**Dokumentacja Projektu E-commerce**

**Wprowadzenie**

Projekt E-commerce jest aplikacją internetową umożliwiającą użytkownikom rejestrację, logowanie, zakupy oraz zarządzanie koszykiem. Projekt jest oparty na wzorcach projektowych MVC (Model-View-Controller), co zapewnia oddzielenie logiki aplikacji od interfejsu użytkownika. Aplikacja korzysta z bazy danych MySQL, a wszystkie operacje na bazie danych są realizowane za pomocą klasy PDO.

**Struktura projektu**

Projekt składa się z kilku głównych komponentów:

* **App** – główny folder aplikacji, zawierający logikę backendową.
  + Validator.php – klasa odpowiedzialna za walidację danych wejściowych użytkowników (np. e-mail, hasło).
  + Request.php – klasa umożliwiająca obsługę żądań HTTP (GET, POST, sesje).
  + UserController.php – kontroler zarządzający operacjami związanymi z użytkownikami, takimi jak logowanie, rejestracja, koszyk.
  + UserModel.php, StoreModel.php – modele do obsługi użytkowników i produktów.
  + AbstractModel.php – klasa bazowa dla modeli łącząca się z bazą danych.
* **App/Views** – folder przechowujący widoki aplikacji.
  + view.php – klasa odpowiedzialna za renderowanie widoków.

**Klasy**

**1. Validator**

Klasa Validator zapewnia metody do walidacji danych wejściowych, takich jak e-mail, hasło, czy obecność użytkownika w bazie danych. Oto dostępne metody:

* **email(string $email): string** – waliduje poprawność e-maila. Zwraca komunikat o błędzie, jeśli e-mail jest pusty lub w niepoprawnym formacie.
* **userExist(int $numberOfUsers): string** – sprawdza, czy użytkownik o podanym e-mailu już istnieje w bazie danych.
* **username(string $username): string** – waliduje nazwę użytkownika. Pozwala na wprowadzenie tylko liter i spacji.
* **checkPassword(string $password): string** – waliduje hasło, sprawdzając, czy zawiera przynajmniej jedną dużą literę i jedną cyfrę oraz ma co najmniej 8 znaków.
* **confirmPassword(string $password, string $confirm\_password): string** – sprawdza, czy hasło i potwierdzenie hasła są takie same.

**2. Request**

Klasa Request umożliwia łatwą obsługę żądań HTTP. Przechowuje dane z żądań GET, POST, sesji oraz serwera.

* **isPost(): bool** – sprawdza, czy żądanie jest metodą POST.
* **hasPost(): bool** – sprawdza, czy dane POST są dostępne.
* **get(string $param, $default = null)** – pobiera wartość parametru GET.
* **post(string $param, $default = null)** – pobiera wartość parametru POST.
* **session(string $param, $default = null)** – pobiera wartość z sesji.
* **hasSession(): bool** – sprawdza, czy dane w sesji są dostępne.

**3. UserModel**

Klasa UserModel odpowiada za operacje na użytkownikach w bazie danych.

* **create(array $data): void** – tworzy nowego użytkownika w bazie danych.
* **getUser(string $email): array** – pobiera dane użytkownika na podstawie e-maila.
* **getByEmail(string $email): int** – sprawdza, czy użytkownik o podanym e-mailu istnieje w bazie.
* **GetUserCart(int $userId): array** – pobiera produkty z koszyka użytkownika.
* **AddProductToCart(array $cartData): void** – dodaje produkt do koszyka użytkownika.
* **DeleteProductFromCart(int $cartId): void** – usuwa produkt z koszyka użytkownika.
* **AddAddress(array $data): string** – dodaje adres dostawy do bazy danych.
* **CreateOrder(array $data): string** – tworzy zamówienie.
* **AddProductsToOrder(array $data, int $orderId): void** – dodaje produkty do zamówienia.
* **updatePaymentStatus(int $orderId, string $status): void** – aktualizuje status płatności za zamówienie.

**4. StoreModel**

Klasa StoreModel obsługuje operacje na produktach i kategoriach.

* **GetProducts(?string $category): array** – pobiera listę produktów, opcjonalnie filtrując je według kategorii.
* **GetProductDetails(int $id): array** – pobiera szczegóły produktu na podstawie jego ID.

**5. AbstractModel**

Klasa AbstractModel jest bazową klasą dla modeli, zapewniającą połączenie z bazą danych MySQL.

* **\_\_construct(array $config)** – inicjalizuje połączenie z bazą danych za pomocą PDO.

**6. UserController**

Klasa UserController zarządza logiką aplikacji związaną z użytkownikami, taką jak rejestracja, logowanie, zarządzanie koszykiem, zamówieniami.

* **sign\_inAction(): void** – obsługuje logowanie użytkownika.
* **sign\_upAction(): void** – obsługuje rejestrację nowego użytkownika.
* **logoutAction(): void** – wylogowuje użytkownika.
* **shopping\_cartAction(): void** – wyświetla zawartość koszyka użytkownika.
* **orderAction(): void** – obsługuje proces składania zamówienia.

**Baza Danych**

Projekt korzysta z bazy danych MySQL. Poniżej przedstawiono kluczowe tabele:

* **users** – przechowuje dane użytkowników (ID, name, email, password, created\_at).
* **products** – przechowuje dane produktów (ID, name,description, size, price, stock, image\_url, created\_at ).
* **cart** – przechowuje produkty dodane do koszyka przez użytkowników (ID, user\_id, product\_id, quantity, created\_at).
* **orders** – przechowuje dane zamówień (ID, user\_id,created\_at, total\_price, address\_id, payment\_status).
* **order\_items** – przechowuje produkty dodane do zamówienia (ID, order\_id, product\_id, quantity, price, created\_at).
* **adress** – przechowuje adresy użytkowników (ID, user\_id, street, city, postal\_code, building\_number, firstname, lastname).
* **category** – przechowuje kategorie dla tabeli products(id, name, created\_at)

**Podsumowanie**

Projekt E-commerce to aplikacja pozwalająca na rejestrację, logowanie, zakupy online oraz zarządzanie koszykiem i zamówieniami. Użytkownicy mogą dodawać produkty do koszyka, składać zamówienia, oraz adresami dostawy. Cała logika aplikacji opiera się na wzorcu MVC, co zapewnia łatwą rozbudowę i konserwację systemu.