Лекция 10: Введение в Firebase и Firestore.

хекслет колледж

Цель занятия:

Сформировать у студентов понимание Firebase как платформы Backend-as-a-Service и приобрести практические навыки создания проекта, настройки Firestore и выполнения базовых операций с данными.

Учебные вопросы:

1. Что такое Firebase и зачем он нужен фронтенд-разработчику?
2. Что такое Firestore?

1. Что такое Firebase и зачем он нужен фронтенд-разработчику?

Ключевая проблема фронтенд-разработчика
Ситуация: Вы сделали красивый интерфейс интернет-магазина на React/Vue, но:

- Некуда сохранять товары и заказы
- Негде регистрировать пользователей
- Негде хранить фотографии товаров
- Нет сервера для обработки данных

Традиционное решение: Изучать бэкенд (Node.js, Python, базы данных), настраивать сервер, писать API...
Проблема: Это требует месяцев обучения, денег на хостинг и времени на поддержку.

Что такое Firebase?

Firebase — это Backend-as-a-Service (BaaS) платформа от Google.
Простая аналогия:
Если веб-разработка — это строительство дома:

- Frontend (HTML/CSS/JS) — это дизайн и отделка комнат
- Традиционный бэкенд — это фундамент, коммуникации, которые нужно строить самому
- Firebase — это готовый "фундамент с коммуникациями", который можно арендовать

Firebase = готовый бэкенд в облаке, который настраивается через консоль и используется из JavaScript.

Зачем именно фронтенд-разработчику?

1. Можно делать fullstack-приложения без изучения бэкенда:

- Вы уже знаете JavaScript
- Firebase имеет JavaScript SDK

Результат: Можете создать полностью рабочее приложение (интернет-магазин, чат, блог) самостоятельно

2. Быстрый старт проекта:

- Создание проекта: 5 минут вместо 5 дней
- Готовые решения: аутентификация, база данных, хостинг в одном месте
- Бесплатный тариф: достаточно для 90% учебных и стартап-проектов

3. Портфолио и практика:

- Проблема студента: "В моём портфолио только статические сайты"
- Решение с Firebase: Можете добавить реальные проекты с:
  - Регистрацией пользователей
  - Базами данных
  - Загрузкой файлов
  - Real-time обновлениями

Ключевые сервисы Firebase для веб-разработчика:

Сервис	За что отвечает	Аналог в традиционной разработке
Firestore	База данных для товаров и заказов	MongoDB + сервер Node.js
Authentication	Регистрация и вход пользователей	Passport.js + сессии/куки
Storage	Хранение фотографий товаров	Multer + облачное хранилище
Hosting	Размещение готового сайта	Nginx/Apache на VPS

Полезные дополнительные:

Сервис	Для чего нужен
Cloud Functions	Серверный код (например, отправка email о заказе)
Cloud Messaging	Push-уведомления для мобильных приложений
Analytics	Аналитика пользовательского поведения

Real-кейс: Интернет-магазин на Firebase

Без Firebase:

- Арендовать сервер (от $10/месяц)
- Установить Node.js, настроить Express
- Настроить базу данных (PostgreSQL/MongoDB)
- Написать API для товаров, заказов, пользователей
- Настроить аутентификацию
- Настроить загрузку файлов
- Развернуть фронтенд

Итого: ~2-4 недели работы, нужен опыт бэкенд-разработки

С Firebase:

- Создать проект в консоли (5 мин)
- Включить Firestore, Authentication, Storage
- Подключить SDK к фронтенду (10 мин)
- Начать писать логику приложения

Итого: ~1 день до первого прототипа, нужен только JavaScript

Когда НЕ использовать Firebase?

Не подходит, если:

- Сложная бизнес-логика на сервере (банковские операции, сложные расчёты)
- Жёсткие требования к данным (юридические, медицинские данные с особыми требованиями)
- Очень большие объёмы данных (может стать дорого)
- Нужен полный контроль над инфраструктурой

Идеально подходит для:

- Стартапов и прототипов
- Учебных проектов
- Приложений со средним трафиком
- Команд без бэкенд-разработчиков
- Real-time приложений (чаты, уведомления)

Firebase в цифрах

Бесплатный тариф (Spark Plan):

- Firestore: 1 ГБ хранилища, 50 тыс. чтений/день
- Storage: 5 ГБ, 1 ГБ трафика/день
- Hosting: 10 ГБ, 360 МБ/день трафик
- Authentication: 10 тыс. активных пользователей

- Достаточно для: товаров с описаниями
- 1000 заказов
- 5000 пользователей
- Ежедневная аудитория до 1000 человек

Альтернативы Firebase

Другие BaaS-платформы:

- Supabase — "Open Source Firebase" на PostgreSQL
- AWS Amplify — от Amazon, больше настроек
- Back4App — на основе Parse Server

Но Firebase:

- Самый популярный (больше туториалов, решений проблем)
- Лучшая документация
- Плавный переход на платные тарифы
- Интеграция с другими сервисами Google

Выводы:

Firebase для студента-фронтендера это:

- Возможность создать полноценное приложение сразу
- Портфолио с реальными проектами
- Понимание работы с данными без углубления в бэкенд
- Быстрый результат для мотивации
- Бесплатный старт для обучения

Главный принцип:
"Не изучайте бэкенд с нуля. Используйте готовый бэкенд (Firebase) и сосредоточьтесь на том, что вы уже умеете - фронтенд-разработке."

2. Что такое Firestore?

Firestore — это облачная NoSQL база данных от Google, которая позволяет фронтенд-разработчикам хранить и синхронизировать данные прямо из JavaScript-кода без написания серверной части.

