

Uniwersytet Rzeszowski
Wydział Nauk Ścisłych i Technicznych
Instytut Informatyki



Dawid Olko i Piotr Smoła
Nr albumu studenta do125148 ps125162

Aplikacja sklepu internetowego Vue.js + Firebase

Praca projektowa z nierelacyjnych baz danych

Prowadzący: dr. Bober

Rzeszów r. a. 2024/2025

Spis treści

| | | |
|------------|--|-----------|
| 1. | Wprowadzenie..... | 3 |
| 2. | Przegląd Projektu | 4 |
| 3. | Struktura i Architektura Projektu..... | 6 |
| 3.1 | Ogólna Architektura..... | 6 |
| 3.2 | Struktura Katalogów | 7 |
| 3.3 | Warstwa Frontendu..... | 8 |
| 3.4 | Warstwa Bazy Danych (Firebase Firestore)..... | 8 |
| 4. | Integracja z Firebase | 9 |
| 4.1 | Konfiguracja i Inicjalizacja | 9 |
| 4.2 | Operacje na Kolekcjach Firestore..... | 10 |
| 4.3 | Reguły Bezpieczeństwa i Autoryzacja..... | 11 |
| 5. | Interfejs Użytkownika (GUI)..... | 12 |
| 5.1 | Widok Strony Głównej (Home) | 12 |
| 5.2 | Szczegóły Produktu (Product Detail) | 13 |
| 5.3 | Widok Koszyka (Cart)..... | 13 |
| 5.4 | Panel Użytkownika (User Panel) | 14 |
| 5.5 | Panel Administratora (Admin Panel) | 14 |
| 5.6 | Stopka (Footer)..... | 14 |
| 6. | Instalacja i Konfiguracja | 15 |
| 6.1 | Wymagania Wstępne | 15 |
| 6.2 | Klonowanie Repozytorium..... | 15 |
| 6.3 | Instalacja Zależności | 15 |
| 6.4 | Konfiguracja Firebase | 15 |
| 6.5 | Konfiguracja Kolekcji w Firestore..... | 16 |
| 6.6 | Uruchomienie Aplikacji..... | 16 |
| 7. | Podsumowanie i Wnioski | 17 |

1. Wprowadzenie

Projekt **Shop-Vue-Firestore-Demo** został stworzony w ramach przedmiotu baz nierelacyjnych. Celem projektu było zastosowanie dokumentowego modelu danych, wykorzystując Firebase Firestore jako główny system przechowywania informacji. Do budowy interfejsu użytkownika wybrano **Vue 3**, co umożliwiło stworzenie nowoczesnego, responsywnego i przyjaznego użytkownikowi frontendu.

W ramach projektu zrealizowano sklep internetowy, który obejmuje pełen zakres funkcji e-commerce, takich jak:

- **Zarządzanie produktami:**
 - Administrator może dodawać, edytować i usuwać produkty.
 - Produkty są przechowywane jako dokumenty w kolekcji Firestore, co pozwala na elastyczne zarządzanie danymi oraz skalowalność rozwiązania.
- **Obsługa koszyka i procesu zakupowego:**
 - Użytkownicy (zarówno zalogowani, jak i goście) mogą dodawać produkty do koszyka.
 - Proces zakupu obejmuje wprowadzenie danych wysyłki, faktury i płatności, a wszystkie informacje są przechowywane w Firestore.
- **Panele użytkownika i administratora:**
 - **Panel użytkownika:** Umożliwia przegląd zakupionych produktów pogrupowanych według daty zakupu oraz wyświetlanie szczegółów wysyłki.
 - **Panel administratora:** Obejmuje narzędzia do zarządzania produktami (w tym paginację, edycję i usuwanie) oraz przeglądanie wiadomości kontaktowych przesłanych przez formularz.
- **Newsletter i formularz kontaktowy:**
 - W stopce znajdują się formularze umożliwiające subskrypcję newslettera oraz wysyłanie wiadomości kontaktowych.
 - Dane te są zapisywane w dedykowanych kolekcjach Firestore, co umożliwia ich dalszą analizę i obsługę.

Front-end projektu został oparty na gotowym szablonie z W3Schools, co zapewnia estetyczny wygląd i pełną responsywność aplikacji.

Shop-Vue-Firestore-Demo to kompleksowy projekt sklepu internetowego, który demonstruje zastosowanie dokumentowego modelu danych w Firebase Firestore oraz nowoczesnych technologii webowych, takich jak Vue 3. Projekt stanowi doskonały przykład praktycznego wykorzystania baz nierelacyjnych w budowie skalowalnych aplikacji e-commerce.

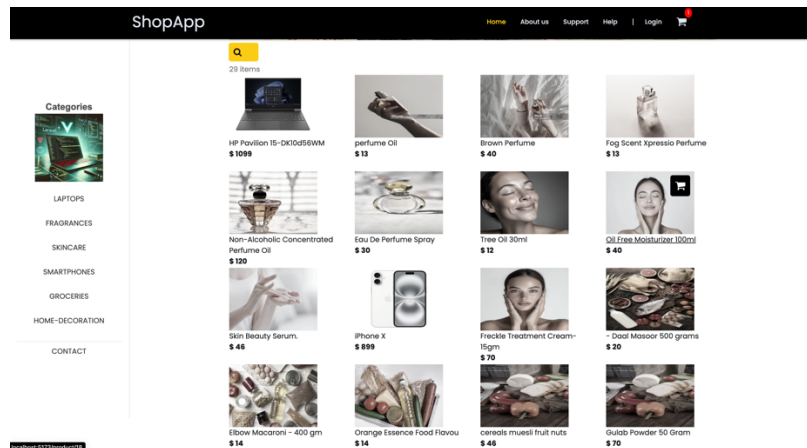
2. Przegląd Projektu

Projekt Shop-Vue-Firebase-Demo to nowoczesna aplikacja e-commerce, która wykorzystuje dokumentowy model danych oparty na Firebase Firestore. Aplikacja umożliwia użytkownikom przeglądanie produktów, dodawanie ich do koszyka, realizację zamówień oraz śledzenie zakupów, a także oferuje rozbudowany panel administracyjny dla administratorów.

