Maciej Bolsęga

Projektowanie aplikacji internetowych

1. Identyfikacja zagadnienia biznesowego

Celem projektu było stworzenie aplikacji webowej służącej do handlu samochodami. Stworzona aplikacja umożliwia niezalogowanemu użytkownikowi przegląd aktualnych ogłoszeń. Po stworzeniu konta i zalogowaniu użytkownik może również dodawać własne ogłoszenia, dodawać je do koszyka oraz dokonywać płatności.

Stworzona strona jest jedynie rozwiązaniem „Proof of concept” - nie jest to rozwiązanie kompletne, mogące stanowić wartość biznesową, a jedynie fundament do stworzenia takiej. Aplikacja ta nie posiada wielu funkcjonalności, które cechują kompletny produkt – m.in. komunikacja z użytkownikiem z wykorzystaniem email(np: odzyskiwanie hasła, wysylanie newsletter’a), dane dotyczące ogłoszeń są bardzo podstawowe, nie jest obslugiwany kontakt ze sprzedającym itp.

2. Wymagania systemowe i funkcjonalne

Wymagania funkcjonalne:

- Rejestracja użytkowników oraz autoryzacja/autentyfikacja
- Przegląd aktualnych ofert oraz filtrowanie według kryteriów: cena, marka samochodu.
- Dodawanie ofert samochodów, wraz z informacjami:
  1. Nazwa oferty
  2. Opis
  3. Marka samochodu
  4. Cena
  5. Zdjęcia
- Funkcjnalność koszyka
- Funkcjonalność płatności z wykorzystaniem serwisu PayPal
- Przegląd historii płatności

Architektura:
Aplikacja składa się z dwóch warstw: części klienckiej - web frontend oraz backendu łączącego się bazą danych. Komunikacja odbywa się poprzez protokół HTTP z wykorzystaniem REST API, a dane przesyłane są w formacie JSON.

Stack technologiczny:
1. Frontend – React
2. Backend – Node.js + Express, MongoDB

3. Analiza zagadnienia i jego modelowanie.

3.1. Modele danych.

Dane w systemie są prezentowane pod postacią 3 modeli, przechowywanych w bazie danych.

3.1.1. User – zawiera podstawowe informacje o zarejestrowanym użytkowniku oraz zawartość jego koszyka, historię płatności.

- Id: string – id użytkownika
- Role: integer – rola użytkownika
- Email: string – email
- Password: string – zaszyfrowane hasło
- Name: string – imię i nazwisko użytkownika
- Token: string – token JWT służący do autoryzacji
- TokenExp: integer – data wygaśnięcia tokenu w formacie timestamp
- Cart: array – tablica zawierająca dane o koszyku użytkownika
  - Id: string – id przedmiotu
- History: array – tablica zawierająca historię płatności użytkownika
  - Id: string – id produktu
  - DateOfPurchase: integer - data płatności w formacie timestamp
  - Name: string – nazwa ogłoszenia
  - Price: double – cena
  - PaymentId: string – id płatności wygenerowany przez serwis PayPal

3.1.2. Product – zawiera informacje o ogłoszeniu

- Price: double – cena
- Images: string[] - adresy zdjęć do ogłoszenia – ścieżka względna na serwerze
- Manufacturer: integer – marka auta
- Title: string – opis ogłoszenia
- Description: string – opis ogłoszenia
- CreatedAt: datetime – data dodania ogłoszenia

3.1.3. Payment – zawiera informacje o dokonanych płatnościach

- User: object – dane użytkownika dokonującego płatności
  - Id: string – id użytkownika
  - Name: string – imię i nazwisko użytkownika
  - Email: string – email użytkownika
- Product: object – tablica zawierająca dane zakupionych produktów
  - Id: string – id produktu
  - Name: string – nazwa ogłoszenia
  - Price: double – cena
  - PaymentId: string – id płatności wygenerowany przez serwis PayPal
- CreatedAt: datetime – data dokonania płatności

4. Implementacja.

1. Frontend

Frontend został stworzony z wykorzystaniem biblioteki React. Stan aplikacji jest przechowywany za pomocą React Hooks oraz React Redux. Dodatkowo wykorzystana została biblioteka AntDesign zawierająca wiele gotowych, ostylowanych z wykorzystaniem CSS komponentów.

Komunikacja z backendem zachodzi z wykorzystaniem biblioteki Axios korzystającej z XMLHttpRequests.

Do dokonywania płatności wykorzystano komponent react-paypal-express-checkout, który po kliknięciu w przycisk przekierowywuje użytkownika na stronę usługi PayPal, a po zakończeniu operacji z powrotem na stronę.

2. Backend

Backend został stworzony z wykorzystaniem Node.js+Express, natomiast dane przechowuje w nierelacyjnej bazie danych MongoDB. Dodatkową biblioteką wykorzystaną w tej warstwie jest Mongoose, która odpowiada za połączenie z bazą danych. Do przechowywania zdjęć aplikacja wykorzystuje bibliotekę multer, która umożliwia zapis zdjęć w katalogu na urządzeniu. Autoryzacja odbywa się za pomocą ciasteczek.

Obsługiwane akcje REST API:

1. /users
    - GET /auth – sprawdzenie
    - POST /register – rejestracja użytkownika z danymi podanymi w formularzu
    - POST /login – logowanie użytkownika, po walidacji zakończonej sukcesem w ciasteczku zostaje przechowywany wygenerowany token JWT
    - POST /logout – wylogowanie użytkownika, usunięcie tokena
    - POST /addToCart – dodanie przedmiotu do koszyka użytkownika
    - DELETE /removeFromCart – usunięcie przedmiotu z koszyka użytkownika
    - GET /userCartInfo – pobranie stanu koszyka użytkownika
    - POST /successBuy – akcja służąca do tworzenia historii płatności
    - GET /getHistory – pobranie historii płatności użytkownika
2. /product
    - POST /uploadImage – dodanie zdjęć do ogłoszenia
    - POST /uploadProduct – dodanie ogłoszenia
    - GET /getProducts – pobranie listy produktów
    - GET /products_by_id – pobranie szczegółów produktu