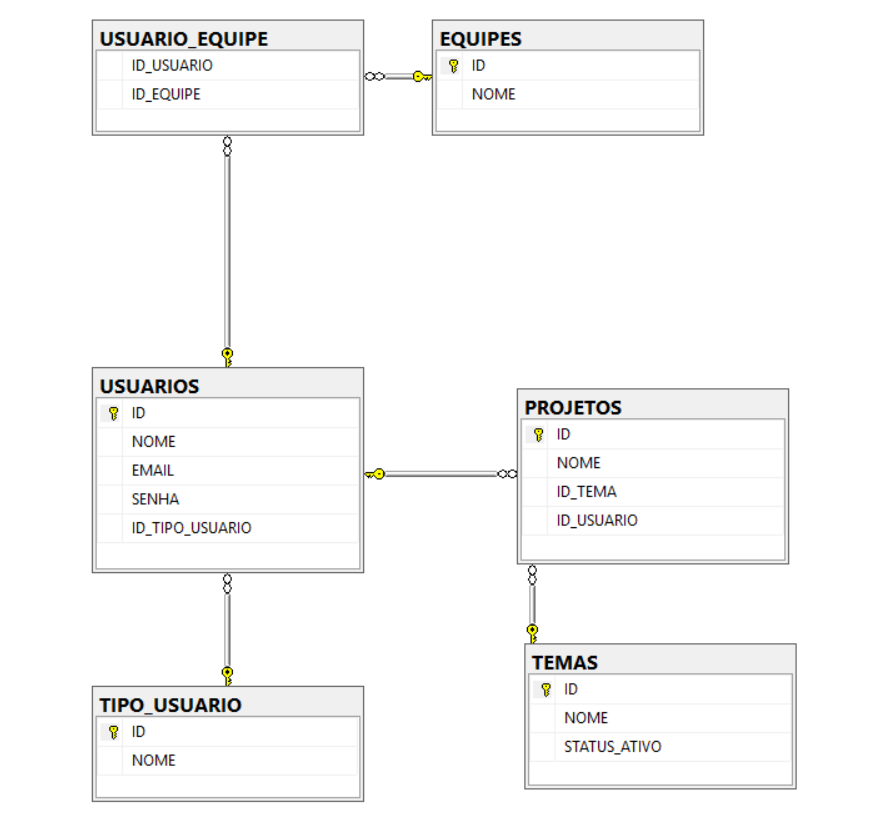
|  |  |  |
| --- | --- | --- |
| Temas | | |
| ID | NOME | STATUS ATIVO |
| 1 | Gestão | Ativo |
| 2 | HQ's | Ativo |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Usuarios | | | | |
| ID | NOME | EMAIL | SENHA | ID\_TIPO\_USUARIO |
| 1 | Administrador | adm@adm | 123456 | 1 |
| 2 | Helena | helena@helena | 123456 | 2 |
| 3 | Fernando | fernando@fernando | 123456 | 2 |

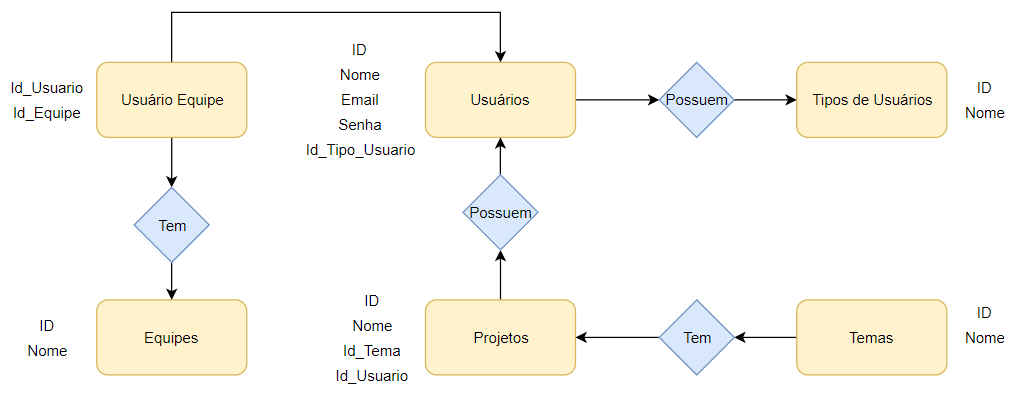
|  |  |
| --- | --- |
| Usuario\_Equipe | |
| ID\_USUARIO | ID\_EQUIPE |
| 3 | 2 |
| 2 | 2 |