Główne funkcjonalności projektu obejmują:

1. Frontend oparty na Vue 3 z gotowym szablonem W3Schools

Interfejs użytkownika został stworzony przy użyciu frameworka Vue 3, wykorzystując gotowy szablon W3Schools (W3Schools Clothing Store Template).

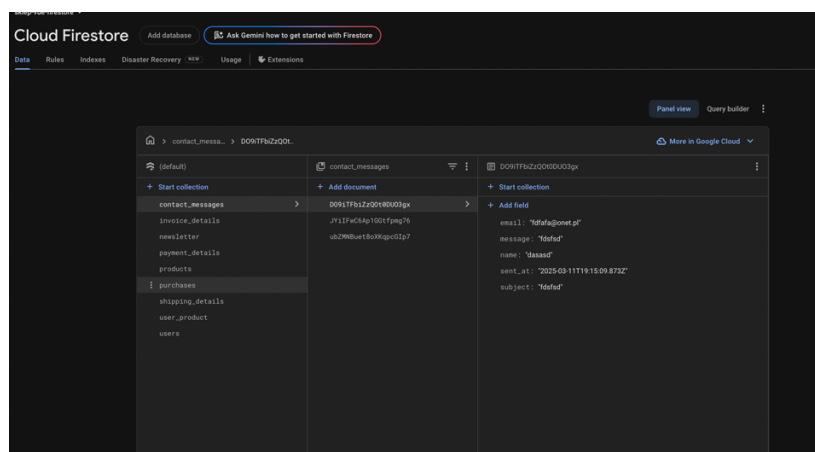


(zdjęcie widoku głównego aplikacji)

2. Integracja z Firebase Firestore

Wszystkie dane w projekcie przechowywane są w Firestore w formie dokumentów. W bazie znajdują się kolekcje takie jak:

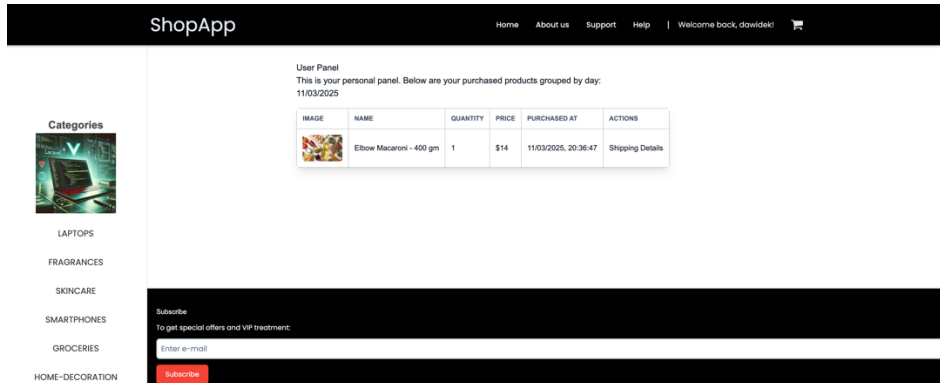
- o **products** – produkty dostępne w sklepie,
- o **purchases** – informacje o dokonanych zakupach,
- o **shipping_details, invoice_details, payment_details** – dane dotyczące procesu zakupowego,
- o **newsletter** – subskrypcje newslettera,
- o **contact_messages** – wiadomości kontaktowe przesłane przez użytkowników.



(zdjęcie przedstawiające strukturę bazy danych)

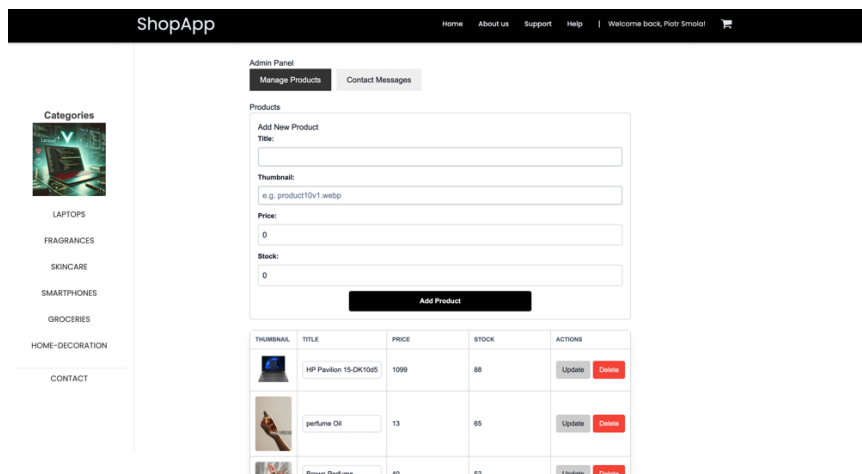
3. Panele Użytkownika i Administratora

- **Panel Użytkownika:** Użytkownik ma dostęp do panelu, w którym widzi swoje zakupy pogrupowane według daty. Po kliknięciu przycisku „Shipping Details” wyświetlane są szczegółowe informacje dotyczące wysyłki, pobierane z danych zamówienia lub z dokumentu invoice_details.



