



# Firebase w praktyce

1

OD IDEI DO ZAAWANSOWANYCH  
ROZWIĄZAŃ W APLIKACJACH  
WEBOWYCH



# Wprowadzenie do prezentacji

Wraz z rozwojem aplikacji webowych i mobilnych coraz większą rolę odgrywają narzędzia, które umożliwiają sprawne zarządzanie danymi oraz infrastrukturą backendową. Jednym z takich narzędzi jest Firebase - innowacyjna platforma oferująca gotowe rozwiązania technologiczne do obsługi baz danych, uwierzytelniania użytkowników oraz hostingu aplikacji. Dzięki automatycznej synchronizacji informacji w czasie rzeczywistym oraz łatwej integracji z frameworkami frontendowymi, takimi jak Vue, Firebase znacząco upraszcza pracę programistów, umożliwiając im skupienie się na logice biznesowej i komforcie użytkownika. W efekcie powstają dynamiczne, wydajne i łatwo skalowalne aplikacje.





# CO TO JEST FIREBASE?

Firebase to kompleksowa platforma Backend-as-a-Service, umożliwiająca szybkie tworzenie, rozwijanie i skalowanie aplikacji webowych oraz mobilnych, bez konieczności zarządzania tradycyjną infrastrukturą serwerową. Platforma oferuje zaawansowane usługi, takie jak bazy danych w czasie rzeczywistym, Cloud Firestore, system uwierzytelniania, hosting, funkcje chmurowe oraz narzędzia analityczne, które pozwalają deweloperom skupić się na implementacji kluczowych funkcjonalności. Integracja z popularnymi frameworkami, takimi jak Vue, umożliwia budowanie dynamicznych interfejsów użytkownika oraz elastyczne modelowanie danych, co czyni Firebase idealnym rozwiązaniem dla projektów wymagających wysokiej skalowalności, bezpieczeństwa i wydajności.





# HISTORIA I GENEZA FIREBASE

Firebase rozpoczęło swoją drogę jako niezależny projekt, stworzony przez Andrew Lee oraz Jamesa Tamplina, z myślą o umożliwieniu deweloperom łatwiejszego zarządzania danymi w aplikacjach w czasie rzeczywistym. Początkowo skoncentrowane na automatycznej synchronizacji danych i uproszczeniu komunikacji między klientem a serwerem, rozwiązanie to szybko zyskało uznanie dzięki innowacyjnemu podejściu do backendu. Kluczowym momentem w rozwoju platformy było przejęcie Firebase przez Google w 2014 roku, co otworzyło nowy rozdział w rozwoju technologii - integrując ją z ekosystemem Google Cloud, rozszerzając funkcjonalności oraz wprowadzając zaawansowane narzędzia dla aplikacji webowych i mobilnych.

# Usługi

Firebase to kompleksowa platforma, która integruje szereg kluczowych usług wspierających rozwój nowoczesnych aplikacji. Wśród nich wyróżniamy:

- Bazy Danych:
  - Firebase Realtime Database oraz Cloud Firestore umożliwiają przechowywanie i synchronizację danych w czasie rzeczywistym, oferując elastyczny model NoSQL i skalowalność na miarę rosnących potrzeb aplikacji.
- Uwierzytelnianie:
  - Firebase Authentication zapewnia bezpieczne i wygodne zarządzanie kontami użytkowników, wspierając różnorodne metody logowania, od tradycyjnych haseł po integrację z mediani społecznościowymi.
- Przechowywanie Plików:
  - Firebase Storage to rozwiązanie dedykowane do skalowalnego przechowywania plików, umożliwiające łatwe zarządzanie zasobami multimedialnymi, dokumentami i innymi danymi.
- Funkcje Chmurowe:
  - Cloud Functions pozwala na uruchamianie kodu backendowego w odpowiedzi na zdarzenia, co umożliwia implementację logiki biznesowej bez konieczności zarządzania serwerem.
- Hosting i Analityka:
  - Firebase Hosting oferuje szybkie i bezpieczne wdrażanie aplikacji webowych, a Firebase Analytics dostarcza szczegółowe dane o zachowaniach użytkowników, wspierając ciągłą optymalizację aplikacji.

Dzięki integracji tych usług Firebase umożliwia deweloperom tworzenie skalowalnych, bezpiecznych i wydajnych aplikacji, eliminując potrzebę zarządzania tradycyjną infrastrukturą serwerową.

# FIREBASE JAKO ROZWIAZANIE DLA DEVELOPERÓW

Firebase stanowi idealne rozwiązanie dla deweloperów, umożliwiając im skupienie się na tworzeniu innowacyjnych funkcji aplikacji, zamiast martwić się o zarządzanie infrastrukturą backendową. Dzięki gotowym narzędziom, automatycznej synchronizacji danych w czasie rzeczywistym i elastycznej architekturze chmurowej, umożliwia szybkie prototypowanie oraz łatwe skalowanie aplikacji. Integracja z popularnymi frameworkami, takimi jak Vue, sprawia, że budowanie nowoczesnych interfejsów użytkownika staje się proste, a zaawansowane mechanizmy bezpieczeństwa gwarantują ochronę danych.



# Kluczowe komponenty Firebase

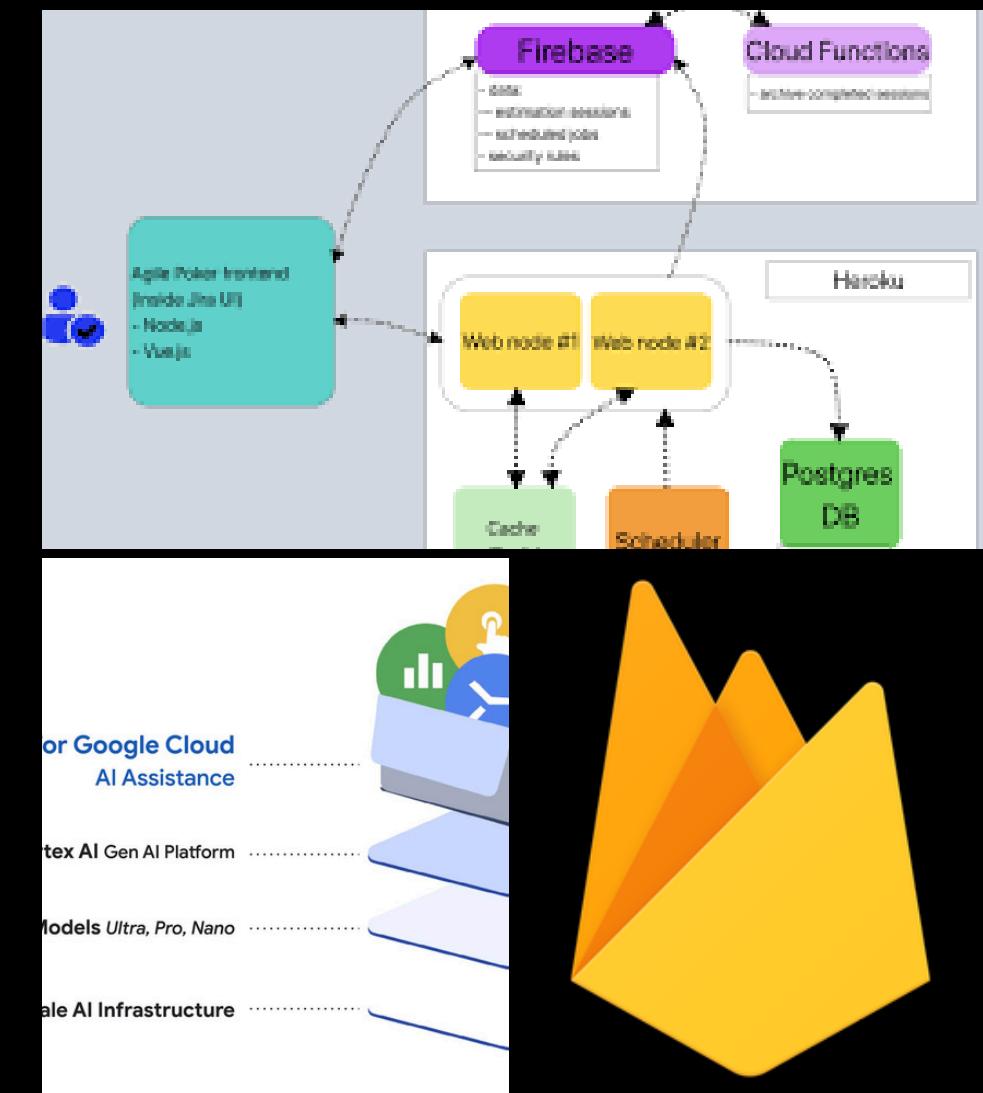
Firebase to platforma, która integruje szereg kluczowych komponentów umożliwiających sprawne zarządzanie danymi oraz logiką backendową. Realtime Database i Cloud Firestore zapewniają elastyczne przechowywanie oraz synchronizację informacji, natomiast Firebase Authentication dba o bezpieczny dostęp użytkowników. Cloud Functions umożliwia uruchamianie kodu backendowego w odpowiedzi na zdarzenia, a Firebase Storage zajmuje się przechowywaniem plików. Te elementy współpracują, tworząc spójne, skalowalne i efektywne środowisko do budowania nowoczesnych aplikacji.

