Raport projekt Django CarShop

Jakub Rymarski

Ogólny opis:

Celem projektu było stworzenie platformy do wypożyczania samochodów. Aplikacja służy użytkownikom zarówno do wyszukiwania ofert, jak i do wystawiania samochodów.

Poszukujący samochodu:

- może wyszukiwać różne samochody, które mają dokładne opisy (marka, typ, rodzaj przekładni)

- może filtrować po różnych kategoriach (sportowe, kombi, sedany, kombi) i po kluczowych słowach

- dopiero po zalogowaniu może złożyć prośbę o wypożyczenie samochodu

Wystawiający samochód do wypożyczenia:

-muszą utworzyć konto i zalogować się (aby korzystać z poniższych możliwości)

-może dodać nowe (wystawić) samochody

- może opisać swój samochód, wstawić ładne zdjęcie

- może edytować już wystawiony samochód

- nie może usuwać samochodu (tą opcję ma tylko administrator, ze względu na wymagania projektowe)

Admin może:

- zarządzać z panelu admina wszystkimi modelami i użytkownikami (dodawać, zmieniać, usuwać)

-może usuwać wystawione samochody

Budowa:

W projekcie są 4 główne aplikacje:

cars-informacje ogólne o samochodach (modele Car i Category)

individuals-informacje o wystawionych autach danego użytkownika

rents-obsługa próśb o wypożyczenie

shop-ogólny wygląd i funkcje strony

Dodatkowo w projekcie znajduje się folder **shop/static** z plikami css, js i folder **sent_emails**, w którym zapisywane są wysyłane przez serwer fake emails. W folderze templates/registration znajdują się templates używane przy logowaniu i resetowaniu hasła.

Potwierdzenie wymagań z pliku requirements:

Core:

- Web application written in Django – aplikacja jest napisana w Django.

- at least 3 views –kilkanaście widoków w plikach views.py w folderach cars, individuals, rents, shop.

- at least 3 models – mam 3 modele: Car Category w folderze cars, Rent w folderze rents.

- at least 1 template –kilkanaście templates we wszystkich folderach
  use of a minimum of 30 different HTML tags – użyłem ponad 30 różnych tagów.

Design:

- Styled with CSS sheet – podstawą definiowania stylu jest kilka plików CSS, w nielicznych przypadkach, gdy nie było potrzeby używać osobnego pliku CSS definiowałem styl bezpośrednio w pliku html.
  Using a minimum of 30 different attributes for at least 10 different selectors. Non-trivial selectors (for identifier, class, descendant) must be used – używałem różnych atrybutów (table, hover, margin, cursor i wiele innych), korzystałem m.in. z identyfikatorów czy klas.

- The application is aesthetically pleasing (the use of pre-made templates is prohibited) – wygląd jest estetyczny, odpowiedni dobór kolorów i wielkości danych elementów.

- The application is responsive (i.e. adapted to at least two resolutions - e.g. computer and tablet or computer and smartphone) – odpowiednia konfiguracja html i stylizacja.

Users:

- The application allows you to create a user account (with confirmation of address-email) – użytkownik może założyć konto (podaje nazwę, email i hasło) Jako, że wykorzystałem folder sent_emails do fake email mechanism w późniejszym podpunkcie odnoszącym się do resetowania hasła, tutaj zaimplementowałem go ręcznie – po poprawnym uzupełnieniu formularza, użytkownik zostaje przekierowany do innej strony (z podanym jego emailem), na której widnieje przycisk do potwierdzenia rejestracji. Dopiero po kliknięciu w ten przycisk użytkownik zostanie dodany do bazy.

- The application allows login – aplikacja pozwala zalogować się (obsługa logowania za pomocą „django.contrib.auth").

- The application allows only the logged-in user to access restricted resources – tylko zalogowani użytkownicy mogą wystawiać samochody i prosić o wypożyczenie, niezalogowani mogą jedynie wyszukiwać ofert. Użyłem dekoratora „@login_required".

- The application allows password reset (required to send emails with a link to reset using the fake mail mechanism) – przy logowaniu po kliknięciu "Forgot your password?" użytkownik zostanie przekierowany do strony z resetowaniem hasła. Po ustawieniu nowego hasła na adres email użytkownika zostaje przysłany link aktywacyjny, po którego kliknięciu hasło zostanie zmienione (maile z odpowiednim adresem email przychodzą do folderu sent_emails).

Data processing and froms:

- The application has a form with at least 4 different types of fields-aplikacja ma kilka formularzy z różnymi typami pól. Np, formularz do dodawania samochodu NewCarForm, plik forms.py w folderze cars, ma pole typu wybór (category), pole tekstowe (name), pole do wpisania liczby (price), a także pole do wgrania pliku (image). Formularz RentForm w folderze rents ma jeszcze pola do wybierania daty.
- The application allows you to save or modify records – dodawanie, edytowanie i usuwanie samochodów powoduje zapisywanie zmian w bazie (views.py w folderze cars).

- The application has a view that allows you to display the saved records – np. w folderze individuals w pliku views.py mamy główny widok „index" wyświetlający wszystkie samochody, których właścicielem jest dany użytkownik.

- The application uses XML + XLS technology – użyłem XML do jednego widoku "contact" w pliku views.py w folderze shop. Widok ten ma za zadanie zwrócić dane kontaktowe o mojej firmie CarShop (owner, email, address, phone), które są zawarte w pliku shop/templates/shop/info_contact.xml. Wygląd zdefinowany jest w pliku contact.xsl.

- The form has validation rules on the server side – przy wypełnianiu formularza, np. do zakładania konta, są m.in sprawdzane: czy już nie istnieje użytkownik o takim samej nazwie, albo czy hasło jest odpowiednio długie.

- The form has validation rules on the client side – aby spełnić to wymaganie zaimplementowane jest sprawdzenie, czy użytkownik nie podaje w trakcie rejestracji nazwy użytkownika „CarShop", czyli nazwy mojej firmy. Logika znajduje się w pliku static/script_Username.js.

- Data can be modified from the admin panel level – zarejestrowałem modele w admin.py, tak że mogą być one modyfikowane z panelu admina.
  At least three models are registered in the admin panel-wszystkie trzy modele są zarejestrowane w panelu admina.

- An erroneously filled form returns meaningful error messages and does not force the user to re-fill the form all over again – błędy są wyświetlane i nie trzeba jeszcze raz wypełniać wszystkich pól, np. gdy podczas zakładania konta, użytkownik źle powtórzy hasło, wyświetla się pod spodem na czerwono komunikat, że hasła nie są takie same.

Common:

- The application follow good programming practices
  DRY principle, short functions, no deeply nested loops and if's, comments and meaningful names of variables, objects, functions – zachowałem zasady dobrego programowania, nie powtarzałem kodu tylko tworzyłem funkcje, rozszerzałem tamplates zamiast przepisywać ten sam kod. Nadawałem odpowiednie, znaczące nazwy dla danych folderów, plików czy klas w CSS. Umieszczałem komentarze w kluczowych miejscach.

- The application has no difficulty in deployment on a separate machine, has a requirements.txt file, the only necessary adjustments take place at most in the settings.py file, it is allowed to use docker – aplikacja ma plik requirements.txt i nie powinno być problem z jej wdrożeniem na innym urządzeniu.

- For particularly high quality design- starałem się, aby wszytko było zapisane schludnie i efektywnie.

Może się przydać:

Konto admina

Username: admin

Password: Django123

Konto zwykłego użytkownika posiadającego samochód

Username: kuba1234

Password: Django1234