Ключевые особенности Firestore

1. Документная модель данных (как папки и файлы)
2. Работает прямо из браузера (без сервера)
3. Real-time обновления (самая крутая фича)
4. Офлайн-режим, работа без интернета, за счет кэширования

Ограничения и важные моменты

Что можно:

✅ Хранить структурированные данные (товары, пользователи, заказы)

✅ Делать сложные запросы (фильтры, сортировка, пагинация)

✅ Работать офлайн

✅ Настроить права доступа (покупатель ≠ админ)

✅ Автоматическое масштабирование

Что нельзя (или сложно):

❌ Сложные SQL-джойны

❌ Полнотекстовый поиск

❌ Сложные агрегации

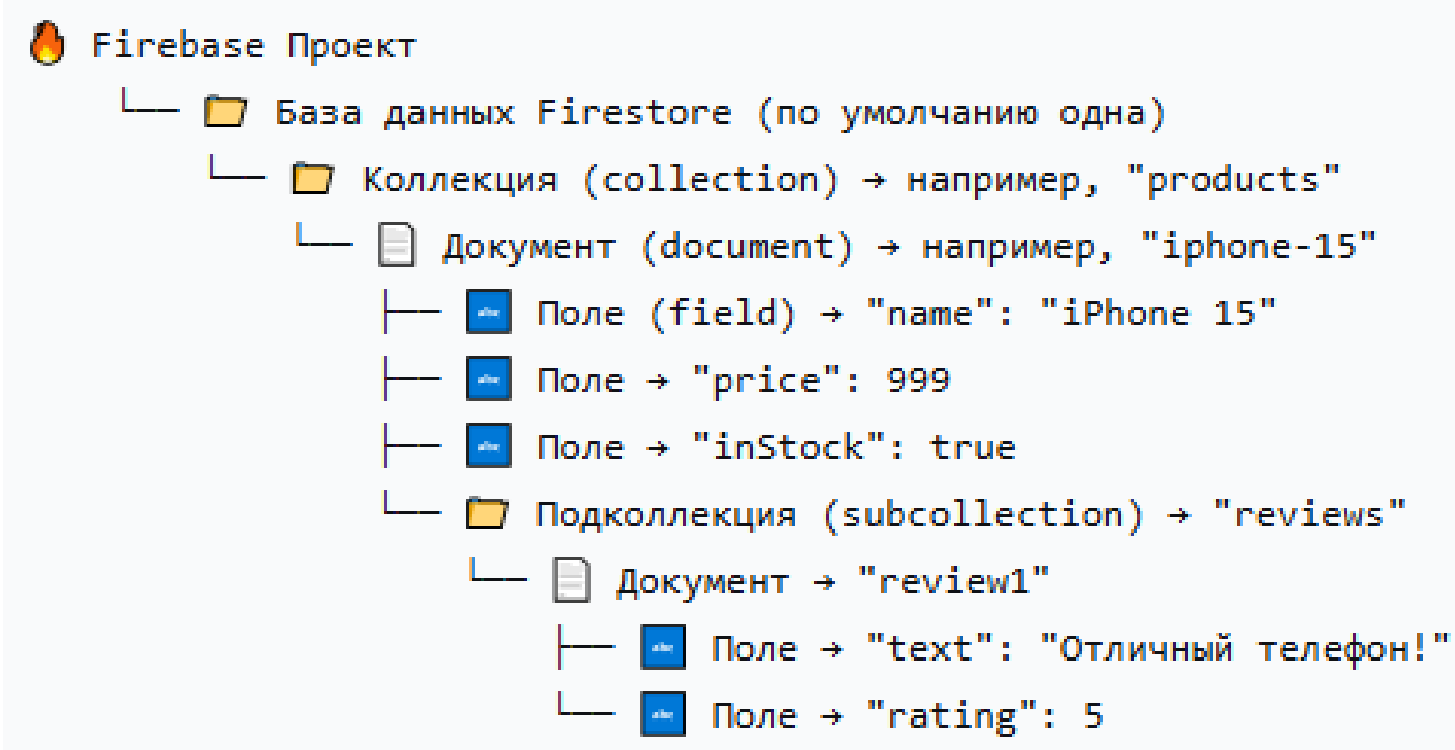
❌ Хранить файлы >1 МБ

3. Как устроена модель данных в Firestore ?

Ключевая метафора: Библиотека
Firestore — это цифровая библиотека:

- Библиотека = База данных Firestore
- Стеллажи = Коллекции
- Книги = Документы
- Главы в книге = Поля документа
- Папка с рецензиями внутри книги = Подколлекция

Иерархия: от общего к частному



Коллекция (Collection) — как таблица в SQL

Что такое:

- Группа однотипных документов. Аналог таблицы в SQL или папки в компьютере.

Примеры для магазина:

- products — все товары
- users — все пользователи
- orders — все заказы
- categories — категории товаров

Особенности:

✅ Не имеет полей — только содержит документы
✅ Нет схемы — документы внутри могут иметь разную структуру
✅ Автоматически не создаётся — создаётся при добавлении первого документа

Документ (Document) — как строка в таблице

Что такое:

- Отдельная запись в коллекции. Содержит данные в формате ключ-значение.