# FIREBASE REALTIME DATABASE VS. CLOUD FIRESTORE

Firebase Realtime Database to rozwiązanie oparte na strukturze drzewa JSON, które umożliwia natychmiastową synchronizację danych w czasie rzeczywistym. Dzięki temu idealnie sprawdza się w aplikacjach wymagających błyskawicznych aktualizacji interfejsu. Z kolei Cloud Firestore wykorzystuje model dokumentowy, organizując dane w kolekcjach i dokumentach. Ta struktura nie tylko ułatwia modelowanie złożonych danych, ale także oferuje zaawansowane możliwości zapytań i lepszą skalowalność. Firestore zapewnia bardziej elastyczne operacje na danych, wspiera transakcje oraz zapewnia silniejsze mechanizmy bezpieczeństwa i offline support. W praktyce wybór między Realtime Database a Cloud Firestore zależy od specyfiki projektu - przy prostych strukturach i wymaganiach dotyczących natychmiastowej synchronizacji Realtime Database może być wystarczający, natomiast dla bardziej złożonych aplikacji Firestore oferuje szerszy zakres funkcjonalności i lepszą wydajność przy rosnącej skali.

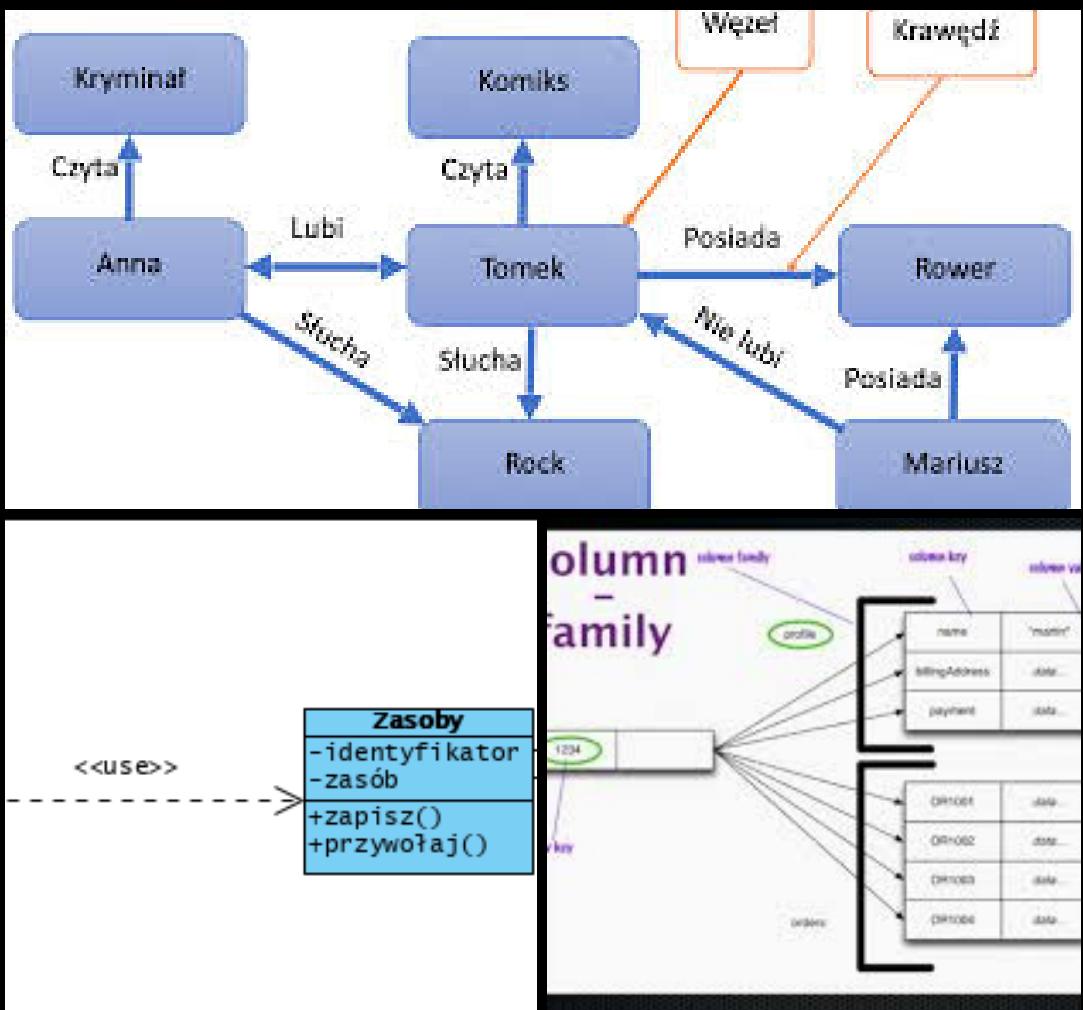


Cloud Firestore wyróżnia się nowoczesną, hierarchiczną strukturą, w której dane organizowane są w dokumentach i kolekcjach. Każdy dokument stanowi zbiór par klucz-wartość i może zawierać zagnieżdżone podkolekcje, co umożliwia intuicyjne modelowanie skomplikowanych struktur danych. Automatyczne indeksowanie wszystkich pól gwarantuje szybkie i efektywne zapytania, nawet przy bardzo rozbudowanych bazach. Dzięki wsparciu dla transakcji ACID, Firestore zapewnia wysoką spójność i bezpieczeństwo danych, a mechanizmy synchronizacji w czasie rzeczywistym oraz tryb offline umożliwiają płynną pracę aplikacji niezależnie od warunków sieciowych. Ta elastyczna i skalowalna architektura czyni Cloud Firestore idealnym rozwiązaniem dla nowoczesnych aplikacji wymagających dynamicznego zarządzania danymi.



