Proposta do Projeto 2
AS64A - Programação Web Fullstack

Prof. Willian Massami Watanabe

---

1 Descrição do projeto

O Projeto 2 da disciplina de Programação Web Fullstack será realizada por meio da implementação de um projeto de aplicação web. O projeto considera o desenvolvimento de uma aplicação web em 3 camadas: **Front-end**, **Back-end HTTP** e **Banco de dados**. Os alunos devem implementar o projeto em grupos de até 2 pessoas.

2 Requisitos funcionais

O projeto deve implementar os seguintes requisitos funcionais:

1. Login;
2. Busca;
3. Inserção.

As funcionalidades de Busca e Inserção devem ser realizadas considerando as temáticas selecionadas no PROJETO 1. Nesse sentido, os dados de busca e inserção devem ser similares aos dados recuperados das APIs no PROJETO 1.

Apenas usuários com sessão ativa no sistema (logados) podem realizar busca e inserção. O sistema pode conter um conjunto de usuários inseridos no banco, para realizar o Login. Não há a necessidade de implementar rotinas de inserção de usuários pela aplicação web.

3 Arquitetura do sistema

O sistema deve ser implementado com 3 camadas. A seguir, são detalhados requisitos técnicos, considerando cada camada do sistema:

Front-end : deve ser implementado utilizando a biblioteca de front-end React.js. Toda a comunicação com o Back-end deve ser realizada por meio de requisições HTTP, caracterizando uma *Single-Page Application* - SPA;

Back-end HTTP : deve ser implementado utilizando Express.js. A comunicação com o Front-end deve seguir o padrão de rotas Restful. Esse servidor terá acesso direto ao banco de dados;

Banco de dados : pode ser utilizado qualquer sistema de gerenciamento de banco de dados da escolha dos alunos.

4 Estrutura do Projeto

O projeto deve ser armazenado em um único repositório de código fonte, com duas pastas separadas. Uma para a aplicação do Back-End (**backend**) e outra para o Front-End (**frontend**).

Na pasta da aplicação do Back-End, a pasta dos códigos-fontes do projeto deve apresentar APENAS as seguintes pastas:

- **src/routes**: com os arquivos de rotas da aplicação. Na definição das rotas, deve ser inserido o código dos controladores. Os controladores não devem ser inseridos em uma pasta separada;
- **src/models**: com arquivos que definem as classes de acesso ao banco de dados;
- **src/config**: com arquivos de configuração do banco de dados, servidores de cache, entre outros.

Na pasta da aplicação do Front-End, a estrutura de arquivos deve seguir a mesma definida no PROJETO 1.

5 Critérios de avaliação

Qualquer biblioteca utilizada não aprovada previamente pelo professor será desconsiderada da avaliação. Se for constatada cópia de artefatos de outros projetos, será atribuída nota zero.

O projeto deverá ser executado e apresentado no laboratório P102, em uma das máquinas disponíveis dentro do laboratório (Linux).. Na apresentação, devem ser apresentados o código-fonte (tirando possíveis dúvidas do professor), a execução do projeto e as funcionalidades implementadas.

Os critérios de avaliação desse projeto são definidos a seguir:

- Implementação de cada requisito (Login, Busca e Inserção) no Frontend com React.js, utilizando a mesma estrutura do PROJETO 1.
- Implementação de cada requisito (Login, Busca e Inserção) no Backend com Express.js, utilizando a estrutura de pastas definida nesta proposta.
- Verificação de preenchimento de campos no servidor.
- Envio de mensagens de validação do servidor.
- Implementação do padrão REST na API desenvolvida.
- Implementação de regras de segurança associadas às seguintes categorias de ataques de aplicações web:
  - Falhas de criptografia (uso de HTTPS e armazenamento de senhas utilizando criptografia);
  - Injeção (uso de *sanitizers* de parâmetros, prevevir ataques de SQL/NoSQL inject e XSS);
  - Falhas de identificação e autenticação (prevenir ataques automatizados, invalidar corretamente tokens de autenticação);
  - Falhas de registro e monitoramento de segurança (registrar erros de autenticação, buscas e postagens em logs).
- Implementação de otimização do Front-end:

  - Compressão de arquivos estátivos;
  - Compressão de respostas do servidor.
- Implementação da estratégia de cache no Back-end.
- Configuração do padrão de pool de conexões.