Modelo Físico

Tabelas organizadas da seguinte forma no banco de dados:



Modelo Conceitual



O banco de dados possui usuários que possuem dois tipos: Administradores e Professores. Os Temas, se estiverem sendo usados, serão mostrados como ativo e se não estiverem sendo usados serão mostrados como não ativos. Temas também possuem Projetos onde esses podem ser alterados, criados e vistos pelos usuários. Os professores podem fazer (ou não) parte de uma Equipe que seriam: Desenvolvimento, Redes e Multimídia.

Trello

<https://trello.com/b/cFmvMYOM/senai-desafio-roman-%C3%B4mega>

GitHub

<https://github.com/ArielMn22/senai-roman-desafio-omega>

Funcionalidades

Back-end – Passo a Passo

Para criar o banco de dados da aplicação, basta executar os scripts SQL na seguinte ordem:

1. Script de criação: “DDL\_Roman\_Omega”. Para criar o banco de dados, suas tabelas e para visualiza-las;
2. Script de inserção: “DML\_Roman\_Omega”. Para inserir os dados básicos para o funcionamento do sistema, como os usuários padrões, alguns temas, projetos e os tipos de usuário;

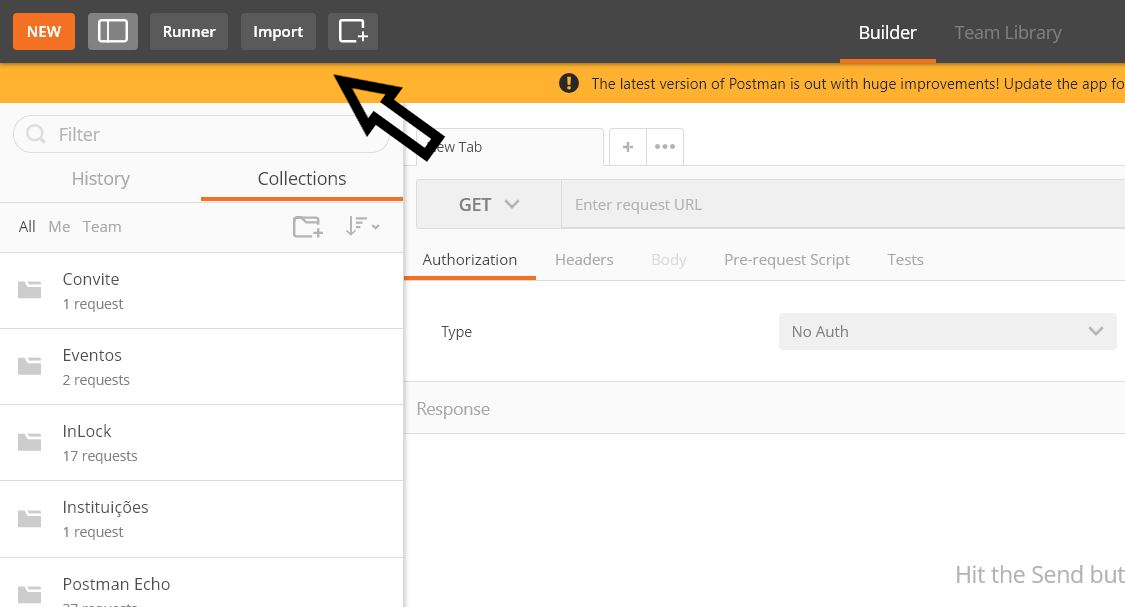
Então, abra a aplicação com Visual Studio, para baixar os seguintes pacotes Nuget:

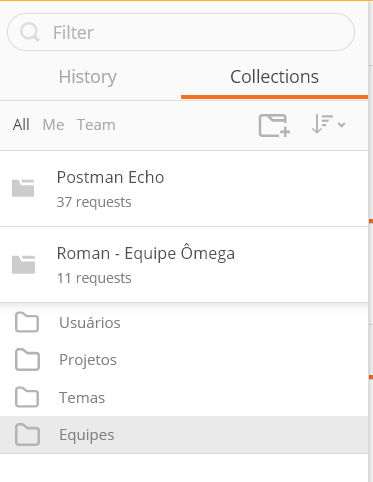
* Microsoft.EntityFrameworkCore.Design;
* Microsoft.EntityFrameworkCore.SqlServer;
* Microsoft.EntityFrameworkCore.Tools;
* Sendgrid;
* Swashbuckle.AspNetCore;
* Swashbuckle.AspNetCore.Swagger;

