

Etap 2 - Wytyczne implementacyjne

Wymagania ogólne

Celem projektu jest wzbogacenie wyników pracy z etapu 2 o logikę przetwarzania po stronie serwera (wykorzystując framework Symfony PHP) w wyniku czego powstanie aplikacja zgodna z przyjętym tematem wiodącym (w ogólności odpowiada ona aplikacji sklepu internetowego). Oczekiwany zakres funkcjonalności:

- Wykorzystanie: wzorca MVC, ORM i OOP
- Prawidłowo skonfigurowane środowisko umożliwiające dostęp z komputera hosta
- Różni użytkownicy:
 - Rejestrację, logowanie i wylogowanie użytkownika
 - Kontrola uprawnień w dostępie do określonych akcji i modułów
 - Rozróżnianie uprawnień użytkowników i wyświetlanie zróżnicowanych opcji
 - Możliwość zarządzania użytkownikami oraz ich uprawnieniami
- Współpraca z bazą danych (dodawanie, modyfikowanie, usuwanie danych):
 - Wyświetlanie produktów oraz kategorii
 - Możliwość wyszukiwania produktów po nazwie lub jej części
 - Strona przedstawiająca informacje o produkcie
 - Możliwość dodawania opinii do produktu
 - Możliwość zarządzania produktami i kategoriami w sklepie
- Koszyk zakupowy:
 - Realizacja koszyka w oparciu o sesję
 - Możliwość zarządzania produktami w koszyku (modyfikacja ilości , usunięcie pozycji, podsumowanie, itp)
 - Możliwość złożenia zamówienia na podstawie zawartości koszyka (wykorzystać transakcje)
- Dostępność w dwóch językach: polskim i angielskim
- Content negotiation - negocjowanie treści przesyłanej do klienta (XHTML czy HTML);
- Zaimplementować funkcjonalności przesyłu plików na serwer z ograniczeniem na typ pliku (po stronie serwera i klienta) i ilość przesyłanych danych oraz podstronę prezentującą przesłane pliki (np. lista przesłanych plików - nazwa, typ, rozmiar, data, czas), tak aby można było zidentyfikować właśnie przesłany plik
- Zaimplementować dwa przykłady wykorzystania technologii AJAX - jeden z wykorzystaniem framework'a Symfony , drugi „czysty AJAX” (samodzielne powołanie w kodzie js obiektu asynchronicznej komunikacji, obsługi funkcji callback oraz logiki wołanej po stronie serwera do wytworzenia odpowiedzi zależnie od otrzymanych parametrów (nie gotowy jeden plik xml do pobrania)

Wymagania szczegółowe

Baza danych powinna być skonstruowana w oparciu o plik schema.yml. Nazewnictwo tabel powinno być zgodne ze wzorcem: numerAlbumu_nazwaTabeli. Nazewnictwo pól w tabelach powinno być zgodne ze wzorem: nazwaTabeli_nazwaPola. Projekt powinien dysponować przygotowanymi danymi w katalogu /data/fixtures/ umożliwiając sprawdzającemu wywołanie polecenia php symfony propel:data-load. Na podstawie tych danych nastąpi weryfikacja działania aplikacji. Dane powinny być wystarczające do prezentacji wszystkich elementów systemu. Nie należy przechowywać haseł w bazie danych w postaci nieszyfrowanej.

Rejestracja, logowanie i wylogowanie użytkownika

Wykorzystanie gotowych mechanizmów autoryzacji wbudowanych w Symfony PHP. Przechowywanie

dodatkowych informacji o użytkowniku po jego uwierzytelnieniu. Wyczyszczenie wszystkich informacji przechowywanych wraz z użytkownikiem podczas wylogowania. Rejestracja i logowanie powinny być realizowane w oparciu o formularze utworzone poprzez dziedziczenie z sfForms i umieszczone w apps/frontend/lib/forms/. Każde pole powinno być sprawdzane za pomocą wbudowanych walidatorów lub post walidatorów (hasła). Należy zrealizować również jeden własny walidator (klasę) i podpiąć go pod dowolne pole formularza.

Kontrola uprawnień w dostępie do określonych akcji i modułów

Należy wykorzystać w tym celu pliki security.yml określając zakres uprawnień wymagany do wyświetlenia danej akcji lub modułu. Należy również wykorzystać wbudowane mechanizmy uprawnień (Credentials) w celu określenia dostępu do określonych akcji - przykład w materiałach wykładowych.

Rozróżnianie uprawnień użytkowników i wyświetlanie zróżnicowanych opcji

Aplikacja powinna udostępniać różne funkcje w zależności od poziomu użytkownika. Przykładowo manager powinien mieć możliwość usuwania wszystkich opinii z produktu, użytkownik wyłącznie swoich. Manager powinien mieć możliwość akceptacji opinii przed ich wyświetleniem na stronie produktu. Menu operacji dla użytkowników z różnymi uprawnieniami powinno być inne.

Wyświetlanie produktów oraz kategorii

Przy wyświetlaniu produktów powinna istnieć możliwość wyboru kategorii - filtrującej listę produktów. Aplikacja powinna udostępniać możliwość sortowania wyników po cenie lub nazwie. Produkty powinny być wyświetlane po 5 sztuk na widok z wykorzystaniem wbudowanej klasy "sfPropelPager". Kliknięcie w produkt powinno przenosić do strony produktu. Szablon strony produktu powinien być jeden uniwersalny.