## ARCHITEKTURA CLOUD FIRESTORE

# 10



Modelowanie danych w bazach NoSQL to podejście, które kładzie nacisk na elastyczność i skalowalność. W przeciwieństwie do tradycyjnych relacyjnych baz, gdzie dane są ściśle zorganizowane w tabele, w systemach NoSQL stosuje się zazwyczaj struktury dokumentowe, umożliwiające hierarchiczne, zagnieżdżone przechowywanie informacji. Taki model pozwala na szybszy dostęp do danych i lepszą optymalizację zapytań, dzięki czemu możliwe jest dostosowywanie struktury do dynamicznych wymagań aplikacji. Denormalizacja, czyli celowe duplikowanie danych tam, gdzie przyśpieszy to operacje odczytu, jest częstym rozwiązaniem w tego typu bazach. Podejście to wymaga przemyślanej architektury, która uwzględnia specyfikę zapytań oraz sposób, w jaki aplikacja korzysta z przechowywanych informacji, co pozwala na osiągnięcie wysokiej wydajności i elastyczności w zarządzaniu danymi.



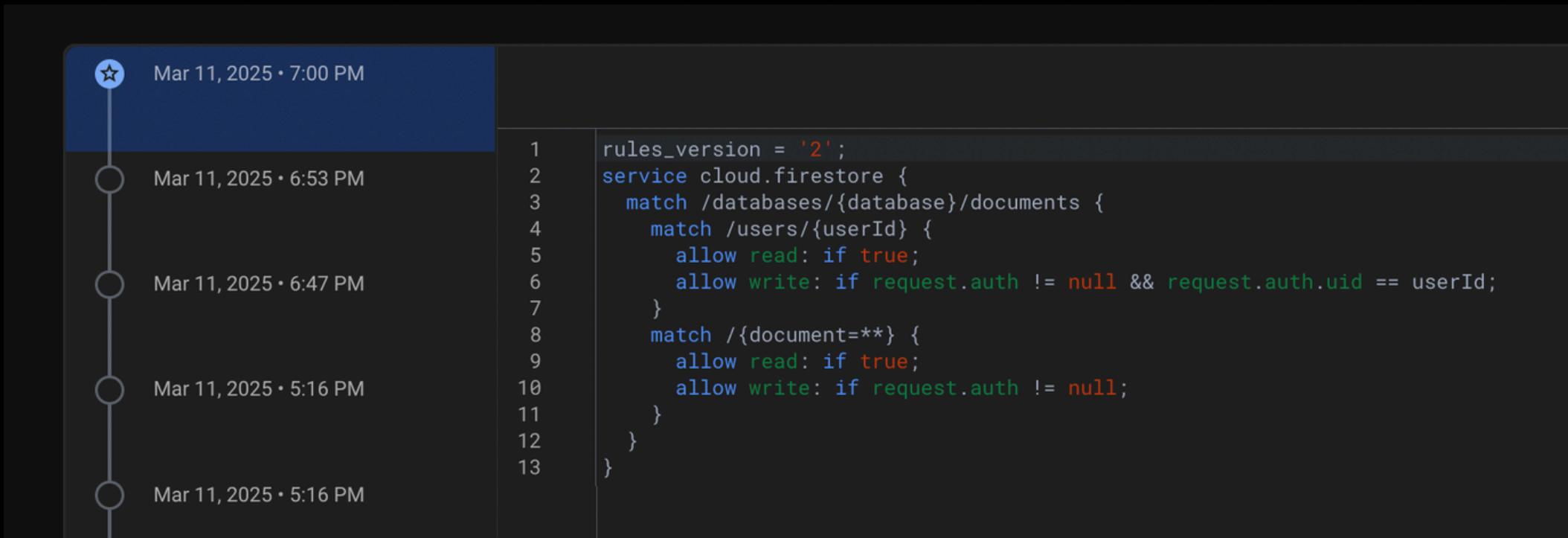
Firebase wyróżnia się szeregiem korzyści, które znacząco ułatwiają rozwój nowoczesnych aplikacji. Platforma umożliwia natychmiastową synchronizację danych oraz automatyczne skalowanie, co skraca czas wdrażania i upraszcza zarządzanie backendem. Dzięki zintegrowanym narzędziom, takim jak uwierzytelnianie, przechowywanie plików czy funkcje chmurowe, deweloperzy mogą skupić się na tworzeniu innowacyjnych rozwiązań, mając pewność, że kluczowe mechanizmy bezpieczeństwa i niezawodności są solidnie zabezpieczone. Dodatkowo, Firebase doskonale współpracuje z nowoczesnymi frameworkami, co umożliwia szybkie i elastyczne tworzenie interfejsów użytkownika, idealnych dla dynamicznych i responsywnych aplikacji.