ID документа:

- Автоматически (если не указать): Lp2r4sT8qK1n (рандомный)
- Вручную (если важно): iphone-15-pro, user-123

Ограничения документа:

- Максимум 1 МБ данных
- Максимум 20 000 полей (но лучше < 100)
- Нельзя хранить файлы > 10 КБ (используем Storage)

Поле (Field) — ячейка с данными

Типы данных в Firestore:

Тип	Пример	Зачем нужно
String	"iPhone 15"	Текст
Number	999, 99.99	Цены, количества
Boolean	true/false	Флаги (в наличии/нет)
Timestamp	Timestamp.now()	Дата создания заказа
Array	["red", "blue"]	Цвета товара, теги
Map/Object	{memory: "256GB"}	Вложенные данные
Null	null	Отсутствие значения
Reference	db.doc("users/abc123")	Ссылка на другой документ
GeoPoint	new GeoPoint(55.75, 37.61)	Координаты доставки

Пример:

```
// Товар в магазине
const product = {
  name: "iPhone 15",                    // String
  price: 999.99,                        // Number (деньги)
  inStock: true,                        // Boolean
  createdAt: firebase.firestore.FieldValue.serverTimestamp(), // Авто-дата
  colors: ["black", "white", "gold"], // Array вариантов
  specs: {                              // Map для сложных данных
    memory: "256GB",
    display: "6.1 inches"
  },
  categoryRef: db.doc("categories/smartphones"), // Reference
  location: new firebase.firestore.GeoPoint(55.75, 37.61) // Для доставки
};
```

Подколлекция (Subcollection) — документ внутри документа

Что такое:

- Коллекция, которая принадлежит конкретному документу.

```
users (коллекция)
└── user_123 (документ пользователя)
    ├── name: "Анна"
    ├── email: "anna@mail.ru"
    └── orders (подколлекция заказов этого пользователя)
        └── order_456 (документ заказа)
            ├── total: 1999
            └── status: "delivered"
```

4. Как создать проект и базу данных в Firebase Console?

Что нужно сделать:

- Создать аккаунт Google (если нет)
- Зайти в Firebase Console
- Создать новый проект
- Добавить Firestore базу данных
- Настроить в тестовом режиме
- Добавить первый товар

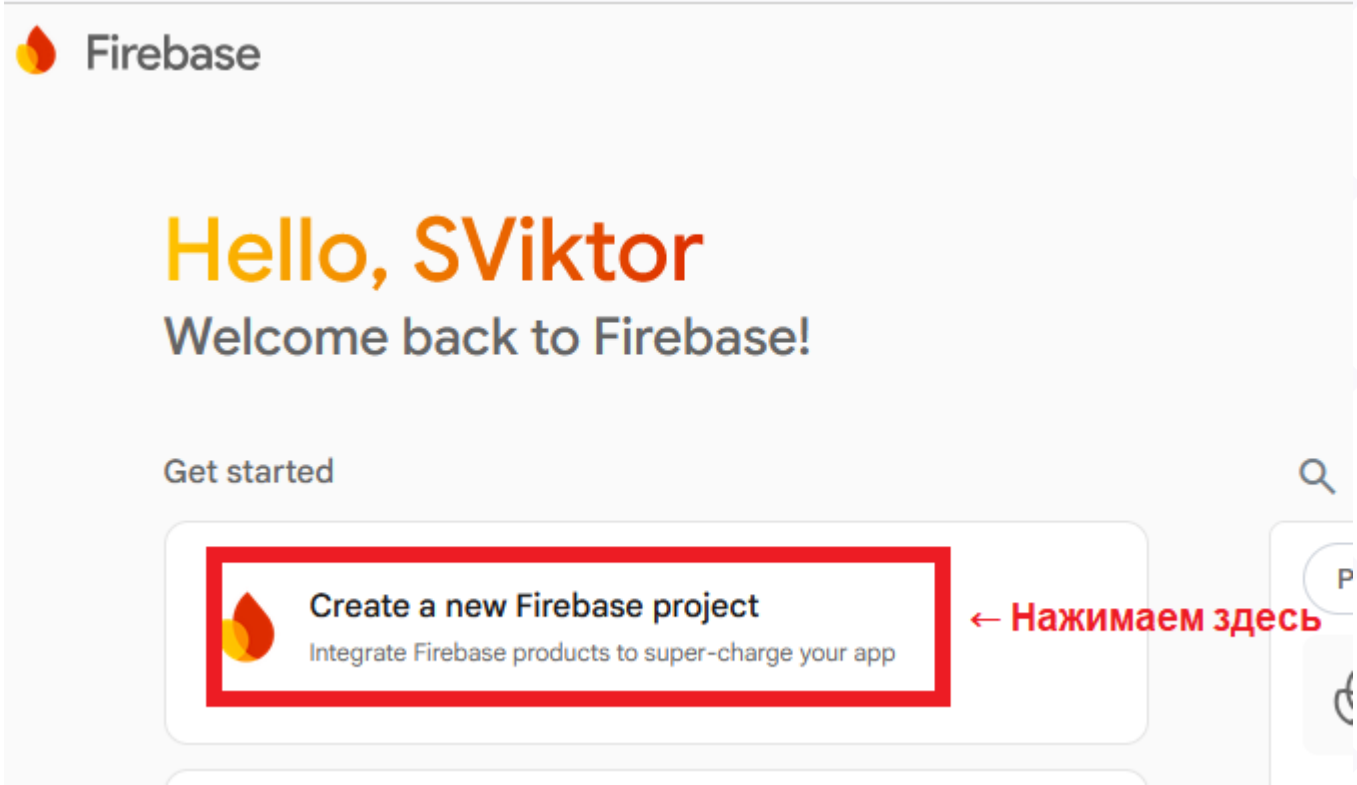
🎯 Конечный результат:

- Готовая база данных с первым товаром для нашего интернет-магазина.