(zdjęcie przedstawiające panel użytkownika)

- **Panel Administratora:** Administrator (użytkownik z role_id równym 1) ma dostęp do panelu umożliwiającego zarządzanie produktami. W panelu admina można dodawać nowe produkty, edytować lub usuwać istniejące produkty (z paginacją – maksymalnie 10 produktów na stronę). Ponadto, panel administratora zawiera widok wiadomości kontaktowych przesłanych przez użytkowników za pomocą formularza kontaktowego.



(Tu ma być zdjęcie przedstawiające panel admina)

4. Proces

Zakupowy

Aplikacja umożliwia pełny proces zakupowy: użytkownik dodaje produkty do koszyka, przechodzi do checkoutu, gdzie wprowadza dane wysyłki, faktury i płatności. Wszystkie te informacje są zapisywane w Firestore, co pozwala na dokładne śledzenie zamówień i ułatwia zarządzanie danymi.

Projekt Shop-Vue-Firebase-Demo demonstruje, jak wykorzystać dokumentowy model danych w Firebase Firestore w połączeniu z nowoczesnym front-endem opartym na Vue 3. Aplikacja łączy funkcjonalności sklepu internetowego z elastycznym zarządzaniem danymi, co czyni ją doskonałym przykładem praktycznego zastosowania baz nierelacyjnych w e-commerce.

3. Struktura i Architektura Projektu

W rozdziale tym omówiono główne założenia architektoniczne aplikacji oraz przedstawiono strukturę katalogów w projekcie. Dzięki temu możliwe jest łatwiejsze zrozumienie sposobu, w jaki poszczególne elementy (frontend, baza danych w Firestore, komponenty Vue) współpracują ze sobą, tworząc spójną całość.

3.1 Ogólna Architektura

Aplikacja **Shop-Vue-Firebase-Demo** została zaprojektowana w modelu **SPA (Single Page Application)**, w którym:

- **Warstwa frontendu** oparta jest na frameworku **Vue 3**. Komponenty Vue zarządzają interfejsem użytkownika oraz logiką obsługi zdarzeń (np. kliknięcie przycisku „Dodaj do koszyka”).
- **Warstwa danych** znajduje się w chmurze – w bazie **Firebase Firestore**, wykorzystującej dokumentowy model danych.
- Komunikacja między Vue a Firestore odbywa się za pomocą oficjalnego SDK Firebase, co pozwala na łatwe wykonywanie operacji CRUD (Create, Read, Update, Delete) na kolekcjach dokumentów.

```
src > JS firebase.js > ...
1  // src/firebase.js
2  import { initializeApp } from "firebase/app";
3  import { getAnalytics } from "firebase/analytics";
4  import { getAuth } from "firebase/auth";
5  import { getFirestore } from "firebase/firestore";
6
7  // Konfiguracja Twojej aplikacji Firebase
8  const firebaseConfig = {
9    apiKey: "AIzaSyDpoR9FLkSQMaYpfrij-gXkmtwXfL4epjw",
10   authDomain: "sklep-vue-firestore.firebaseio.com",
11   projectId: "sklep-vue-firestore",
12   storageBucket: "sklep-vue-firestore.appspot.com",
13   messagingSenderId: "260406460193",
14   appId: "1:260406460193:web:cb9931fe9fdb9d96698860",
15   measurementId: "G-NJ6SPBZFVB",
16 };
17
18 // Inicjalizacja Firebase
19 const app = initializeApp(firebaseConfig);
20 const analytics = getAnalytics(app);
21 const auth = getAuth(app);
22 const db = getFirestore(app);
23
24 export { app, analytics, auth, db };
25
```

(diagram przedstawiający przepływ danych między Vue a Firestore)

3.2 Struktura Katalogów

Poniżej zaprezentowano przykładową strukturę katalogów w projekcie, wraz z krótkim opisem zawartości:

```
Shop-Vue-Firebase-Demo/
├── docs/
│   └── img/
│       └── app.png          // Zrzuty ekranu, dokumentacja, obrazy
pomocnicze
├── database/
│   ├── users.json
│   ├── products.json
│   ├── user_product.json
│   ├── purchases.json
│   ├── shipping_details.json
│   ├── invoice_details.json
│   ├── payment_details.json
│   ├── newsletter.json
│   └── contact_messages.json
├── public/
├── src/
│   ├── assets/
│   │   └── styles/          // Pliki CSS i zasoby graficzne
│   ├── components/
│   │   ├── Checkout.vue    // Komponent formularza zamówienia
│   │   └── ...              // Inne wielokrotnie używane komponenty
│   ├── views/
│   │   ├── HomeView.vue    // Widok strony głównej
│   │   ├── CartView.vue    // Widok koszyka
│   │   ├── ProductDetailView.vue // Szczegóły produktu
│   │   ├── CategoryView.vue
│   │   ├── BrandView.vue
│   │   └── Auth/
│   │       ├── LoginView.vue
│   │       ├── RegisterView.vue
│   │       ├── PanelView.vue // Panel użytkownika / admina
│   │       └── SettingsView.vue
│   │   └── ...
│   ├── firebase.js          // Konfiguracja Firebase i inicjalizacja
Firestore
├── router/
│   └── index.js              // Definicje tras (ścieżek) w aplikacji
Vue
├── stores/
│   └── auth.js                // Pinia Store zarządzający stanem
autoryzacji
├── App.vue                    // Główny komponent aplikacji
└── main.js                    // Punkt wejścia (bootstrap) aplikacji
Vue
├── package.json
├── vite.config.js
└── README.md
```

