BACKEND COM JAVASCRIPT

Atividade Prática: Desenvolvimento de um Sistema CRUD com Autenticação utilizando Node.js, Express e SQLite

Professor: Diego de Souza Rodrigues

Ouro Preto, Minas Gerais
2025

BACKEND COM JAVASCRIPT

Ouro Preto, Minas Gerais
2026

Índice

1. Objetivo da Atividade

Desenvolver uma aplicação web completa baseada em arquitetura MVC, implementando autenticação de usuários, controle de sessão, proteção contra ataques comuns e operações CRUD persistidas em banco de dados relacional SQLite.
A atividade integra conceitos fundamentais de:

- Estruturação de aplicações backend
- Segurança em aplicações web
- Controle de sessão e autenticação
- Criptografia de senhas
- Validação de dados
- Persistência com banco relacional
- Organização modular do código

2. Dependências Obrigatórias

O projeto deve utilizar explicitamente as seguintes dependências, cada uma aplicada conforme sua finalidade arquitetural e de segurança:

- **express** – Framework para criação do servidor HTTP, definição de rotas e middlewares.
- **ejs** – Motor de templates para renderização dinâmica das views.
- **sqlite3** – Driver de conexão com o banco de dados SQLite.
- **sqlite** – Interface baseada em Promises para interação com o SQLite utilizando async/await.
- **express-session** – Gerenciamento de sessões do usuário autenticado.
- **connect-sqlite3** – Armazenamento persistente das sessões no banco SQLite.
- **bcryptjs** – Criptografia de senhas por meio de hashing seguro.
- **validator** – Validação e sanitização de dados de entrada (ex.: validação de e-mail).
- **connect-flash** – Exibição de mensagens temporárias (ex.: erro de login, sucesso no cadastro).
- **csurf** – Proteção contra ataques CSRF (Cross-Site Request Forgery).

- **helmet** – Configuração de headers HTTP de segurança.

3. Requisitos Funcionais

3.1 Páginas Obrigatórias

O sistema deve conter, no mínimo, as seguintes rotas e views:

1. **Home**
    - Página inicial pública.
    - Exibir links para login e cadastro.
    - Caso usuário esteja autenticado, exibir menu de navegação.
    - Exibir itens cadastrados
2. **Cadastro de Usuário**
    - Campos:
        - Nome
        - Sobrenome
        - Email
        - Senha
        - Repetir senha
    - Validação de:
        - Email válido
        - Senha mínima (ex: 6 caracteres)
        - Senhas iguais
    - Senha deve ser criptografada com bcryptjs.
3. **Login**
    - Campos:
        - Email
        - Senha
    - Criar sessão após autenticação válida.
    - Exibir mensagens de erro com connect-flash.
4. **Cadastro de Itens**
    - O aluno deve escolher o tipo de entidade:

        - Produto
        - Cliente
        - Pessoa
        - Paciente
        - Ou outra coisa que deseje
    - Apenas usuários autenticados podem acessar.
5. **Listagem de Itens**
    - Listar todos os registros cadastrados.
    - Exibir botões de editar e excluir.
6. **Edição de Itens**
    - Permitir atualizar dados.
    - Apenas usuário autenticado.
7. **Exclusão de Itens**
    - Apenas usuário autenticado.
    - Deve exigir token CSRF.
8. **Edição de Perfil do Usuário**
    - Permitir alterar:
        - Nome
        - Sobrenome
        - Email
    - Alteração de senha opcional (com confirmação).

4. Requisitos de Segurança

O sistema deve implementar obrigatoriamente:

- helmet para segurança de headers HTTP
- express-session para controle de sessão
- connect-sqlite3 como store de sessão
- csurf para proteção contra CSRF
- bcryptjs para hash de senha
- validator para validação de dados

O acesso às rotas de CRUD deve ser protegido por middleware de autenticação.

5. Estrutura Recomendada do Projeto

```
/database
/src
  /controllers
  /models
  /routes
  /middlewares
  /views
  /database
server.js
```

Separar claramente:

- Camada de roteamento
- Camada de controle
- Camada de acesso ao banco
- Middlewares de autenticação

6. Requisitos Técnicos de Implementação

O aluno deverá implementar:

- Middleware isAuthenticated
- Middleware global de CSRF
- Middleware para mensagens flash
- Uso de async/await
- Tratamento de erros
- Organização modular

7. Critérios de Avaliação

Critério	Peso
Organização do Projeto	5.0
Implementação correta do CRUD	5.0
Sistema de Autenticação	5.0
Segurança (CSRF, hash, sessão)	5.0
Validação de tratamento de erros	5.0
Apresentação	10

9. Entregáveis

O aluno deve entregar o projeto completo totalmente versionado no GitHub, contendo:

- Código completo do Projeto
- Script SQL ou inicialização automática do banco
- README contendo:
  - Descrição do projeto.
  - Como instalar
  - Como executar
  - Estrutura do projeto

10. Referências Bibliográficas

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine.

Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, 14(6), 80–83.

OWASP Foundation. (2021). *OWASP Top Ten Web Application Security Risks*. Disponível em: https://owasp.org/www-project-top-ten/

Express.js Documentation. (2023). *Express Guide*. Disponível em: https://expressjs.com/

SQLite Documentation. (2023). *SQLite Architecture*. Disponível em: https://www.sqlite.org/docs.html