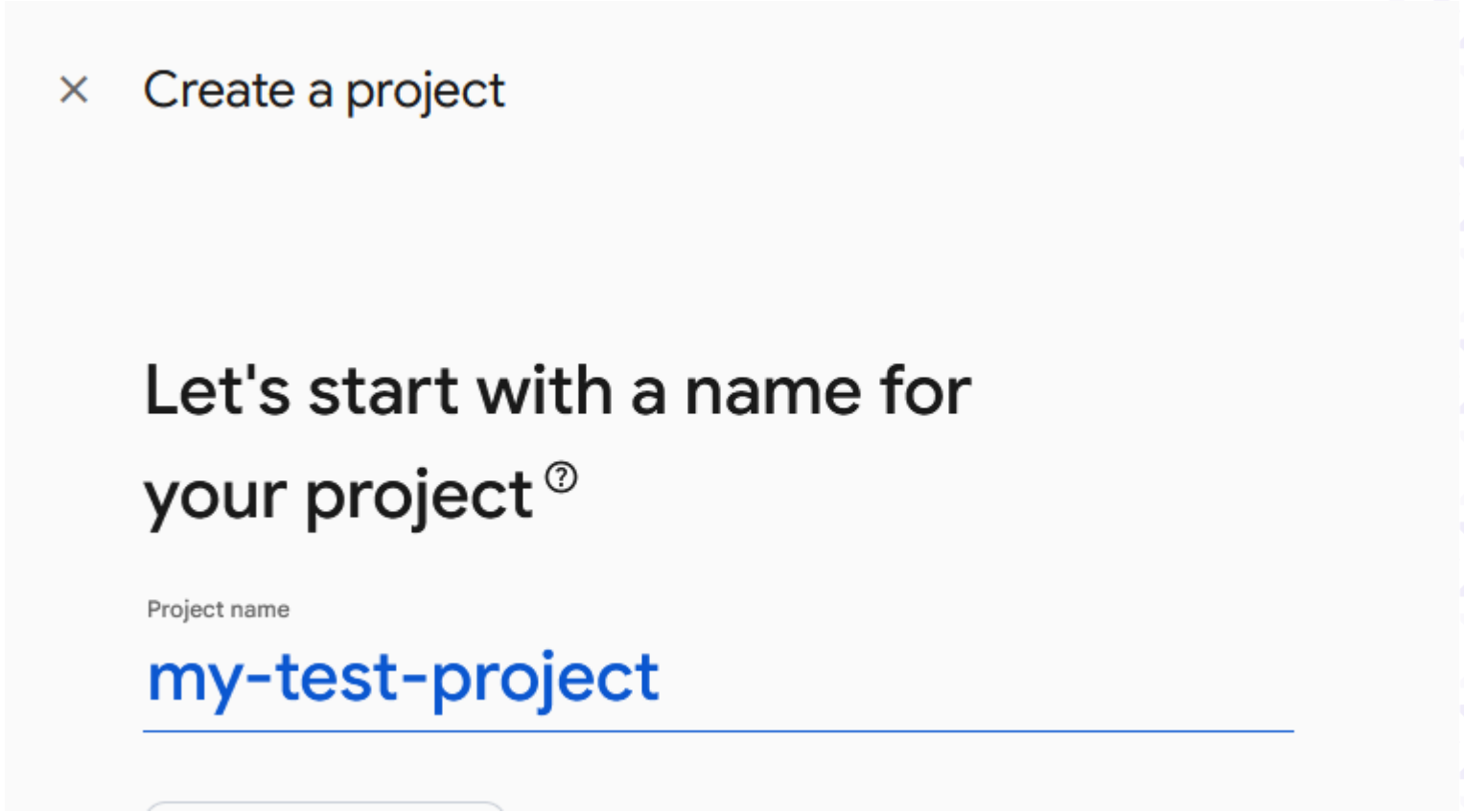
Шаг 1: Вход в Firebase Console

- Откройте браузер (Chrome, Firefox, Edge)
- В адресной строке введите:
  console.firebase.google.com
- Войдите под своим Google-аккаунтом

Шаг 2: Создание проекта



Название проекта



Create a project

Get campaign insights and recommendations to engage your users through Firebase Cloud Messaging

Generate schema and explore your data using natural language in Firebase Data Connect