1. **docs/** – Katalog z dokumentacją, zrzutami ekranu i materiałami pomocniczymi.
2. **database/** – Przechowuje pliki JSON z przykładowymi danymi do importu w Firestore.
3. **src/** – Zawiera właściwy kod aplikacji Vue:
 - **assets/** – Pliki statyczne (CSS, grafiki).
 - **components/** – Komponenty wielokrotnego użytku, np. formularz zamówienia, nagłówek, stopka.
 - **views/** – Główne widoki aplikacji (strony), np. HomeView, CartView, PanelView.
 - **firebase.js** – Inicjalizacja Firebase i eksport obiektów auth oraz db.
 - **router/** – Definicje tras Vue (ścieżek URL do poszczególnych widoków).
 - **stores/** – Pliki Pinia Store, odpowiedzialne za zarządzanie stanem aplikacji, np. autoryzacja użytkowników.
 - **App.vue** i **main.js** – Główne pliki startowe aplikacji.

3.3 Warstwa Frontendu

Warstwa frontendu jest odpowiedzialna za:

- **Wyświetlanie danych** pobranych z Firestore (np. listy produktów, zawartości koszyka).
- **Obsługę interakcji użytkownika**, np. kliknięcia przycisków „Dodaj do koszyka” czy wypełnienie formularza rejestracyjnego.
- **Zarządzanie stanem** aplikacji (np. informacje o zalogowanym użytkowniku, lista produktów w koszyku) przy pomocy Pinia Store.

Dzięki wykorzystaniu frameworka Vue 3 możliwe jest komponowanie interfejsu w postaci modułowych komponentów oraz łatwe skalowanie projektu.

3.4 Warstwa Bazy Danych (Firestore)

Aplikacja wykorzystuje Firestore jako główne repozytorium danych w modelu dokumentowym:

- **Kolekcje** (np. products, purchases, newsletter) przechowują dokumenty, z których każdy może mieć złożone struktury danych (tablice, zagnieżdżone obiekty).
- **Operacje CRUD** (Create, Read, Update, Delete) realizowane są za pomocą oficjalnego SDK Firebase, co umożliwia bezpośrednie wywołania w kodzie Vue (np. addDoc, getDoc, updateDoc, deleteDoc).
- **Reguły bezpieczeństwa** w Firestore pozwalają kontrolować, kto ma dostęp do poszczególnych dokumentów i kolekcji.

Dzięki takiemu podejściu aplikacja może dynamicznie reagować na zmiany w bazie danych, zapewniając użytkownikom aktualne informacje (np. stan magazynowy produktu) bez konieczności odświeżania strony.

4. Integracja z Firebase

W tej części omówiono sposób, w jaki projekt **Shop-Vue-Firestore-Demo** został połączony z usługami Firebase, w szczególności z bazą danych Firestore. Dzięki temu aplikacja może w prosty sposób zarządzać danymi, takimi jak produkty, zamówienia czy wiadomości kontaktowe, korzystając z modelu dokumentowego.

4.1 Konfiguracja i Inicjalizacja

Podstawowa konfiguracja Firebase odbywa się w pliku (na przykład) `src/firebase.js`, w którym importowane są odpowiednie moduły z pakietu **firebase**. Następnie wywołuje się funkcję `initializeApp` z obiektem konfiguracji pobranym z konsoli Firebase. Poniżej znajduje się przykładowy fragment kodu:

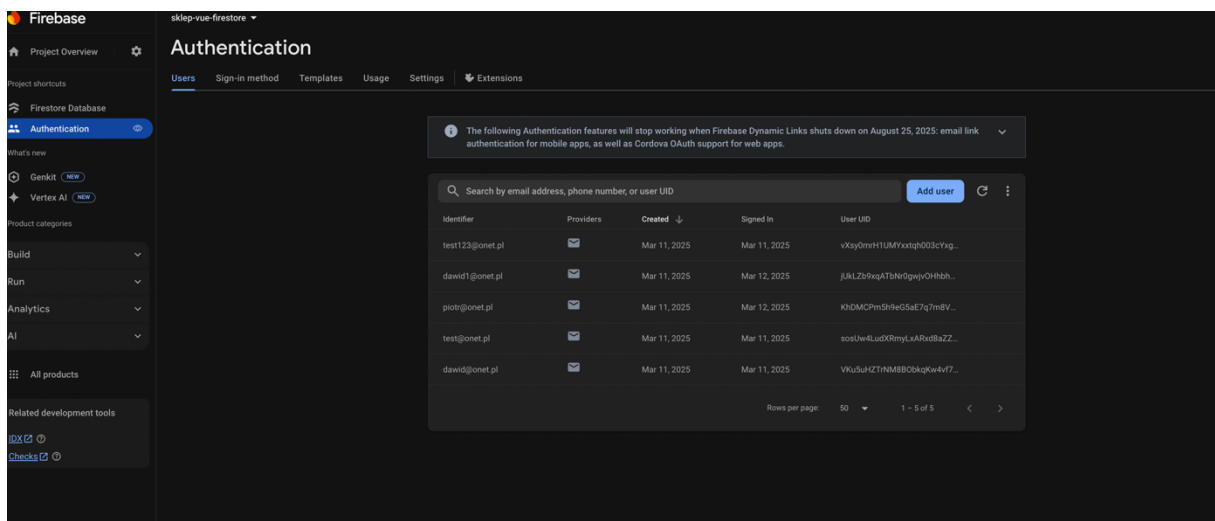
```
import { initializeApp } from "firebase/app";
import { getFirestore } from "firebase/firestore";
import { getAuth } from "firebase/auth";

const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_AUTH_DOMAIN",
  projectId: "YOUR_PROJECT_ID",
  // pozostałe pola konfiguracyjne...
};

// Inicjalizacja aplikacji Firebase
const app = initializeApp(firebaseConfig);

// Eksport obiektów do użycia w innych częściach projektu
export const db = getFirestore(app);
export const auth = getAuth(app);
```

Dzięki temu w dowolnym miejscu projektu można zaimportować `db` (Firestore) lub `auth` (Firebase Authentication) i wykonywać operacje związane z bazą danych lub logowaniem użytkowników.