Możliwość wyszukiwania produktów po nazwie lub jej części

Aplikacja powinna udostępniać pole do wyszukania produktu po nazwie lub jej części. Należy zwrócić uwagę na realizację zapytania do bazy danych. Zapytanie powinno być zrealizowane z wykorzystaniem ORM (Propel) bez omijania go zapytaniem SQL.

Realizacja koszyka w oparciu o sesję

Należy utworzyć koszyk, który będzie przechowywany razem z informacjami dodatkowymi o użytkowniku w jego tablicy sesyjnej (w kontrolerze \$this->getUser()). Informacje o stanie koszyka powinny być dostępne w każdym widoku jeśli użytkownik jest klientem i zalogował się prawidłowo.

Dostępność w dwóch językach: polskim i angielskim

Aplikacja powinna wykorzystywać moduł i18n zawarty w Symfony php do internacjonalizacji witryny. Opcje jakie powinny być dostępne do możliwość zmiany języka sklepu w dowolnym momencie. Etykiety w menu i formularzach powinny zmieniać się w zależności od języka - tłumaczenia powinny być zawarte w plikach w katalogu i18n/. Nazwy produktów i kategorii również powinny być zależne od wybranego języka - realizacja na podstawie tabel w bazie danych. Wszystkie liczby będące walutą powinny być sformatowane według danej lokalizacji oraz powinny zawierać oznaczenie waluty. Jeśli aplikacja wyświetla daty należy je również przedstawiać zgodnie z wybranym językiem.

Możliwość zarządzania produktami w koszyku

Aplikacja powinna umożliwiać klientowi zarządzanie produktami w koszu. Akcje jakie powinny być udostępnione to: zmień ilość (dot. produktu), usuń produkt (z kosza), wyczyść kosz. Akcje powinny być przeprowadzone z użyciem technologii AJAX. Przynajmniej jedna musi być zrealizowana przy pomocy

samodzielnie napisanego skryptu („czysty AJAX”).

Możliwość złożenia zamówienia na podstawie zawartości koszyka

Aplikacja powinna umożliwiać klientowi złożenie zamówienia w dowolnym momencie w aplikacji. W rezultacie wyboru tej akcji klient powinien zostać przekierowany do strony z formularzem zamówienia, a następnie po akceptacji dane powinny zostać raz jeszcze przedstawione do potwierdzenia. Po potwierdzeniu użytkownik ma zostać poinformowany o przyjęciu zamówienia. Koszyk ma zostać opróżniony.

Możliwość dodawania opinii do produktu

Klient powinien mieć możliwość złożenia opinii o produkcie na stronie produktu. Wymagane jest by był zalogowany i miał uprawnienia użytkownika. Opinia składa się z oceny oraz treści opinii. Wprowadzona opinia oczekuje na akceptację managera - nie może być szybciej wyświetlana.

Strona przedstawiająca informacje o produkcie

Strona przedstawiająca informacje o produkcie powinna składać się z elementów takich jak: zdjęcie, nazwa, opis, cena, opinie, możliwość zakupu, 3 losowe produkty z danej kategorii. Strona powinna udostępniać galerię produktu po wciśnięciu na zdjęcie. Galeria powinna być zrealizowana przy pomocy Javascript. Nie wolno używać gotowych pluginów i skryptów js.

Możliwość zarządzania produktami i kategoriami w sklepie (+ przesył plików)

Projekt powinien udostępniać drugą aplikację tzw. backend umożliwiającą dostęp wyłącznie dla użytkownika z uprawnieniami administratora (rola np. "admin"). Backend powinien umożliwiać zarządzanie produktami przez formularze wygenerowane na podstawie generators.yml natomiast zarządzanie kategoriami należy zrealizować samodzielnie. Administrator powinien mieć możliwość dodawania, edycji, usunięcia i listowania wszystkich produktów i kategorii. Formularz dodawania produktu powinien udostępniać możliwość wgrania zdjęć produktu na serwer - tworzących galerię.

Możliwość zarządzania użytkownikami oraz ich uprawnieniami

Aplikacja powinna udostępniać możliwość zarządzania użytkownikami i grupami dla użytkowników z uprawnieniami administratora. Należy przygotować w aplikacji backend samodzielnie opcje umożliwiające zarządzanie grupami (dodawanie, edycja, usuwanie, listowanie) oraz użytkownikami (dodawanie, edycja, usuwanie, listowanie). Aplikacja powinna sprawdzać czy usuwana grupa nie jest związana z jakimś użytkownikiem oraz czy usuwając administratora nie jest on aktualnie zalogowany lub też nie jest jedynym administratorem w systemie.

Uwagi końcowe

W miejscach gdzie nie zostało określone inaczej nie wolno używać modułów, pluginów i skryptów gotowych, dostępnych w Internecie.

Aplikacja nie powinna zawierać ostrzeżeń i błędów podczas prezentacji jej w środowisku dev.

Do realizacji wyglądu aplikacji należy wykorzystać plik layout.php, pliki css, elementy takie jak component oraz partial. Wskazane jest realizacja poszczególnych componentów i partiali tak, aby były jak najczęściej wykorzystywane w całej aplikacji.

Całość powinna się prawidłowo walidować na <http://jigsaw.w3.org/cssvalidator> oraz <http://validator.w3.org>
Projekt powinien być spakowany w plik zip o następującej nazwie: nazwisko_imie_indeks.zip