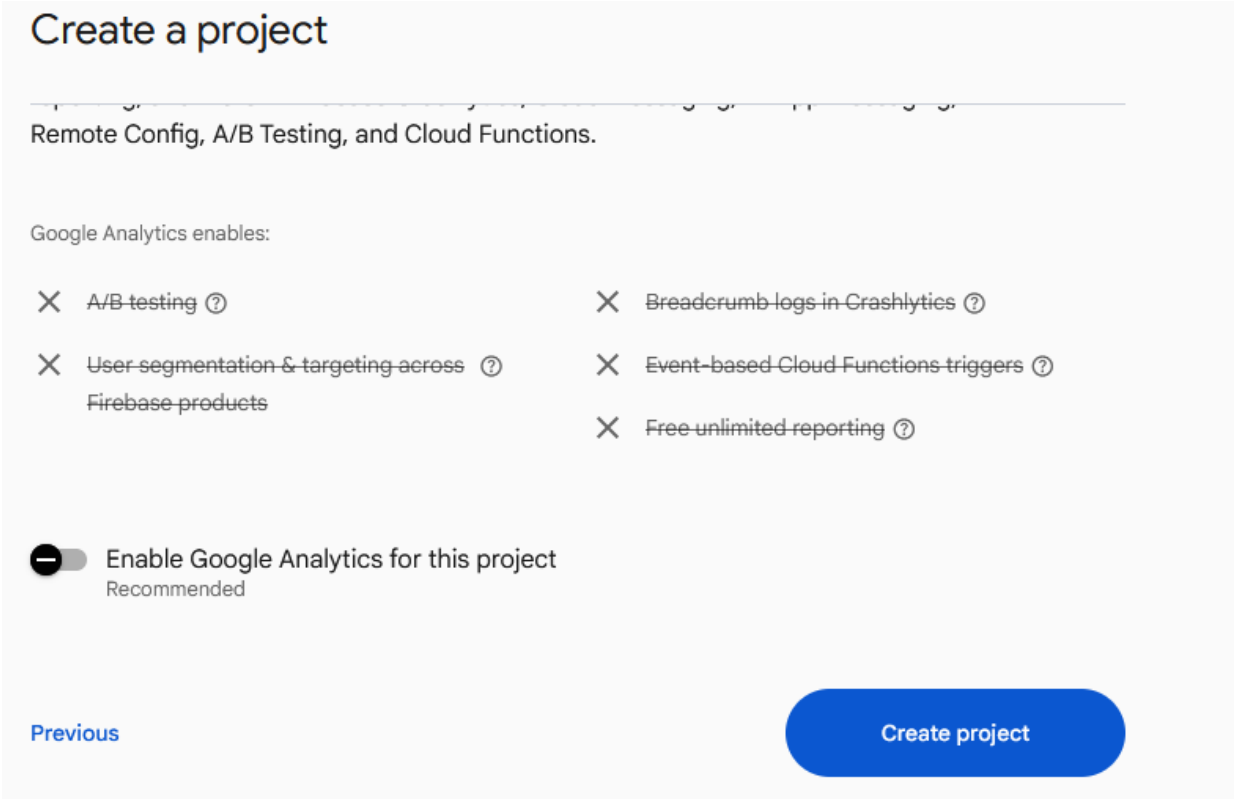
Enable Gemini in Firebase
Recommended

Disclaimers:
Gemini in Firebase is still under development and may give inaccurate information. Test any code it generates before using it in your apps. Gemini can interact with your Firebase data and tailor response to your app, and it may use your prompts or data for training. Learn more about how we train our models Use of Gemini in Firebase is subject to Google Terms of Service and the Generative AI Prohibited Use Policy