(Zrzut ekranu z konsoli Firebase, pokazujący projekt i dane konfiguracyjne)

4.2 Operacje na Kolekcjach Firestore

W projekcie zastosowano **oficjalne SDK Firebase** do komunikacji z Firestore. Przykładowo, w panelu administratora:

- **Dodawanie produktu** realizowane jest przez wywołanie `addDoc` na kolekcji `products`, przekazując obiekt z polami takimi jak `title`, `price` czy `stock`.
- **Edycja produktu** wykorzystuje `updateDoc`, gdzie należy podać referencję do dokumentu (np. `doc(db, "products", prod.id)`) i obiekt z nowymi wartościami pól.
- **Usuwanie produktu** polega na wywołaniu `deleteDoc` z analogiczną referencją do dokumentu.

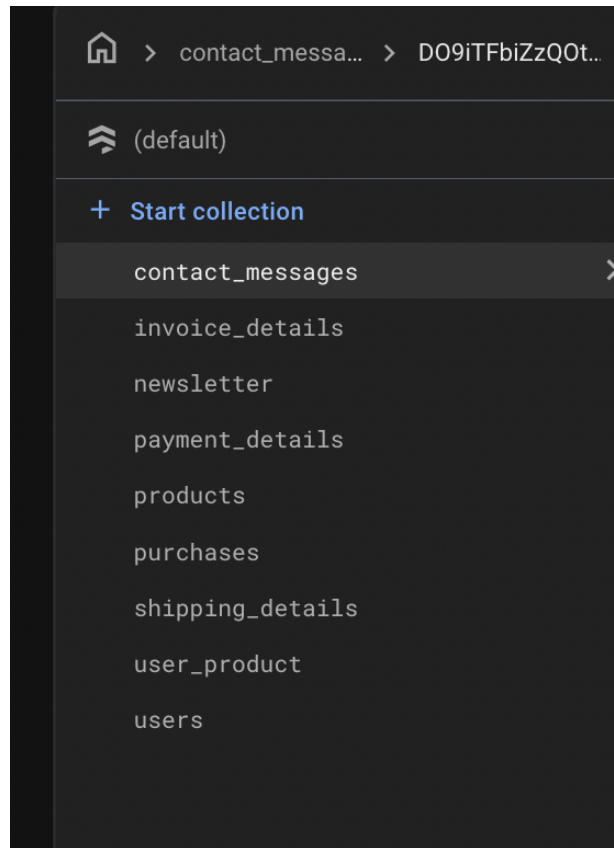
Poniższy fragment kodu obrazuje wywołanie `updateDoc`:

```
import { doc, updateDoc } from "firebase/firestore";
import { db } from "@firebase";

async function handleUpdateProduct(prod) {
  // Walidacja (np. czy cena >= 0 i stock >= 0)
  if (prod.price < 0 || prod.stock < 0) {
    alert("Price and stock must be 0 or greater.");
    return;
  }

  await updateDoc(doc(db, "products", prod.id), {
    title: prod.title,
    price: prod.price,
    stock: prod.stock,
    // inne pola...
  });
  alert("Product updated.");
}
```

Podobnie przebiegają operacje w innych częściach aplikacji, np. w panelu użytkownika (pobieranie listy zamówień) czy w formularzu kontaktowym (zapisywanie wiadomości do kolekcji `contact_messages`).



(Zrzut ekranu z Firestore w konsoli Firebase, prezentujący struktury kolekcji products, purchases, newsletter, itp.)

4.3 Reguły Bezpieczeństwa i Autoryzacja

W projekcie, w środowisku deweloperskim, można ustawić dość otwarte reguły bezpieczeństwa, aby ułatwić testowanie. W środowisku produkcyjnym należy jednak odpowiednio skonfigurować reguły Firestore, by chronić dane przed nieautoryzowanym dostępem (np. umożliwić modyfikację produktów tylko użytkownikowi z rolą administratora).

