Projeto Pelinho na Régua

O projeto "Pelinho na Régua" é um sistema de gerenciamento para petshops construído com ASP.NET Core no backend e Vue.js no frontend.

Estrutura do Projeto

Backend

O backend do projeto é construído com ASP.NET Core e usa Entity Framework Core para acesso ao banco de dados.

- **backend**: Diretório principal do backend.
- **backend/Controllers**: Contém os controladores da API.
- **backend/Data**: Contém a configuração do contexto do banco de dados.
- **backend/Migrations**: Contém as migrações do Entity Framework Core.
- **backend/Models**: Contém os modelos de dados.
- **backend/Program.cs**: Configuração inicial e execução do aplicativo ASP.NET Core.

Frontend

O frontend do projeto é construído com Vue.js.

- **frontend**: Diretório principal do frontend.
- **frontend/index.html**: Página HTML principal do aplicativo Vue.js.
- **frontend/src**: Diretório principal do código-fonte do frontend.
  - **/App.vue**: Componente raiz do Vue.js.
  - **/components**: Contém componentes reutilizáveis do Vue.js.
  - **/main.js**: Arquivo principal do JavaScript que inicializa o aplicativo Vue.js.
  - **/router**: Contém a configuração das rotas do Vue.js.
  - **/views**: Contém as páginas principais da aplicação.

Implementação

A comunicação entre o frontend e o backend é feita por meio de APIs fornecidas pelo backend para consumo do frontend. As APIs são definidas nos controladores do backend e são acessíveis por meio de endpoints HTTP. O frontend utiliza Fetch API para fazer requisições HTTP ao backend e obter dados ou enviar informações.

Exemplos de Requisições

frontend

```
function handleLogin() {
  fetch('http://195.200.2.145:5000/api/Users/login', {
    method: 'POST',
    headers: {
```

```
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      username: username.value,
      password: password.value
    })
  })
    .then(response => response.json())
    .then(data => {
      if (data.error) {
        alert(data.error)
        return
      }

      alert('Login feito com sucesso')
      router.push('/')
    })
    .catch(error => {
      console.error('Error:', error)
      alert('Erro ao fazer login')
    })
};
```

backend

```
[HttpPost("login")]
    public IActionResult Login(User user)
    {
      var existingUser = _context.Users.FirstOrDefault(u => u.Username == user.Username);
      if (
        existingUser == null ||
        !BCrypt.Net.BCrypt.Verify(user.Password, existingUser.Password))
      {
        return Unauthorized(new { error = "Usuario ou senha inválidos" });
      }

      return Ok(new { message = "Logado com sucesso" });
    }
```

Estruturação dos dados

Model do Usuário

```
public class User
{
  public int Id { get; set; }
  public required string Username { get; set; }
  public required string Password { get; set; }
}
```

Como Executar o Projeto

Backend

1. Navegue até o diretório do backend, restaure as dependências do projeto, aplique as migrações do banco de dados e execute a aplicação:

```
cd backend
dotnet restore
dotnet ef database update
dotnet run
```

Frontend

1. Navegue até o diretório do frontend, instale as dependências do projeto e execute o aplicativo:

```
cd frontend
npm install
npm run dev
```