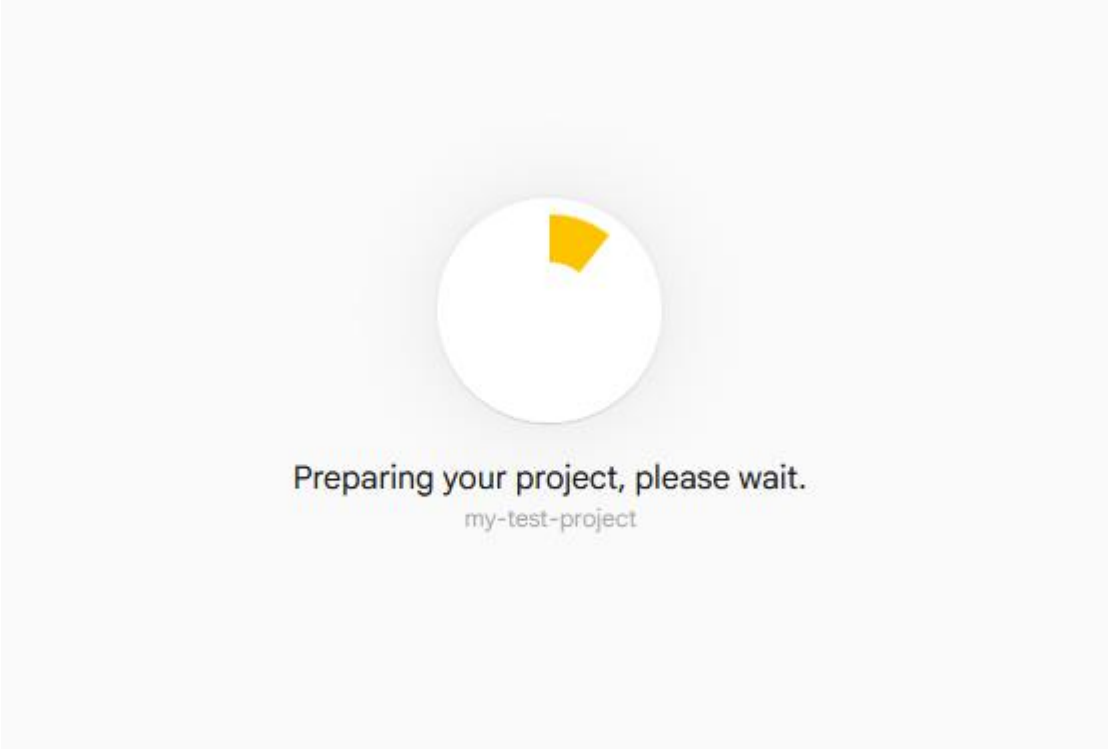
Previous

Continue

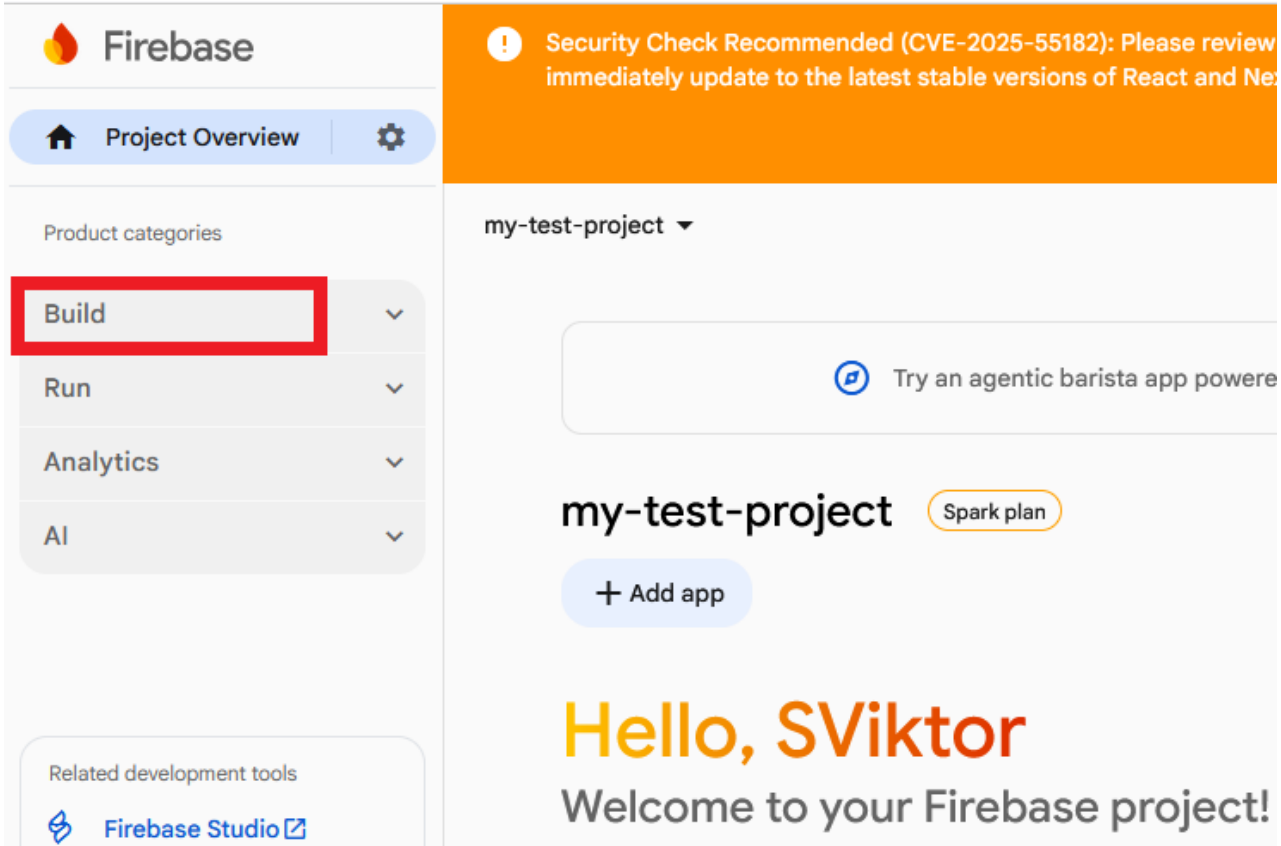
Google Analytics (пропускаем)