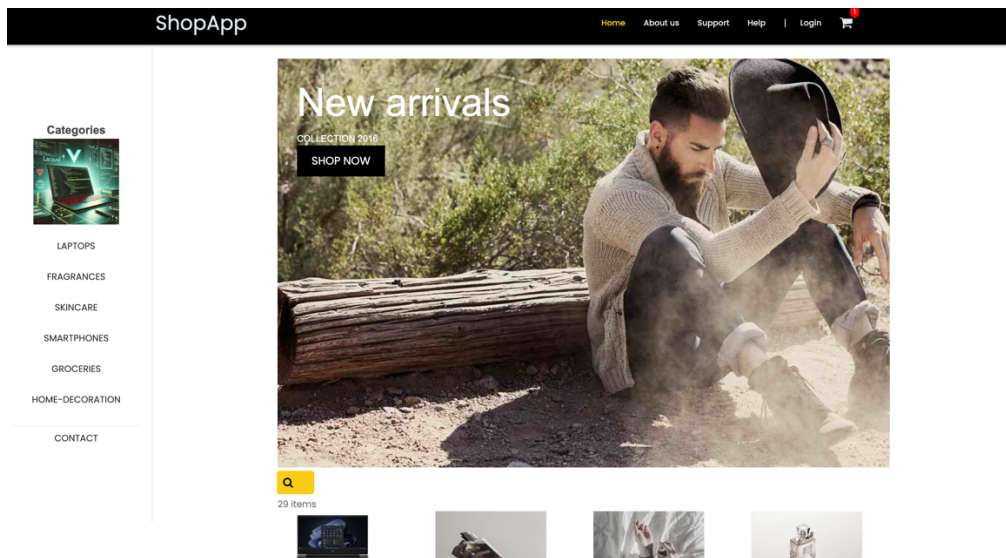
Zastosowanie modelu dokumentowego w Firestore upraszcza zarządzanie danymi – każdy produkt, zamówienie czy wiadomość kontaktowa to odrębny dokument, co pozwala na łatwe skalowanie i rozdzielenie odpowiedzialności pomiędzy różne kolekcje.

5. Interfejs Użytkownika (GUI)

W tej części zostanie przedstawiony wygląd aplikacji **Shop-Vue-Firestore-Demo** od strony wizualnej. Interfejs użytkownika bazuje na gotowym szablonie z W3Schools, dzięki czemu uzyskano nowoczesny design oraz pełną responsywność na urządzeniach mobilnych i desktopowych.

5.1 Widok Strony Głównej (Home)

Po uruchomieniu aplikacji użytkownik trafia na stronę główną, gdzie prezentowane są przykładowe produkty i przyciski umożliwiające przejście do dalszych sekcji sklepu.

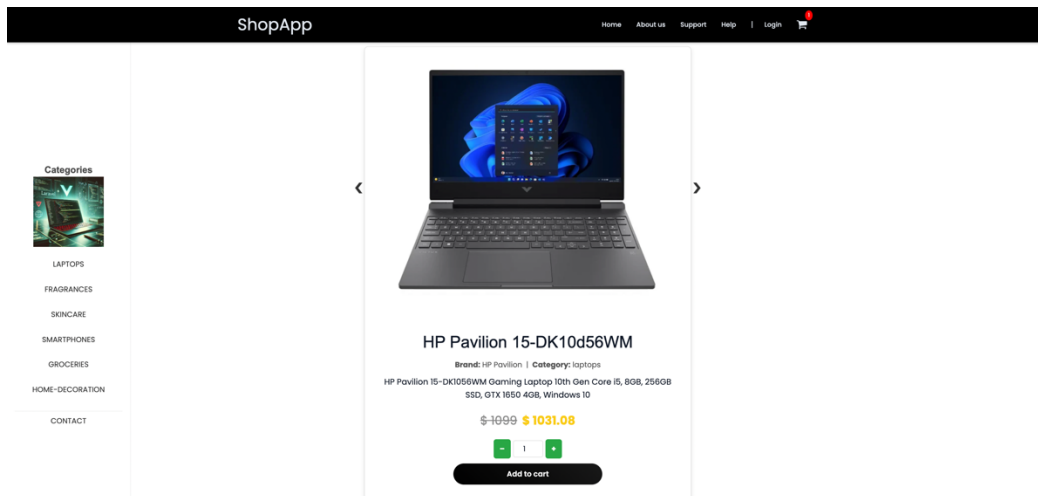


(Zrzut ekranu głównego widoku – home.png)

1. **Nagłówek (header)** – Zawiera logo, menu nawigacyjne oraz przycisk koszyka z licznikiem produktów.
2. **Baner lub sekcja promocyjna** – Może wyświetlać nowe kolekcje lub promocje.
3. **Lista produktów** – Użytkownik może przeglądać produkty, klikając w miniaturki lub przechodząc do szczegółów wybranego przedmiotu.

5.2 Szczegóły Produktu (Product Detail)

Kliknięcie na konkretny produkt przekierowuje do widoku szczegółów, w którym prezentowane są informacje takie jak nazwa, cena, dostępny stan magazynowy (stock) oraz galeria zdjęć.

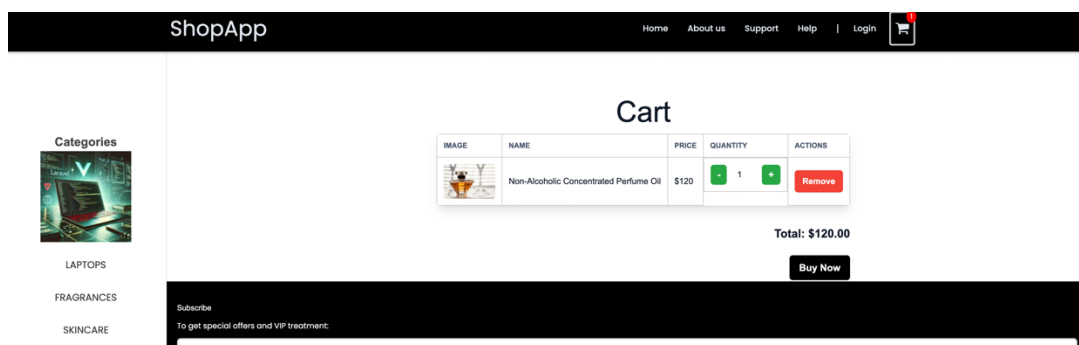


(Zrzut ekranu widoku szczegółów produktu – *product-detail.png*)

- **Zdjęcie produktu** – Główna miniaturka lub slider.
- **Opis i cena** – Informacje o produkcie, np. kategoria, marka, rabat.
- **Przycisk „Dodaj do koszyka”** – Umożliwia dodanie produktu do koszyka; w przypadku braku zalogowania można nadal dodać do koszyka jako gość.