## ZALETY KORZYSTANIA Z FIREBASE

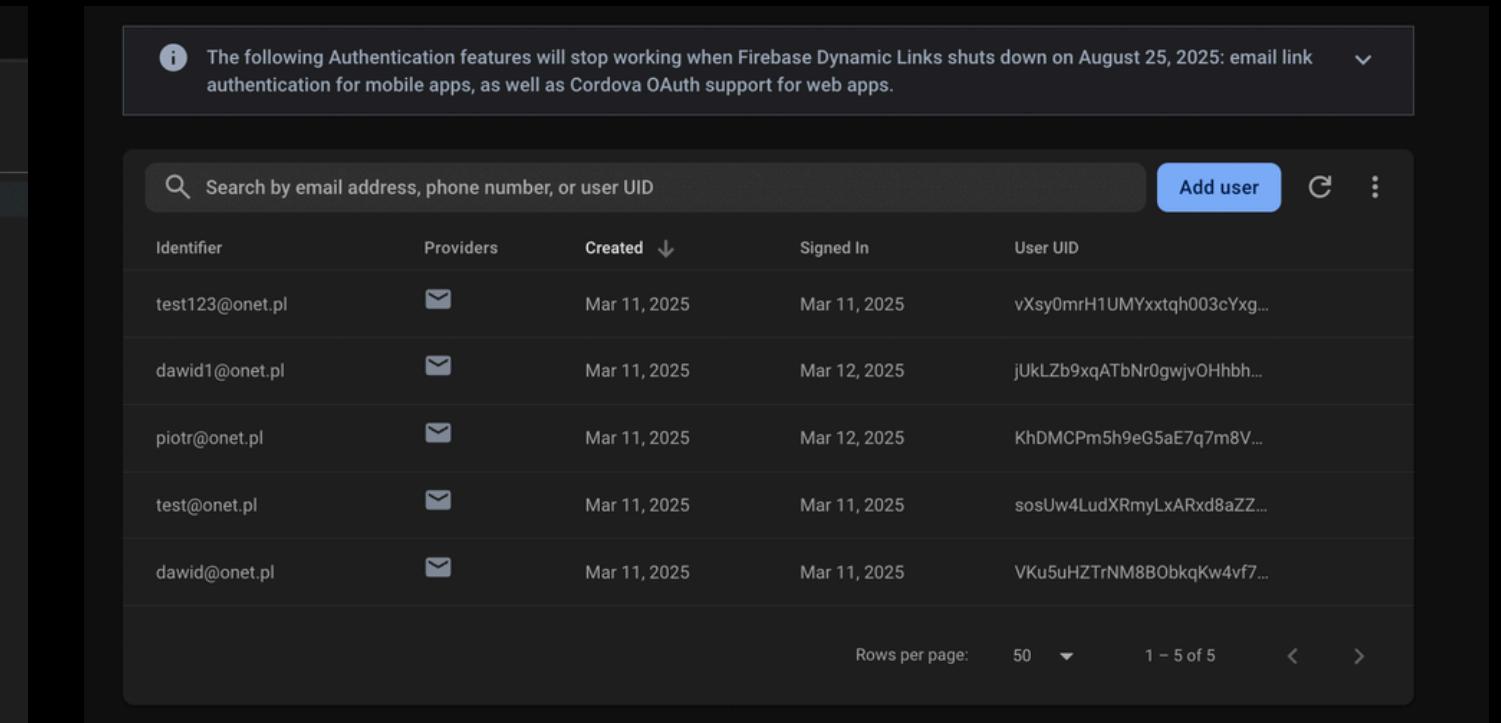
## PIERWSZE KROKI Z FIREBASE – KONFIGURACJA PROJEKTU

12

Konfiguracja projektu w Firebase polega na utworzeniu nowego projektu w konsoli Firebase, pobraniu pliku konfiguracyjnego oraz integracji kluczy API i identyfikatora projektu z aplikacją. Proces rozpoczyna się od logowania do konsoli, gdzie wybieramy opcję utworzenia nowego projektu. Kreator prowadzi przez kolejne etapy, umożliwiając precyzyjne ustawienie parametrów potrzebnych do prawidłowej synchronizacji z usługami Firebase, takimi jak baza danych, uwierzytelnianie czy funkcje chmurowe.

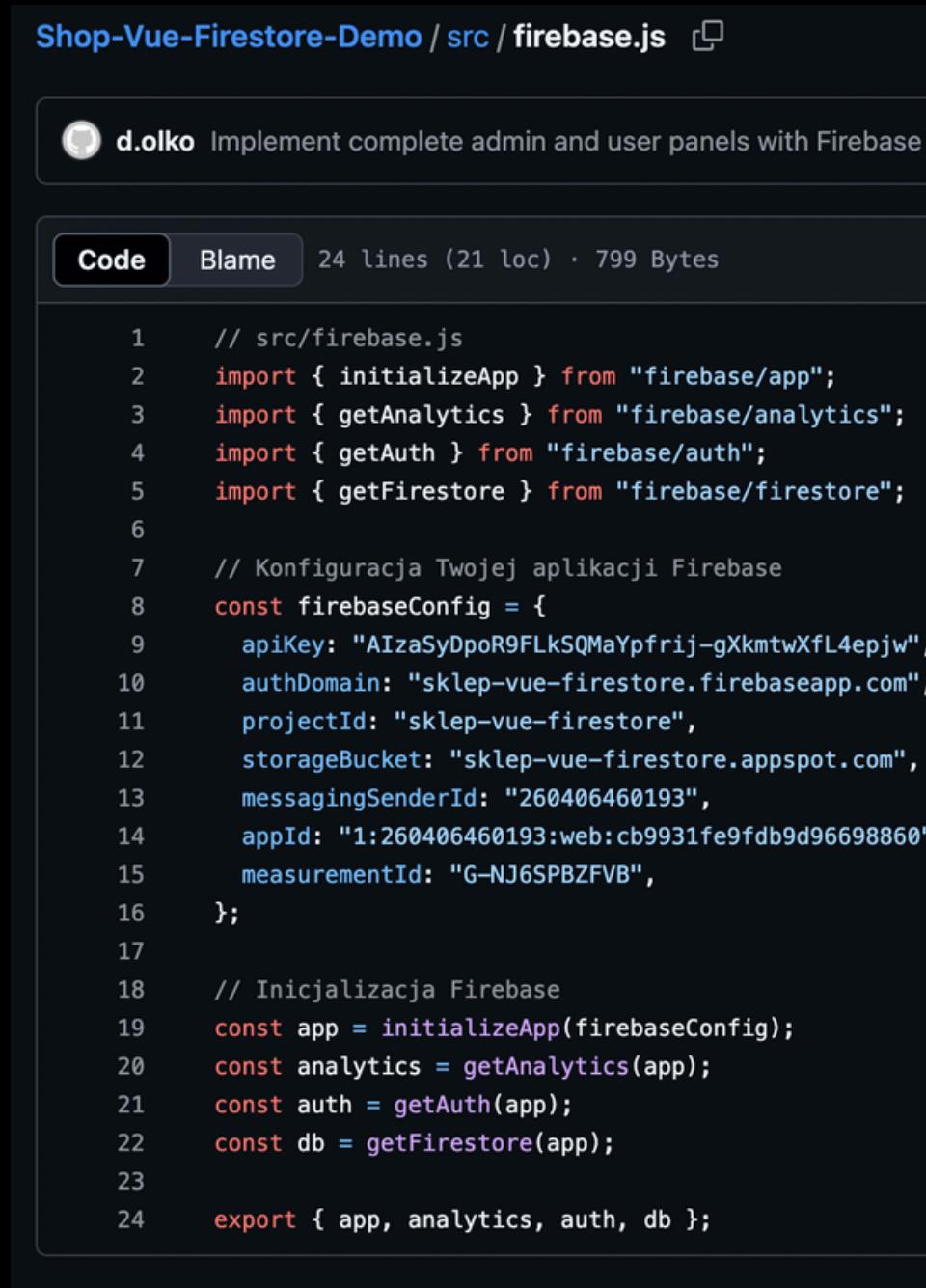


```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read: if true;
      allow write: if request.auth != null && request.auth.uid == userId;
    }
    match /{document=**} {
      allow read: if true;
      allow write: if request.auth != null;
    }
  }
}
```



Identifier	Providers	Created	Signed In	User UID
test123@onet.pl		Mar 11, 2025	Mar 11, 2025	vXsy0mrH1UMYxtqh003cYxg...
dawid1@onet.pl		Mar 11, 2025	Mar 12, 2025	jUkLZb9xqATbNr0gwjv0Hhbh...
piotr@onet.pl		Mar 11, 2025	Mar 12, 2025	KhDMCPm5h9eG5aE7q7m8V...
test@onet.pl		Mar 11, 2025	Mar 11, 2025	sosUw4LudXRmyLxARxd8aZZ...
dawid@onet.pl		Mar 11, 2025	Mar 11, 2025	VKu5uHZTrNM8B0bkqKw4vf7...