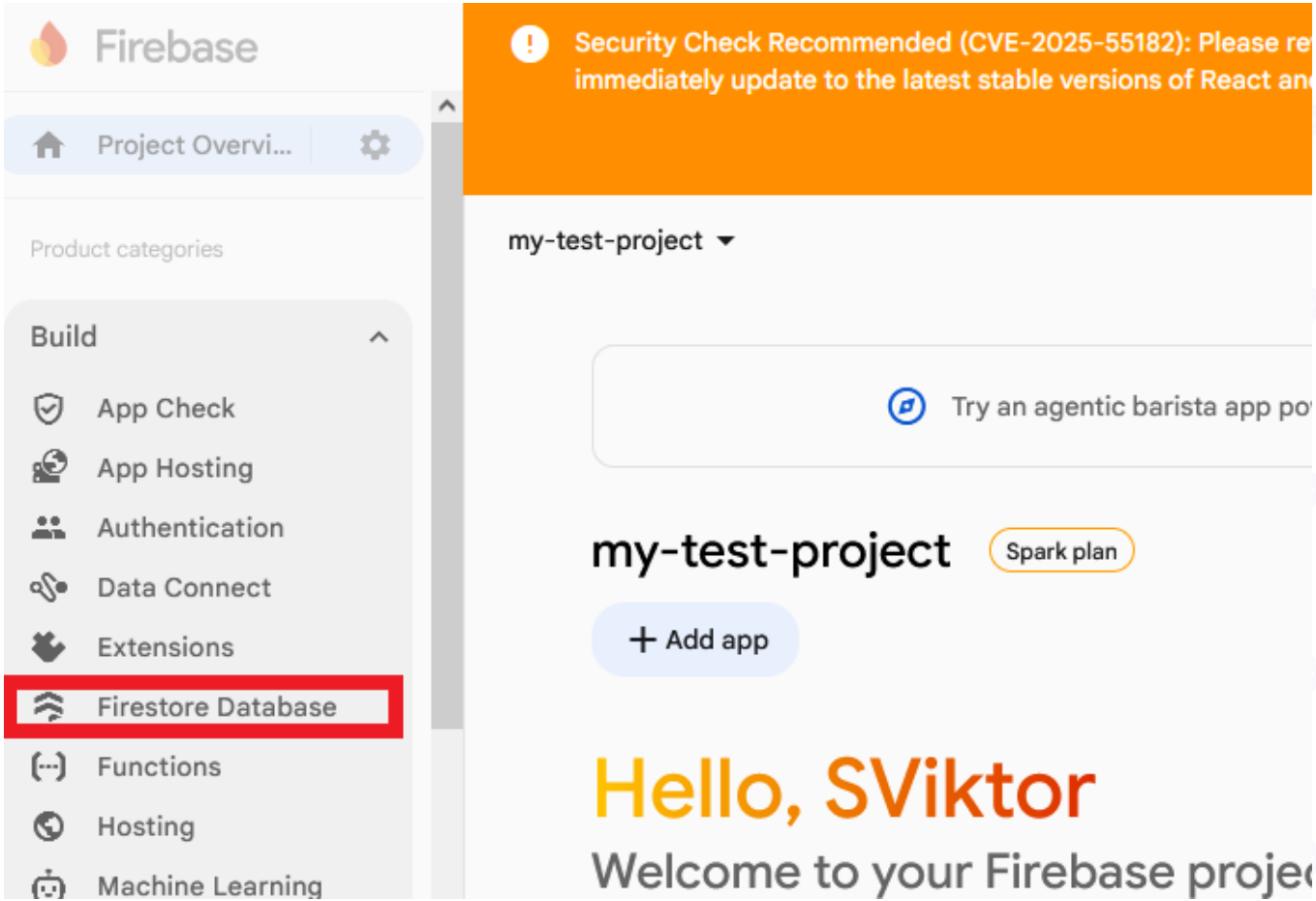


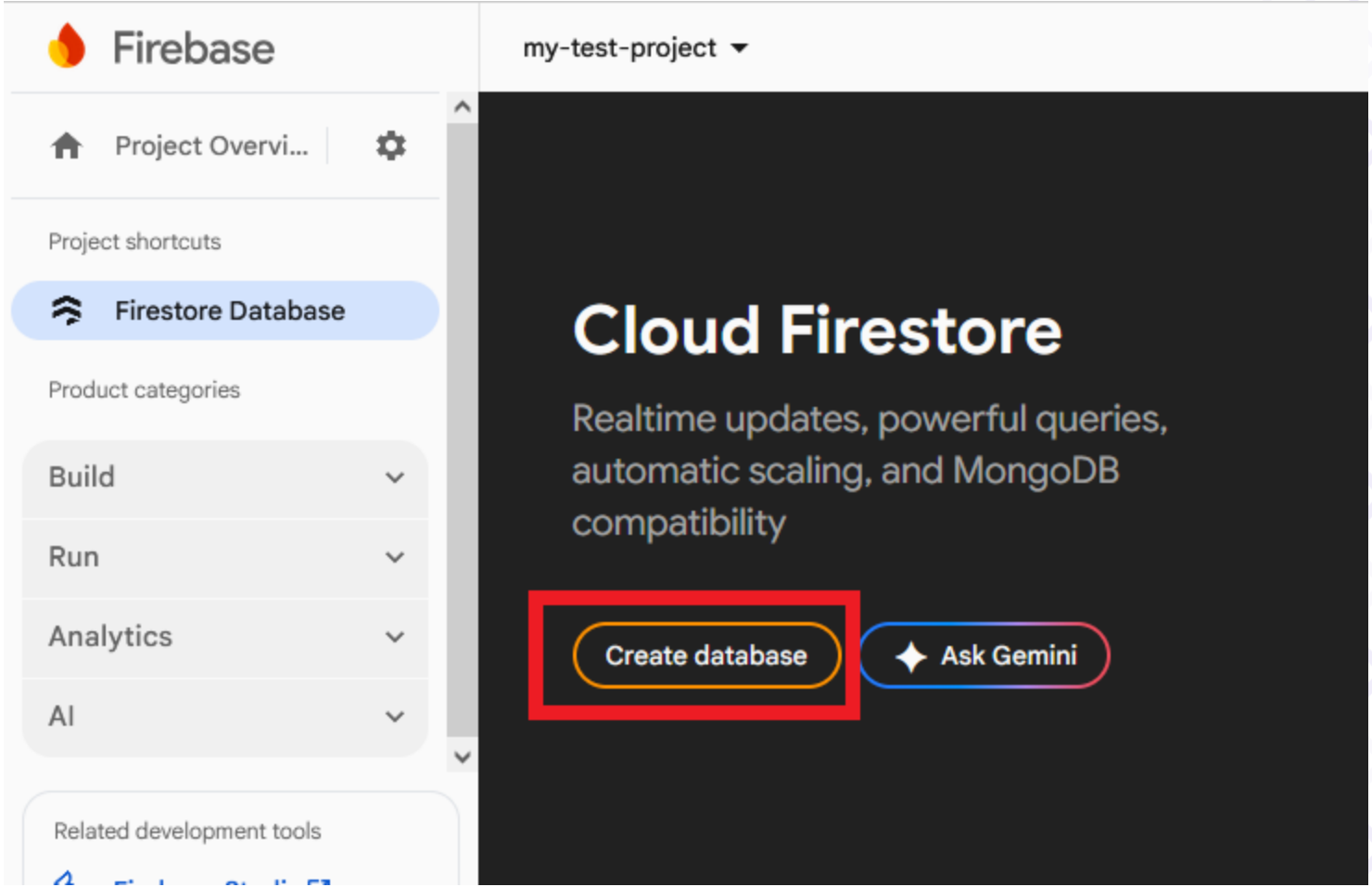
Ждём создания