5.3 Widok Koszyka (Cart)

Po kliknięciu w ikonę koszyka w nagłówku, użytkownik trafia do widoku koszyka, gdzie może przeglądać dodane produkty, zmieniać ich ilość, usuwać niechciane pozycje i przejść do checkoutu.



(Zrzut ekranu widoku koszyka – *cart.png*)

- **Tabela produktów** – Zawiera informacje o nazwie, cenie, liczbie sztuk oraz łącznej kwocie.
- **Podsumowanie zamówienia** – Wyświetla całkowitą cenę koszyka.
- **Przycisk „Buy Now”** – Przenosi do procesu checkout, gdzie użytkownik wypełnia dane wysyłki i płatności.

5.4 Panel Użytkownika (User Panel)

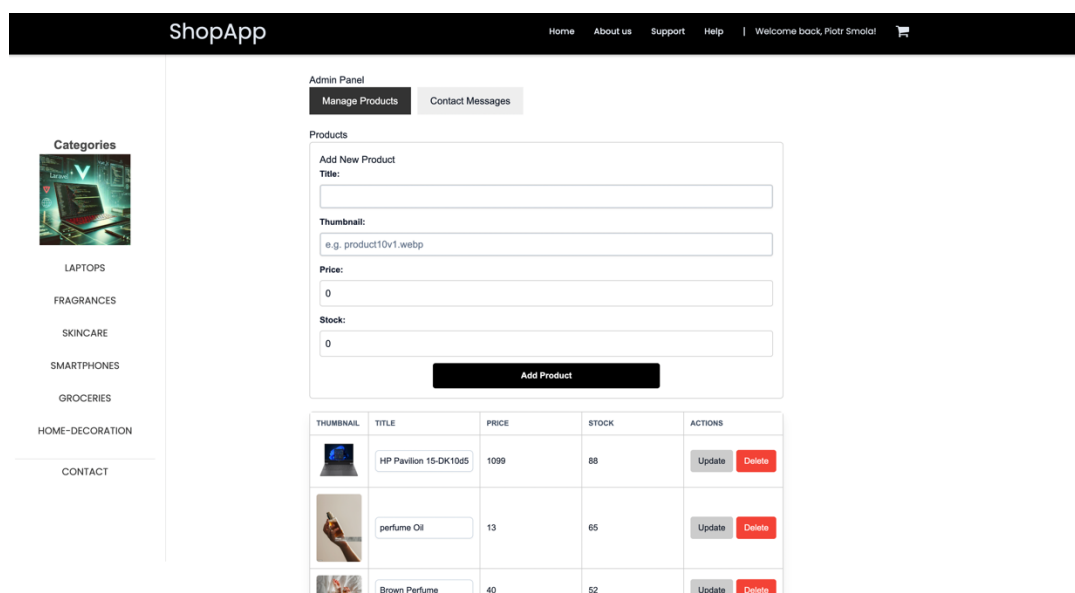
Po zalogowaniu się jako zwykły użytkownik ($\text{role_id} = 0$) dostępny jest panel z listą dokonanych zakupów. Produkty wyświetlane są w grupach według daty zakupu, a przycisk „Shipping Details” pozwala zobaczyć szczegółowe informacje o wysyłce.

(Tu ma być zrzut ekranu panelu użytkownika – np. *user-panel.png*)

- **Lista zamówień** – Każde zamówienie jest pogrupowane według daty.
- **Szczegóły wysyłki** – Pobierane z dokumentu zakupu (pole shipping) lub z kolekcji invoice_details (jeśli w zamówieniu nie ma takich informacji).

5.5 Panel Administratora (Admin Panel)

Po zalogowaniu się jako administrator ($\text{role_id} = 1$) w panelu pojawiają się dodatkowe zakładki, m.in. „Manage Products” i „Contact Messages”.



(Zrzut ekranu panelu administratora – *admin-panel.png*)

1. **Manage Products** – Pozwala dodać nowy produkt (wymagane pola: tytuł, miniaturka, cena, stan magazynowy). Edycja i usuwanie istniejących produktów są dostępne w formie tabeli z wierszami do edycji.
2. **Contact Messages** – Wyświetla listę wiadomości z formularza kontaktowego (np. imię, e-mail, treść, data wysłania).

5.6 Stopka (Footer)

Na dole każdej podstrony znajduje się stopka, w której użytkownik może:

- **Zapisać się do newslettera** – Wpisując adres e-mail w polu subskrypcji.
- **Wysłać wiadomość kontaktową** – Formularz do zgłaszania pytań lub problemów, zapisywany w kolekcji contact_messages.

6. Instalacja i Konfiguracja

W tym rozdziale przedstawiono kroki niezbędne do uruchomienia projektu **Shop-Vue-Firestore-Demo** na lokalnej maszynie. Aplikacja została zbudowana przy użyciu **Vue 3** oraz **Firebase Firestore**, dlatego konieczne jest skonfigurowanie środowiska Node.js oraz projektu w Firebase.

6.1 Wymagania Wstępne

1. **Node.js** – Wersja co najmniej 14+ (zalecane 16+).
2. **Konto w Firebase** – Dostęp do [Firebase Console](#), gdzie utworzony zostanie projekt i skonfigurowana baza Firestore.
3. **Menadżer pakietów** – np. **npm** lub **yarn**, w zależności od preferencji.