A seguir, com a solução aberta no Visual Studio, abra o Console do Gerenciador de Pacotes em “Ferramentas > Gerenciador de Pacotes do Nuget > Console do Gerenciador de Pacotes” e execute o seguinte comando:

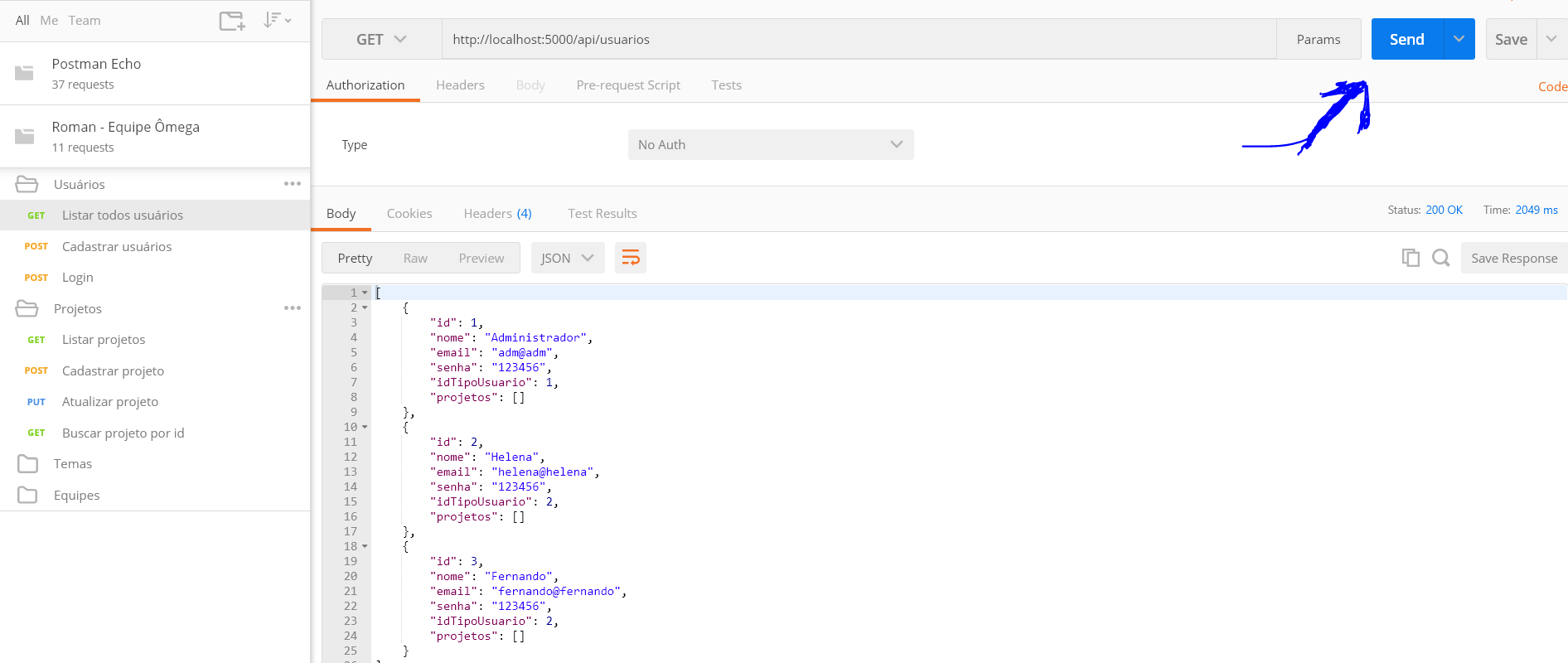
* Scaffold-DbContext "Data Source=.\SqlExpress; Initial Catalog=ROMAN\_OMEGA; user id=sa; pwd=132" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Domains -ContextDir Contexts –Context RomanContext

