|  | Ministério da Educação  UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  Campus Medianeira  Bacharelado em Ciência da Computação |  |
| --- | --- | --- |

| **PROJETO FINAL**  **ETAPA 1 - BACK-END** |
| --- |

| **Professor** |
| --- |
| Ricardo Sobjak |

| **Disciplina** |
| --- |
| Linguagem de apresentação e estruturação de conteúdo |

| **Licença** |
| --- |
|  |

| **Descrição** |
| --- |
| Esta atividade refere-se à etapa 1 do desenvolvimento de um projeto final, resultando no back-end (PHP, banco de dados) da aplicação. |

| **Objetivo** |
| --- |
| Aplicar os conceitos de PHP no de uma API (back-end) de uma aplicação, utilizando banco de dados. |

| **Bibliografia recomendada** |
| --- |
| **Livros disponíveis na Minha Biblioteca da UTFPR**:   * Desenvolvimento de sistemas com PHP: <https://integrada.minhabiblioteca.com.br/reader/books/9788595023222> * PHP: programe de forma rápida e prática: <https://integrada.minhabiblioteca.com.br/reader/books/9786558110224>   **Banco de Dados**   * Funções do MySQL (PDO\_MYSQL): <https://www.php.net/manual/pt_BR/ref.pdo-mysql.php> * PHP PDO: Como se conectar ao banco de dados: <https://www.devmedia.com.br/php-pdo-como-se-conectar-ao-banco-de-dados/37211> |

**Orientações**:

* Pode ser executado em duplas.
* Deverá ser apresentado para o professor.

**Equipes e temas do e-commerce:**

| **EQUIPES** | **SEGMENTO** | **Status** |
| --- | --- | --- |
| Henrique e João Vitor Ferraz | Auto peças | Banco: OK  API: em andamento |
| Isabela | Plataforma de jogos | Banco: OK  API: quase pronta.  Segurança: … |
| Osvaldo e Estefani | Peças de cerâmica | Banco: OK  API: quase pronta.  Segurança: |
| Paulo e Eduarda | Batom | Banco: OK  API: não iniciaram |
| Daniel | Bike | Banco: OK.  API: criando usuário. |
| Enzo e Bernard | Bebidas | Banco: OK  API: quase pronta.  Segurança: …. |
| William e Johnny | Glórias passadas: camisas de times | Não iniciado |
| Vinicius e Isis | Videolocadora | Não iniciado |
| Bruno e Laerte | Materiais elétricos e eletrônicos | Não iniciado |
| Mohamed e Felipe Zonta | Vendas em geral | Banco: OK  API: iniciando |
| Rafael Henrique |  |  |
| Tales |  | Não iniciado |
| João Otávio e Matheus Bieger | Loja de música |  |
| Matheus Cunha | Bijuteria | Banco: |
| Gabriel Tadeu |  | Banco: quase pronto |
| João Gomes e Marcos Vinicius | Mercado | Banco: ok.  API: quase pronta.  Segurança: … |
| David e Orlando |  |  |

**Desenvolvimento de um sistema de compras online**

O projeto final será composto por duas etapas, você deve desenvolver um **sistema de compras online** utilizando as tecnologias de back-end (PHP) e front-end (HTML, CSS e JavaScript).

A primeira etapa do projeto envolve apenas as tecnologias de **back-end**.

A aplicação deve permitir que o usuário cadastre uma **conta de usuário**, **navegue por diferentes categorias de produtos**, adicione itens ao **carrinho de compras** e **finalize a compra**.

O **back-end** deve ser responsável por gerenciar os **dados de produtos**, como nome, descrição, preço e quantidade em estoque, além de gerenciar o processo de **compra dos produtos.**

O **front-end** (Etapa 2) deve ser responsável por **exibir os produtos** em diferentes categorias, permitir a adição de produtos ao **carrinho de compras**, exibir o valor total da compra e permitir que o usuário **finalize a compra**.

Será necessário utilizar técnicas de validação de formulários para garantir que os dados de cadastro e pagamento sejam inseridos corretamente e que a segurança da aplicação seja garantida.

Ao final da atividade, você terá desenvolvido uma aplicação web completa de e-commerce, utilizando as tecnologias fundamentais para o desenvolvimento de sistemas web.

**BACK-END**

O **back-end** da aplicação será implementado na forma de uma API (Application Programming Interface) web, para disponibilizar dados em formato JSON para o front-end.

A API deverá ser implementada na linguagem **PHP**.