6.2 Klonowanie Repozytorium

Aby rozpocząć pracę, sklonuj repozytorium z GitHub (lub pobierz je jako archiwum ZIP i rozpakuj):

```
git clone https://github.com/dawidolko/Shop-Vue-Firestore-Demo.git
cd Shop-Vue-Firestore-Demo
```

6.3 Instalacja Zależności

W katalogu głównym projektu zainstaluj wszystkie niezbędne paczki:

```
npm install
lub
yarn install
```

Proces ten pobierze i zainstaluje biblioteki wymagane przez projekt, m.in. **Vue**, **Firebase**, **Pinia** czy **Vite**.

6.4 Konfiguracja Firebase

1. **Utwórz projekt w Firebase** – Wejdź na [Firebase Console](#) i załóż nowy projekt.
2. **Włącz Firestore** – W sekcji *Build* → *Firestore Database* utwórz bazę danych w trybie produkcyjnym lub testowym (w zależności od potrzeb).
3. **Skopiuj konfigurację** – Z zakładki *Ustawienia projektu* → *Konfiguracja aplikacji sieciowej* pobierz obiekt konfiguracyjny (apiKey, authDomain, projectId itp.).
4. **Umieść konfigurację w pliku** – W projekcie (np. w src/firebase.js) wklej dane konfiguracyjne, a następnie zainicjuj Firebase:

```
import { initializeApp } from "firebase/app";
import { getFirestore } from "firebase/firestore";
import { getAuth } from "firebase/auth";

const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_AUTH_DOMAIN",
  projectId: "YOUR_PROJECT_ID",
  // inne pola...
```

```
};  
const app = initializeApp(firebaseConfig);  
export const db = getFirestore(app);  
export const auth = getAuth(app);
```

5. **Reguły bezpieczeństwa** – Upewnij się, że w Firestore masz ustawione odpowiednie reguły dostępu. Dla testów można ustawić reguły umożliwiające otwarty dostęp, ale w środowisku produkcyjnym zaleca się restrykcyjne reguły, np. wymagające autoryzacji użytkownika z rolą administratora do edycji produktów.

6.5 Konfiguracja Kolekcji w Firestore

Projekt zakłada istnienie następujących kolekcji w Firestore:

- `products` – lista produktów (dokumenty z polami takimi jak `title`, `price`, `stock` itp.).
- `purchases` – dane o zakupach (można przechowywać w polu `items` tablicę obiektów opisujących zamówione produkty).
- `shipping_details`, `invoice_details`, `payment_details` – szczegółowe informacje związane z zamówieniami.
- `newsletter` – adresy e-mail subskrybentów newslettera.
- `contact_messages` – wiadomości kontaktowe przesłane przez formularz w stopce.

Możesz ręcznie utworzyć te kolekcje w Firestore lub skorzystać z plików JSON w katalogu `database` oraz skryptu `importData.js` (o ile jest dostępny w projekcie), aby automatycznie zaimportować przykładowe dane.

6.6 Uruchomienie Aplikacji

Po wykonaniu powyższych kroków możesz uruchomić projekt w trybie deweloperskim:

```
npm run dev
```

lub

```
yarn dev
```

Domyślnie aplikacja będzie dostępna pod adresem:

```
http://localhost:5173/
```

W oknie przeglądarki powinien pojawić się główny widok sklepu, z którego można przejść do panelu użytkownika (po zalogowaniu) lub panelu administratora (jeśli zalogowany użytkownik ma rolę admina).

7. Podsumowanie i Wnioski

Projekt **Shop-Vue-Firestore-Demo** prezentuje kompleksowe zastosowanie baz nierelacyjnych w kontekście aplikacji e-commerce. Dzięki wykorzystaniu **Firestore** możliwe było zademonstrowanie elastycznego i skalowalnego modelu dokumentowego, który ułatwia zarządzanie danymi takimi jak produkty, zamówienia czy wiadomości kontaktowe.

Zastosowanie frameworka **Vue 3** pozwoliło na stworzenie responsywnego i przyjaznego interfejsu użytkownika, opartego na gotowym szablonie W3Schools. Rozdzielenie warstwy frontendu (Vue) od warstwy danych (Firestore) zapewnia przejrzystą architekturę i łatwość wprowadzania modyfikacji.

W projekcie zaimplementowano między innymi:

- **Panel użytkownika** umożliwiający przegląd zakupów,
- **Panel administratora** oferujący zarządzanie produktami (z paginacją i walidacją danych) oraz wgląd w wiadomości kontaktowe,
- **Proces zakupowy** obejmujący koszyk, checkout i zapisywanie danych wysyłki, faktury i płatności w Firestore,
- **Funkcjonalność newslettera i formularza kontaktowego**, które zapisują dane do odpowiednich kolekcji.

Przedstawiony projekt może stanowić podstawę do dalszego rozwoju w kierunku bardziej zaawansowanych rozwiązań, takich jak integracja z bramkami płatności czy zaawansowana analityka danych w chmurze. W kontekście przedmiotu baz nierelacyjnych, projekt ten pokazuje, jak dokumentowy model danych sprawdza się w realnych aplikacjach, zapewniając dużą elastyczność przy zmianach schematu oraz szybki dostęp do danych w chmurze.

Dzięki temu przykładowi można lepiej zrozumieć zarówno specyfikę tworzenia aplikacji webowych z wykorzystaniem **Vue 3**, jak i praktyczne zastosowanie **Firestore** w roli bazy nierelacyjnej, co stanowi istotny aspekt rozwoju nowoczesnych aplikacji internetowych.