# INTEGRACJA FIREBASE Z APLIKACJĄ FRONTENDOWĄ



The screenshot shows a GitHub code editor interface for a repository named 'Shop-Vue-Firestore-Demo'. The file being viewed is 'src/firebase.js'. The code is written in JavaScript and imports various Firebase modules: 'firebase/app', 'firebase/analytics', 'firebase/auth', and 'firebase/firestore'. It then defines a 'firebaseConfig' object with specific API keys and configurations for the application. Finally, it initializes the app and retrieves references to the analytics, auth, and firestore services.

```
// src.firebaseio.js
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";

// Konfiguracja Twojej aplikacji Firebase
const firebaseConfig = {
  apiKey: "AIzaSyDpoR9FLkSQMaYpfrij-gXkmtwXfL4epjw",
  authDomain: "sklep-vue-firebase.firebaseio.com",
  projectId: "sklep-vue-firebase",
  storageBucket: "sklep-vue-firebase.appspot.com",
  messagingSenderId: "260406460193",
  appId: "1:260406460193:web:cb9931fe9fdb9d96698860",
  measurementId: "G-NJ6SPBZFVB",
};

// Inicjalizacja Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
const auth = getAuth(app);
const db = getFirestore(app);

export { app, analytics, auth, db };
```

Integracja Firebase z aplikacją frontendową polega na połączeniu usług platformy z kodem opartym na frameworku takim jak Vue, co umożliwia bezproblemową synchronizację danych i interakcję w czasie rzeczywistym. Proces integracji obejmuje instalację SDK Firebase, dodanie konfiguracji do projektu oraz wykorzystanie metod API, które pozwalają na obsługę bazy danych, uwierzytelnianie użytkowników i wywoływanie funkcji chmurowych. W efekcie aplikacja zyskuje dynamiczny interfejs, który reaguje na zmiany danych w czasie rzeczywistym, co podnosi jakość interakcji użytkownika i upraszcza rozwój zaawansowanych funkcjonalności.

# TWORZENIE KOLEKCJI I DOKUMENTÓW

The screenshot shows a GitHub repository interface. On the left, there's a sidebar with a tree view of files and folders. The main area displays the content of a file named `importData.js`. The code is written in JavaScript and uses the Firebase Admin SDK to import data from JSON files into Firestore collections.

```
Shop-Vue-Firestore-Demo / database / importData.js

d.olko Implement complete admin and user panels with Firebase Firestore inte... · 81 lines (73 loc) · 2.27 KB

Code Blame
1 import admin from "firebase-admin";
2 import fs from "fs";
3 import path from "path";
4 import { fileURLToPath } from "url";
5
6 // Ustal __dirname w ES module
7 const __filename = fileURLToPath(import.meta.url);
8 const __dirname = path.dirname(__filename);
9
10 // Inicjalizacja Firebase Admin SDK przy użyciu klucza Service Account
11 const serviceAccount = JSON.parse(
12   fs.readFileSync(path.join(__dirname, "serviceAccountKey.json"), "utf8")
13 );
14
15 admin.initializeApp({
16   credential: admin.credential.cert(serviceAccount),
17 });
18
19 const db = admin.firestore();
20
21 // Funkcja importująca dane z pliku JSON do kolekcji
22 async function importCollection(collectionName, jsonFilePath) {
23   const data = JSON.parse(fs.readFileSync(jsonFilePath, "utf8"));
24
25   for (const item of data) {
26     // Jeśli w danych jest pole "id", użyj go jako identyfikatora dokumentu
27     const docId = item.id ? item.id.toString() : undefined;
28     if (docId) {
29       await db.collection(collectionName).doc(docId).set(item);
30     } else {
31       await db.collection(collectionName).add(item);
32     }
33     console.log(`Imported document into ${collectionName}`);
34   }
35 }
36
37 async function main() {
38   try {
39     importCollection("products", "products.json");
40     importCollection("purchases", "purchases.json");
41     importCollection("contact-messages", "contact_messages.json");
42     importCollection("invoice-details", "invoice_details.json");
43     importCollection("shipping-details", "shipping_details.json");
44     importCollection("payment-details", "payment_details.json");
45     importCollection("newsletter", "newsletter.json");
46     importCollection("products", "products.json");
47     importCollection("purchases", "purchases.json");
48     importCollection("contact-messages", "contact_messages.json");
49     importCollection("invoice-details", "invoice_details.json");
50     importCollection("shipping-details", "shipping_details.json");
51     importCollection("payment-details", "payment_details.json");
52     importCollection("newsletter", "newsletter.json");
53   } catch (error) {
54     console.error(error);
55   }
56 }
57
58 main().catch((error) => {
59   console.error(error);
60 });

dawidolko Adding file of rules to database.

Name
..
README.md
contact_messages.json
invoice_details.json
newsletter.json
payment_details.json
products.json
purchases.json
rules.txt
shipping_details.json
```

Tworzenie kolekcji i dokumentów w Firebase stanowi fundament organizacji danych w aplikacji. W Cloud Firestore dane są przechowywane w dokumentach, które są grupowane w kolekcje, umożliwiając elastyczne i hierarchiczne modelowanie informacji. Każdy dokument zawiera zestaw par klucz-wartość, a jego struktura może być rozszerzana o zagnieżdżone podkolekcje, co pozwala na dynamiczne dostosowywanie bazy do specyficznych potrzeb projektu. Intuicyjne operacje dodawania, modyfikowania i usuwania danych przekładają się na szybkie reagowanie aplikacji na zmiany w środowisku użytkowników, co jest kluczowe dla optymalizacji zapytań oraz zapewnienia wydajności i skalowalności całego systemu.

Bezpieczeństwo i reguły dostępu w Firebase stanowią fundament ochrony danych oraz precyzyjnej kontroli operacji odczytu i zapisu. Reguły definiowane w Firebase pozwalają na ustalenie, które operacje mogą być wykonywane przez poszczególnych użytkowników, co umożliwia wdrażanie zaawansowanych polityk bezpieczeństwa w oparciu o role i kontekst żądań. Mechanizmy te eliminują ryzyko nieautoryzowanego dostępu i pozwalają na dynamiczne dostosowywanie uprawnień do specyficznych potrzeb aplikacji. Dzięki narzędziom do testowania i symulacji reguł, deweloperzy mogą weryfikować ich skuteczność przed wdrożeniem, co gwarantuje wysoki poziom ochrony danych i stabilność systemu.

```
1 rules_version = '2';
2 service cloud.firestore {
3     match /databases/{database}/documents {
4         match /users/{userId} {
5             allow read: if true;
6             allow write: if request.auth != null && request.auth.uid == userId;
7         }
8         match /{document=**} {
9             allow read: if true;
10            allow write: if request.auth != null;
11        }
12    }
13 }
```

# Bezpieczeństwo i reguły dostępu



Synchronizacja danych w czasie rzeczywistym to kluczowa cecha Firebase, która umożliwia natychmiastowe odzwierciedlenie zmian w bazie danych u wszystkich użytkowników bez potrzeby ręcznego odświeżania. Mechanizm ten opiera się na stałym nasłuchiwaniu zdarzeń, co pozwala na automatyczne przesyłanie aktualizacji między klientem a serwerem. Dzięki temu rozwiązaniu aplikacje stają się bardziej dynamiczne i responsywne – każda zmiana, wprowadzona przez jednego użytkownika, jest błyskawicznie widoczna dla pozostałych. To podejście gwarantuje spójność danych oraz umożliwia tworzenie interfejsów, które płynnie reagują na bieżące interakcje.

---

## SYNCHRONIZACJA DANYCH W CZASIE RZECZYWISTYM

---



# Obsługa zdarzeń i powiadomień

Obsługa zdarzeń i powiadomień w Firebase umożliwia dynamiczne reagowanie na zmiany w aplikacji oraz automatyczne informowanie użytkowników o istotnych wydarzeniach. Firebase monitoruje operacje na danych - takie jak dodanie, modyfikacja czy usunięcie - i pozwala na konfigurację funkcji, które natychmiast wywołują określone akcje w odpowiedzi na te zdarzenia. W połączeniu z Firebase Cloud Messaging, platforma umożliwia wysyłanie powiadomień push, co podnosi interaktywność aplikacji oraz angażuje użytkowników, dostarczając im aktualne informacje i przypomnienia w czasie rzeczywistym.