**1) BANCO DE DADOS**

Armazenar os dados no banco de dados MySQL. Será necessário criar as seguintes tabelas:

* **tb\_usuario**: armazenar os dados dos usuários cadastrados no sistema, como nome, data de nascimento, email, senha e flag indicando se o usuário é administrador.

| **tb\_usuario** |
| --- |
| id: integer (chave primária) |
| nome: varchar |
| email: varchar |
| senha: varchar |
| nascimento: date |
| admin: interger |

* **tb\_produto**: Armazena as informações dos produtos disponíveis para compra, como nome, descrição, preço e quantidade em estoque.

| **tb\_produto** |
| --- |
| id: integer (chave primária) |
| nome: varchar |
| descricao: varchar |
| preco: float |
| quantidade: integer |

* **tb\_categoria**: armazena as categorias em que os produtos podem ser classificados.

| **tb\_categoria** |
| --- |
| id: integer (chave primária) |
| nome: varchar |

* **tb\_produto\_categoria**: é uma tabela associativa entre produtos e categorias, permitindo que um produto pertença a mais de uma categoria.
  + *Alteração em 02/06/2023 → Esta tabela torna-se opcional. Se você quiser simplificar o projeto, pode eliminar esta tabela e fazer uma chave estrangeira entre a tabela tb\_produto vs tb\_categoria.*

| **tb\_produto\_categoria** |
| --- |
| id\_produto: integer (chave estrangeira) |
| id\_categoria: integer (chave estrangeira) |

* **tb\_compra**: armazena informações sobre as compras realizadas pelos usuários, incluindo o valor total da compra e a data da compra.

| **tb\_compra** |
| --- |
| id: integer (chave primária) |
| id\_usuario: integer (chave estrangeira) |
| valor: float |
| data: date |

* **tb\_compra\_produto**: é uma tabela associativa entre compras e produtos, permitindo que uma compra contenha mais de um produto.

| **tb\_compra\_produto** |
| --- |
| id\_compra: integer (chave estrangeira) |
| id\_produto: integer (chave estrangeira) |
| quantidade: float |

Obs.: Criar um arquivo em PHP no projeto para definir toda a estrutura do banco de dados, ou seja, criar as tabelas.

**2) RECURSOS HTTP**

Implementar os recursos HTTP para serem acessados a partir do front-end.

Considere que o termo **<app>** usado na sequência seja a URL raiz da aplicação dentro do servidor web, por exemplo, <http://localhost/ecommerce/api>.

Você pode utilizar a API de exemplo desenvolvida em aula:

* <https://github.com/ricardosobjak/cc53b/tree/master/5.FrontBack/webapp>

**Usuário**

| **URL do recurso** | **Descrição** | **Método HTTP** | **Permissão** |
| --- | --- | --- | --- |
| <app>/usuario/create.php | Criar um usuário | POST | Público |
| <app>/usuario/get.php | Obter todos os usuários ou somente pelo ID. | GET | Admin |
| <app>/usuario/put.php | Atualizar um usuário | POST | Usuário logado ou admin |
| <app>/usuario/delete.php | Deletar um usuário | GET | Usuário logado ou admin |

**Categoria**

| **URL do recurso** | **Descrição** | **Método HTTP** | **Permissão** |
| --- | --- | --- | --- |
| <app>/categoria/create.php | Criar uma categoria | POST | Admin |
| <app>/categoria/get.php | Obter todas as categorias ou somente pelo ID. | GET | Público |
| <app>/categoria/put.php | Atualizar uma categoria | POST | Admin |
| <app>/categoria/delete.php | Deletar uma categoria | GET | Admin |

**Produto**

| **URL do recurso** | **Descrição** | **Método HTTP** | **Permissão** |
| --- | --- | --- | --- |
| <app>/produto/create.php | Criar um produto | POST | Admin |
| <app>/produto/get.php | Obter todos os produtos, por categoria ou somente pelo ID. | GET | Público |
| <app>/produto/put.php | Atualizar um produto | POST | Admin |
| <app>/produto/delete.php | Deletar um produto | GET | Admin |

**Compra**

| **URL do recurso** | **Descrição** | **Método HTTP** | **Permissão** |
| --- | --- | --- | --- |
| <app>/compra/create.php | Criar uma compra | POST | Usuário logado ou admin |
| <app>/compra/get.php | Obter todas as compras de um usuário ou pelo ID | GET | Usuário logado ou admin |
| <app>/compra/put.php | Atualizar uma compra | POST | Usuário logado ou admin |
| <app>/compra/delete.php | Deletar uma compra | GET | Usuário logado ou admin |

**Login**

| **URL do recurso** | **Descrição** | **Método HTTP** | **Permissão** |
| --- | --- | --- | --- |
| <app>/auth/login.php | Fazer login, ou seja, obter um token de autenticação | POST | Público |

**3) SEGURANÇA**

A API deve permitir que o usuário **faça autenticação**, gerando um token JWT e **restringir o acesso** aos recursos da aplicação.

Utilize a especificação de token JWT → <https://jwt.io>.

Você pode utilizar o código de exemplo desenvolvido em aula:

* <https://github.com/ricardosobjak/cc53b/tree/master/4.PHP/autenticacao-jwt>