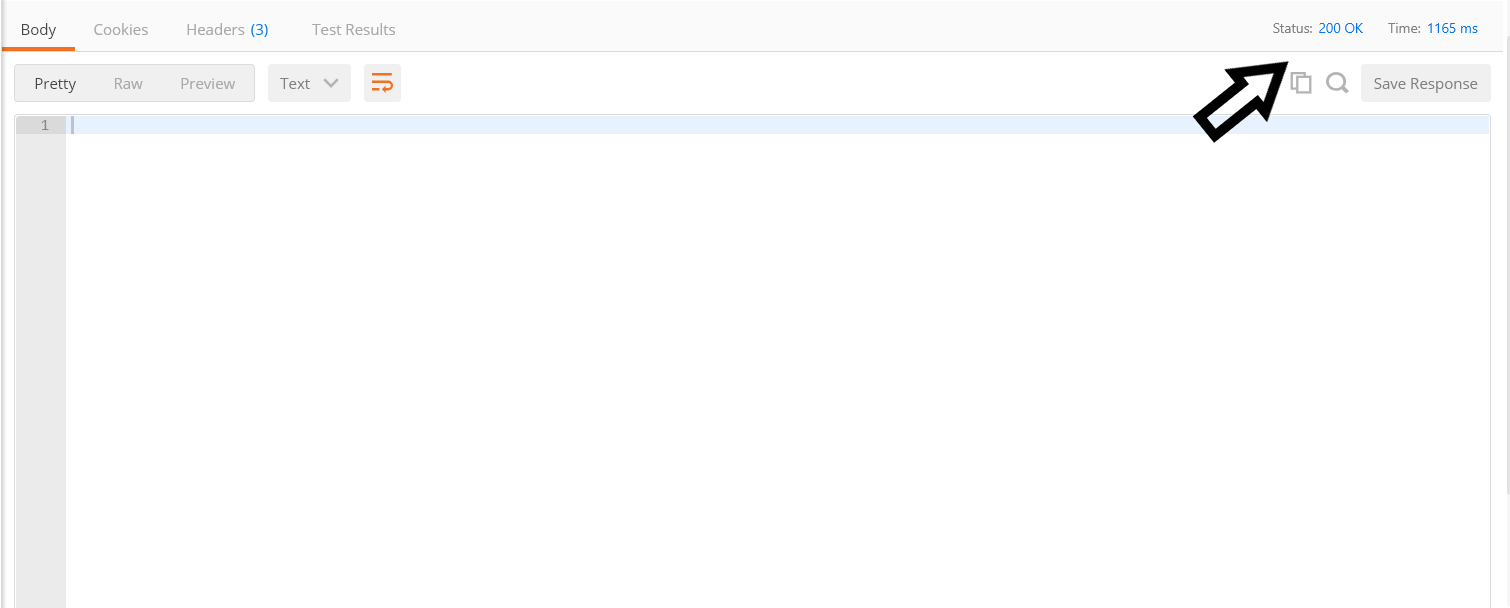
Rode a Solução e então abra o Postman:

* Clique em “Import” para selecionar o arquivo “.json” e importar os links de conexão com as funcionalidades da solução.
* Agora é possível ver a sua Collection Roman com todas as funcionalidades.



Para utilizar um dos métodos, basta selecioná-los, preencher com as informações necessárias e então clicar em “Send”.



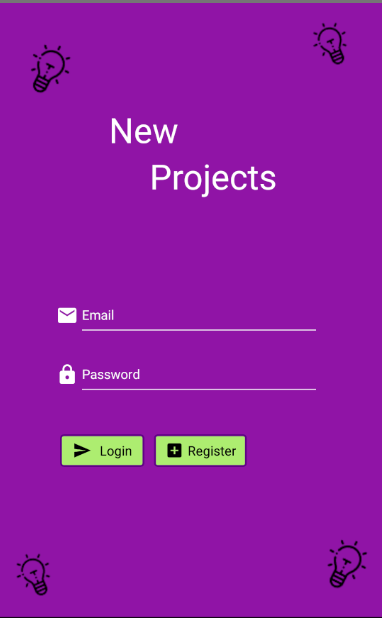
Após clicar em “Send”, tenha o *status code* da requisição:

Web

* Listar todos os Usuários
* Cadastrar usuários novos
* Iniciar uma sessão com seu usuário.
* Listar os projetos
* Cadastrar um Projeto
* Atualizar um projeto
* Listar todos os temas.
* Atualizar um tema.
* Cadastrar um tema.
* Listar Equipe

Mobile – Pagina de Login

Página de login



Referências

Links

http://github.com/senai-desenvolvimento

Livros