# INTEGRACJA Z FIREBASE AUTHENTICATION

Identifier	Providers	Created ↓	Signed In	User UID
test123@onet.pl	✉	Mar 11, 2025	Mar 11, 2025	vXsy0mrH1UMYxtqh003cYxg...
dawid1@onet.pl	✉	Mar 11, 2025	Mar 12, 2025	jUkLZb9xqATbNr0gwjvOHHbh...
piotr@onet.pl	✉	Mar 11, 2025	Mar 12, 2025	KhDMCPm5h9eG5aE7q7m8V...
test@onet.pl	✉	Mar 11, 2025	Mar 11, 2025	sosUw4LudXRmyLxARxd8aZZ...

Integracja z Firebase Authentication umożliwia wdrożenie bezpiecznego systemu uwierzytelniania użytkowników w Twojej aplikacji. Rozpoczyna się od dodania odpowiednich bibliotek SDK oraz skonfigurowania metod logowania w konsoli Firebase, co pozwala na wybór spośród wielu opcji, takich jak logowanie za pomocą adresu e-mail i hasła, uwierzytelnianie przy użyciu numeru telefonu czy integracja z kontami mediów społecznościowych (Google, Facebook, Twitter). Następnie implementuje się interfejs użytkownika, który obsługuje rejestrację, logowanie oraz zarządzanie sesjami, co znacząco upraszcza proces wdrażania autoryzacji. Firebase Authentication automatycznie dba o weryfikację tożsamości, obsługę sesji oraz zabezpieczenia, eliminując potrzebę pisania niestandardowych rozwiązań. Dzięki temu deweloperzy mogą skoncentrować się na budowaniu funkcjonalności aplikacji, mając pewność, że dostęp do zasobów jest chroniony przez zaawansowane mechanizmy bezpieczeństwa.

---

## FIREBASE STORAGE I PRZECHOWYWANIE MULTIMEDIÓW

---

Firebase Storage umożliwia łatwe przechowywanie i udostępnianie plików, w tym multimedów takich jak zdjęcia, filmy, dokumenty czy pliki audio. Usługa ta opiera się na infrastrukturze Google Cloud Storage, co gwarantuje niezawodność, wysoką dostępność oraz skalowalność, niezależnie od wielkości przesyłanych danych. Dzięki zintegrowanym mechanizmom bezpieczeństwa, takim jak reguły dostępu, Firebase Storage pozwala na precyzyjne kontrolowanie, kto i w jaki sposób może odczytywać lub modyfikować przechowywane zasoby. Integracja z Firebase Authentication umożliwia tworzenie spersonalizowanych polityk bezpieczeństwa, co jest kluczowe przy zarządzaniu danymi użytkowników. Dodatkowo, usługa wspiera operacje asynchroniczne, co zapewnia płynny przesył plików oraz możliwość monitorowania postępu transferu w czasie rzeczywistym. Dzięki tym właściwościom, Firebase Storage stanowi idealne rozwiązanie dla aplikacji, w których dynamiczne zarządzanie multimediami i wysoka wydajność są priorytetami.



# WPROWADZENIE DO FIREBASE CLOUD FUNCTIONS

Firebase Cloud Functions umożliwiają uruchamianie kodu backendowego bez zarządzania serwerami. Działają one w modelu serverless, co pozwala deweloperom skupić się na logice aplikacji, zamiast martwić się o infrastrukturę. Funkcje te reagują na różnorodne zdarzenia, takie jak zmiany w bazie danych, wywołania HTTP czy zdarzenia uwierzytelniania, co umożliwia automatyczne przetwarzanie danych, wysyłanie powiadomień czy integrację z zewnętrznymi API. Dzięki automatycznej skalowalności, Firebase Cloud Functions dostosowują się do obciążenia, zapewniając wysoką wydajność i niezawodność systemu. Integracja z pozostałymi usługami Firebase pozwala na tworzenie spójnych, zaawansowanych rozwiązań backendowych przy minimalnych kosztach operacyjnych.

# ≡ Monitoring i analiza aplikacji z Firebase Analytics

Firebase Analytics to zaawansowane narzędzie do monitorowania i analizy aplikacji, umożliwiające zbieranie szczegółowych danych o zachowaniach użytkowników oraz wydajności aplikacji. Dzięki gromadzeniu danych w czasie rzeczywistym, umożliwia identyfikację trendów, wykrywanie problemów oraz szybkie reagowanie na zmiany w interakcjach użytkowników. Raporty generowane przez Firebase Analytics pozwalają na segmentację odbiorców, analizę efektywności poszczególnych funkcji oraz mierzenie skuteczności kampanii marketingowych. Integracja z innymi usługami Firebase dodatkowo usprawnia proces optymalizacji aplikacji, umożliwiając podejmowanie świadomych decyzji opartych na rzetelnych danych.



Projekt sklepu komputerowego oparty na Firebase to przykład nowoczesnego rozwiązania e-commerce, w którym integracja front-endu i back-endu odbywa się za pomocą zaawansowanych usług platformy. Dzięki automatycznej synchronizacji danych w czasie rzeczywistym, system umożliwia bieżące aktualizowanie stanów magazynowych, statusów zamówień oraz interakcji użytkowników, co jest kluczowe w dynamicznym środowisku sklepowym. Front-end oparty na Vue współpracuje bezpośrednio z backendem Firebase, realizując uwierzytelnianie, zarządzanie danymi oraz obsługę płatności. Podejście to upraszcza zarządzanie projektem, umożliwia szybkie wdrażanie nowych funkcjonalności oraz gwarantuje skalowalność i wysoki poziom bezpieczeństwa, niezbędne dla efektywnej obsługi współczesnych platform e-commerce.

## DZIAŁANIE FIREBASE W PROJEKCIE SKLEPU KOMPUTEROWEGO

W projekcie zastosowano prostą integrację Vue z Firebase, której podstawą jest plik konfiguracyjny `firebase.js`. Zawiera on klucz API oraz identyfikator projektu, dzięki czemu aplikacja nawiązuje bezpieczne połaczenie z bazą Firebase. Następnie utworzono kolekcje danych poprzez `import.js`, dzięki czemu utworzone zostały wszystkie kolekcje użyte w projekcie, pełnią funkcję tabel, umożliwiających przechowywanie i odczyt informacji o produktach oraz zamówieniach. Aplikacja nie korzysta z Firebase Storage - wszystkie zdjęcia są przechowywane lokalnie w strukturze projektu Vue. Dodatkowo zaimplementowano moduł Firebase Authentication do obsługi procesu rejestracji i logowania użytkowników, co zapewnia kontrolę dostępu oraz bezpieczeństwo danych.

23

```
15 admin.initializeApp({
16   credential: admin.credential.cert(serviceAccount),
17 });
18
19 const db = admin.firestore();
20
21 // Funkcja importująca dane z pliku JSON do kolekcji
22 async function importCollection(collectionName, jsonFilePath) {
23   const data = JSON.parse(fs.readFileSync(jsonFilePath, "utf8"));
24
25   for (const item of data) {
26     // Jeśli w danych jest pole "id", użyj go jako identyfikatora dokumentu
27     const docId = item.id ? item.id.toString() : undefined;
28     if (docId) {
29       await db.collection(collectionName).doc(docId).set(item);
30     } else {
```

```
9   apiKey: "AIzaSyDpoR9FLkSQMaYpfrij-gXkmtwXfL4epjw",
10  authDomain: "sklep-vue-firebase.firebaseio.com",
11  projectId: "sklep-vue-firebase",
12  storageBucket: "sklep-vue-firebase.appspot.com",
13  messagingSenderId: "260406460193",
14  appId: "1:260406460193:web:cb9931fe9fdb9d96698860",
15  measurementId: "G-NJ6SPBZFVB",
16  };
--
```



# Zarządzanie kolekcjami w sklepie

24

---

Zarządzanie kolekcjami w sklepie komputerowym opiera się na efektywnym przeprowadzaniu operacji CRUD (tworzenie, odczyt, aktualizacja i usuwanie danych) przy użyciu mechanizmu „doc” z biblioteki Firebase w środowisku Vue. Komponent ten umożliwia dostęp do konkretnych dokumentów w kolekcjach, co pozwala na wygodne i szybkie aktualizowanie informacji.

Przykładowo, operacje takie jak odczyt (getDocs), aktualizacja (updateDoc) czy usuwanie dokumentów (deleteDoc) odbywają się bezpośrednio w komponencie Vue za pomocą prostych funkcji Firebase Firestore, co znacznie upraszcza zarządzanie danymi. Integracja Vue z Firebase pozwala także na natychmiastowe odzwierciedlanie zmian w interfejsie użytkownika, zapewniając aplikacji wysoką responsywność i aktualność danych.

---



Operacje CRUD (tworzenie, odczyt, aktualizacja, usuwanie) w Firebase wykonywane są za pomocą metody doc() dostępnej w bibliotece firebase/firestore. Metoda ta wskazuje konkretny dokument w bazie danych, na którym następnie można przeprowadzać operacje przy użyciu funkcji takich jak getDoc, setDoc, updateDoc czy deleteDoc. W projekcie sklepu komputerowego, stworzonym w Vue, podejście to pozwala na łatwą realizację zadań związanych z produktami, koszykiem oraz danymi użytkowników. Integracja Firebase z Vue umożliwia natychmiastowe odzwierciedlenie zmian w interfejsie użytkownika, co znaczco poprawia responsywność aplikacji i upraszcza proces zarządzania danymi.

```
<script setup>
import { ref, onMounted, computed }
import { useAuthStore } from "@/stores/auth";
import {
  collection,
  query,
  where,
  getDocs,
  doc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
} from "firebase/firestore";
import { db } from "@/firebase";
```

## OPERACJE CRUD W FIREBASE Z WYKORZYSTANIEM „DOC” W VUE

Tworzenie (Create) danych w Firebase odbywa się poprzez funkcje setDoc lub addDoc. Obie metody umożliwiają zapisanie nowego dokumentu w wybranej kolekcji. Funkcja setDoc pozwala określić identyfikator dokumentu ręcznie, natomiast addDoc automatycznie generuje unikalny identyfikator. W projekcie sklepu komputerowego funkcje te są używane do dodawania produktów do koszyka lub do tworzenia nowych produktów przez administratora. Dane przekazane z komponentów Vue są bezpośrednio wysyłane do Firebase, gdzie zostają zapisane i natychmiast dostępne dla innych użytkowników aplikacji, zapewniając aktualność informacji bez konieczności odświeżania strony.

```
async function handleAddProduct() {
  if (newProduct.value.price < 0 || newProduct.value.stock < 0) {
    showNotification("Price and stock must be 0 or greater.", "error");
    return;
  }
  try {
    const docRef = await addDoc(collection(db, "products"), newProduct.value);
    products.value.push({ ...newProduct.value, id: docRef.id });
    newProduct.value = { title: "", thumbnail: "", price: 0, stock: 0 };
    showNotification("Product added successfully.", "success");
  } catch (error) {
    console.error("Error adding product:", error);
    showNotification("Error adding product", "error");
  }
}
```

## Tworzenie (Create) danych w Firebase



Usuwanie (Delete) danych w Firebase odbywa się za pomocą funkcji deleteDoc, która umożliwia trwałe usunięcie konkretnego dokumentu wskazanego przez metodę doc(). W projekcie sklepu komputerowego mechanizm ten wykorzystywany jest między innymi przez administratora do usuwania produktów z oferty. Po wykonaniu tej operacji dokument natychmiast znika z bazy danych Firebase oraz z interfejsu użytkownika aplikacji Vue, zapewniając tym samym błyskawiczną aktualizację wyświetlanych informacji. Działanie tej funkcji jest bezpośrednie, proste i gwarantuje utrzymanie spójności danych w czasie rzeczywistym.

---

## USUWANIE (DELETE) DANYCH W FIREBASE

---

```
async function handleDeleteProduct(id) {
  try {
    await deleteDoc(doc(db, "products", id.toString()));
    products.value = products.value.filter((p) => p.id !== id);
    showNotification("Product deleted successfully.", "success");
  } catch (error) {
    console.error("Error deleting product:", error);
    showNotification("Error deleting product", "error");
  }
}
```

## Aktualizacja (Update) danych w Firebase

```
async function handleUpdateProduct(prod) {
  if (prod.price < 0 || prod.stock < 0) {
    showNotification("Price and stock must be 0 or greater.", "error");
    return;
  }
  try {
    await updateDoc(doc(db, "products", prod.id.toString()), {
      title: prod.title,
      thumbnail: prod.thumbnail,
      price: prod.price,
      stock: prod.stock,
    });
    showNotification("Product updated successfully.", "success");
  } catch (error) {
    console.error("Error updating product:", error);
    showNotification("Error updating product", "error");
  }
}
```

Aktualizacja (Update) danych w Firebase realizowana jest za pomocą funkcji `updateDoc`, która pozwala na bezpośrednią modyfikację istniejących dokumentów w bazie danych. W połączeniu z funkcją `doc()` wskazuje konkretny dokument, który należy zaktualizować. W projekcie sklepu komputerowego mechanizm ten wykorzystywany jest na przykład przy zmianie ilości produktów w koszyku użytkownika lub podczas edycji szczegółów produktu przez administratora. Każda zmiana wykonana przy pomocy `updateDoc` natychmiast odzwierciedla się zarówno w bazie Firebase, jak i w aplikacji Vue, gwarantując pełną synchronizację danych i aktualność wyświetlanych informacji.



# Odczyt (Read) danych w Firebase

```
// CRUD dla produktów
async function fetchProducts() {
  try {
    const snapshot = await getDocs(collection(db, "products"));
    let prods = [];
    snapshot.forEach(docSnap => {
      prods.push({ id: docSnap.id, ...docSnap.data() });
    });
    products.value = prods;
  } catch (error) {
    console.error("Error fetching products:", error);
    showNotification("Error fetching products", "error");
  }
}
```

Odczyt (Read) danych w Firebase realizowany jest przy użyciu funkcji `getDoc` dla pojedynczych dokumentów oraz `getDocs` dla całych kolekcji. Funkcje te umożliwiają pobranie danych z bazy Firebase bezpośrednio do aplikacji Vue, gdzie mogą być dynamicznie prezentowane użytkownikom. W projekcie sklepu komputerowego odczyt wykorzystywany jest między innymi do wyświetlania produktów, informacji w koszyku oraz historii zamówień użytkownika. Po pobraniu dane są przypisywane do zmiennych reaktywnych Vue, co pozwala na automatyczne odświeżanie widoku aplikacji przy każdej zmianie zawartości bazy danych. Dzięki temu aplikacja jest zawsze aktualna i responsywna.

## REALIZACJA PROCESU ZAMÓWIEŃ I OBSŁUGI UŻYTKOWNIKA W FIREBASE

Signed	Created	Mar 11, 2025	Mar 12, 2025	Mar 12, 2025	Mar 11, 2025	Mar 11, 2025
Identifier	test123@onet.pl	dawid1@onet.pl	piotr@onet.pl	test@onet.pl	dawid@onet.pl	
Providers	...	...	...	...	...	...

W projekcie sklepu komputerowego wykorzystano Firebase Cloud Functions oraz Cloud Firestore do implementacji kluczowych procesów biznesowych, takich jak obsługa zamówień, zarządzanie stanami magazynowymi oraz operacje związane z użytkownikami. Cloud Functions automatycznie reagują na zdarzenia, np. utworzenie nowego zamówienia, wysyłając powiadomienia lub zmieniając status produktu. Dodatkowo, Firebase Authentication integruje się bezpośrednio z Cloud Firestore, umożliwiając przypisywanie zamówień do konkretnych użytkowników oraz zarządzanie ich dostępem do zasobów aplikacji. Taka architektura zapewnia stabilność, skalowalność oraz bezpieczeństwo operacji, upraszczając rozwój i utrzymanie projektu.

# WYKORZYSTANIE FIREBASE EXTENSIONS DO ROZSzerZENIA MOżLIWOŚCI APLIKACJI

Firebase Extensions to gotowe do użycia rozszerzenia, które umożliwiają szybkie wdrożenie dodatkowych funkcji bez potrzeby pisania własnego kodu backendowego. Rozwiązania te bazują na Cloud Functions oraz innych usługach Firebase, dostarczając m.in. automatyczne tłumaczenia tekstów, wysyłanie e-maili, eksport danych czy integrację z usługami zewnętrznymi takimi jak Stripe czy Mailchimp. Dzięki Firebase Extensions możliwe jest błyskawiczne wzbogacenie aplikacji o zaawansowane funkcjonalności, co znaczaco skraca czas wdrażania nowych rozwiązań oraz obniża koszty ich implementacji.



# Obsługa błędów, debugowanie i testowanie

Obsługa błędów, debugowanie i testowanie w Firebase to fundament zapewnienia niezawodności i wysokiej jakości aplikacji. Platforma oferuje zaawansowane narzędzia, które umożliwiają bieżące monitorowanie awarii oraz analizę logów zarówno po stronie klienta, jak i backendu. Mechanizmy takie jak Crashlytics automatycznie zbierają i raportują informacje o błędach, co pozwala na szybkie ich wykrycie i naprawę. Dodatkowo, funkcje logowania dostępne w Cloud Functions umożliwiają śledzenie zachowań kodu, a dedykowane środowiska testowe Firebase pozwalają na symulację reguł bezpieczeństwa i walidację konfiguracji projektu przed wdrożeniem. Dzięki temu możliwe jest tworzenie stabilnych i odpornych na błędy aplikacji, które sprostają wymaganiom środowisk produkcyjnych.

# Skalowanie aplikacji w Firebase

Skalowanie aplikacji w Firebase opiera się na zaawansowanych mechanizmach automatycznego dostosowywania zasobów do bieżącego obciążenia. Dzięki infrastrukturze Google Cloud, usługi takie jak Cloud Functions, Cloud Firestore i Realtime Database dynamicznie reagują na wzrost ruchu, uruchamiając dodatkowe instancje i przydzielając więcej mocy obliczeniowej tam, gdzie jest to potrzebne. Monitorowanie wydajności oraz integracja z narzędziami analitycznymi pozwalają na optymalizację konfiguracji w czasie rzeczywistym, co minimalizuje ryzyko przeciążenia systemu. Taka elastyczność gwarantuje wysoką dostępność i płynność działania aplikacji nawet przy intensywnym użytkowaniu, co stanowi kluczowy element sukcesu nowoczesnych rozwiązań webowych.

Firebase oferuje szereg zaawansowanych funkcji, które znacznie rozszerzają możliwości tworzenia nowoczesnych aplikacji. Platforma umożliwia obsługę transakcji i złożonych zapytań w Cloud Firestore, co zapewnia spójność danych nawet przy dynamicznych zmianach. Dodatkowo, wsparcie dla trybu offline pozwala aplikacjom kontynuować pracę w warunkach tymczasowej utraty łączności, synchronizując zmiany, gdy tylko połączenie zostanie przywrócone. Firebase integruje się również z ML Kit, umożliwiając wdrażanie funkcji sztucznej inteligencji, takich jak rozpoznawanie obrazów czy przetwarzanie języka naturalnego, bez konieczności budowania dedykowanych modeli od podstaw. Cloud Functions pozwalają na tworzenie skalowalnych rozwiązań serverless, które automatycznie dostosowują się do obciążenia, a zaawansowane reguły bezpieczeństwa chronią dane na każdym etapie ich przetwarzania. Dzięki tym funkcjom Firebase stanowi kompleksowe narzędzie, które nie tylko upraszcza zarządzanie danymi, ale także umożliwia budowanie intelligentnych, elastycznych i bezpiecznych aplikacji.

---

## ZAAWANSOWANE FUNKCJE FIREBASE

---



Firebase wyróżnia się na tle innych platform BaaS dzięki kompleksowości i głębszej integracji z ekosystemem Google. W przeciwieństwie do konkurencyjnych rozwiązań, takich jak AWS Amplify, Parse czy Backendless, Firebase oferuje szeroki wachlarz usług - od baz danych (Realtime Database, Cloud Firestore), przez funkcje chmurowe (Cloud Functions), po zaawansowane narzędzia analityczne i uwierzytelnianie. Jego automatyczna synchronizacja danych, dynamiczna skalowalność i łatwość integracji z popularnymi frameworkami frontendowymi, np. Vue, sprawiają, że jest to rozwiązanie idealne zarówno dla startupów, jak i dużych przedsiębiorstw. Dodatkowo, Firebase umożliwia szybkie prototypowanie oraz minimalizuje potrzebę zarządzania infrastrukturą, co przekłada się na efektywność i bezpieczeństwo wdrożeń.

---

## PORÓWNANIE FIREBASE Z INNYMI PLATFORMAMI BAAS

---

# Podsumowanie i wnioski

Firebase jako kompleksowy ekosystem wspierający tworzenie nowoczesnych aplikacji. Poznaliśmy historię i ewolucję platformy, która zrewolucjonizowała podejście do zarządzania danymi dzięki mechanizmom synchronizacji w czasie rzeczywistym, elastycznemu modelowaniu danych i automatycznemu skalowaniu. Zaprezentowaliśmy kluczowe komponenty Firebase, takie jak Realtime Database, Cloud Firestore, Authentication, Storage i Cloud Functions, oraz omówiliśmy ich integrację z frameworkiem Vue, co umożliwia tworzenie dynamicznych i responsywnych interfejsów użytkownika. Przeanalizowaliśmy również aspekty bezpieczeństwa, reguły dostępu, obsługę błędów, debugowanie, testowanie oraz zaawansowane funkcje, w tym integrację z systemami płatności i rozwiązaniami zewnętrznymi. Podsumowując, Firebase to narzędzie, które nie tylko przyspiesza proces tworzenia aplikacji, ale także podnosi ich jakość i wydajność, stanowiąc idealne rozwiązanie dla deweloperów dążących do budowania skalowalnych, bezpiecznych i nowoczesnych projektów.



# DZIĘKUJEMY



*Olko Dawid*



*Piotr Smoła*