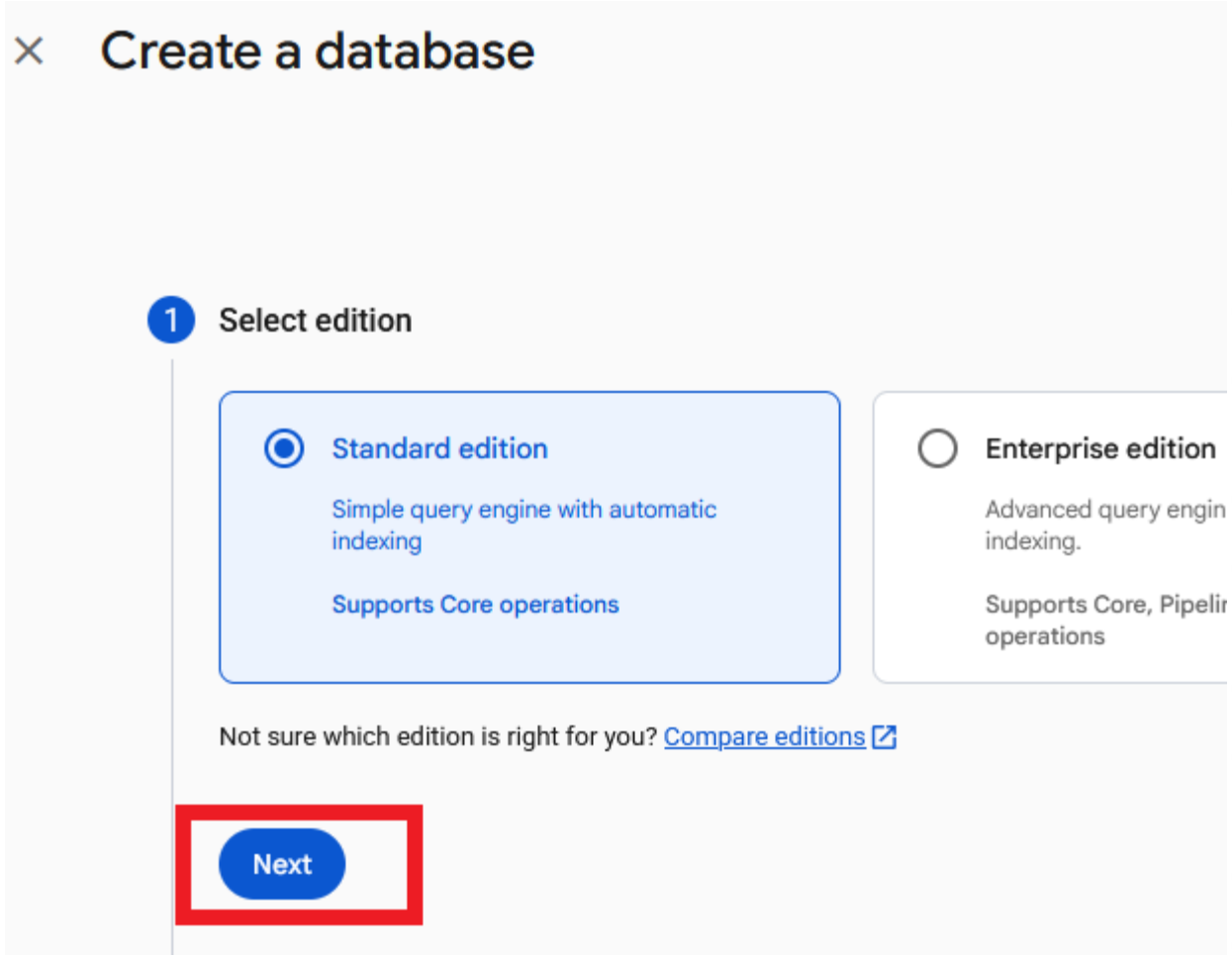


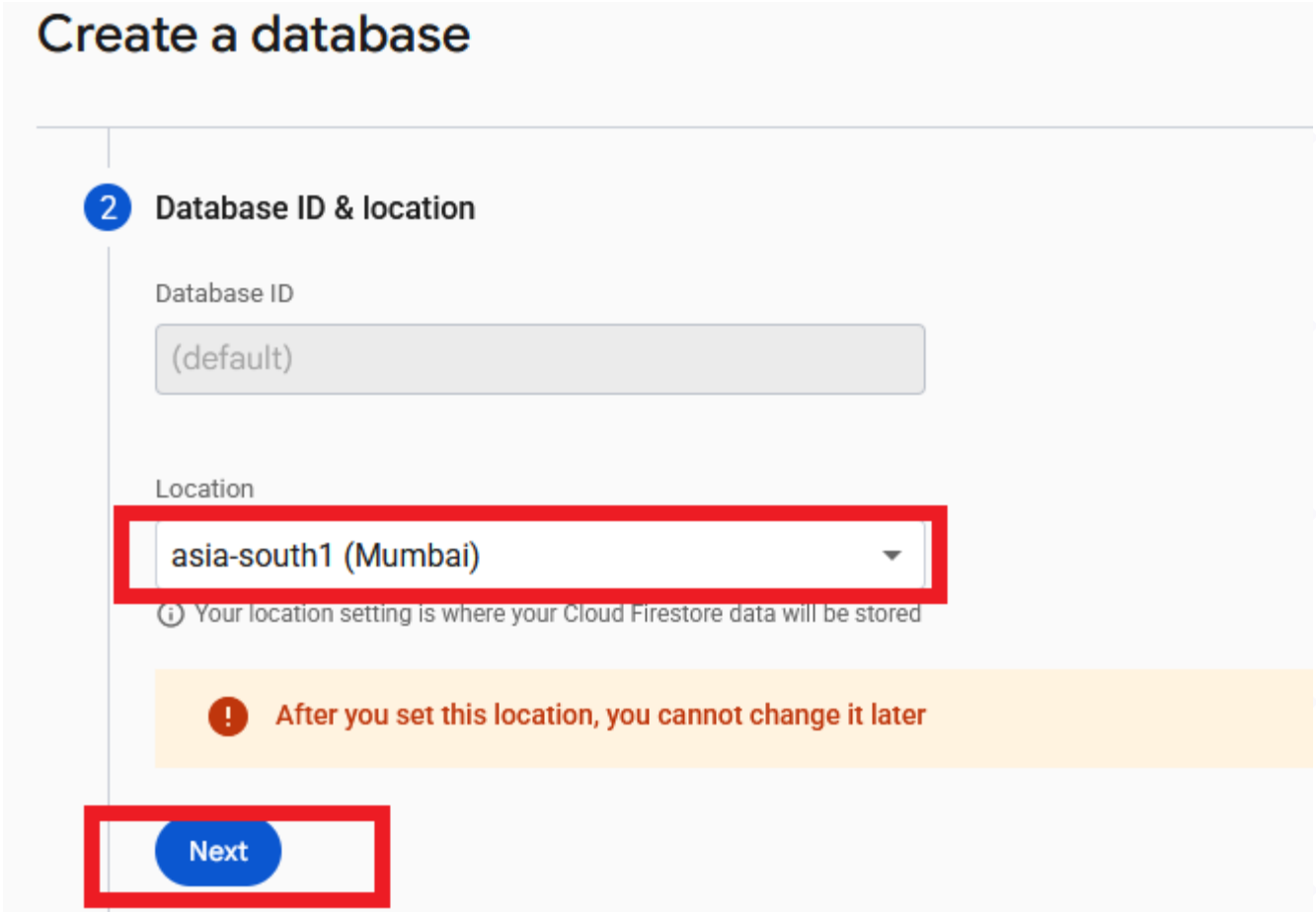
Шаг 3: Добавление базы данных Firestore



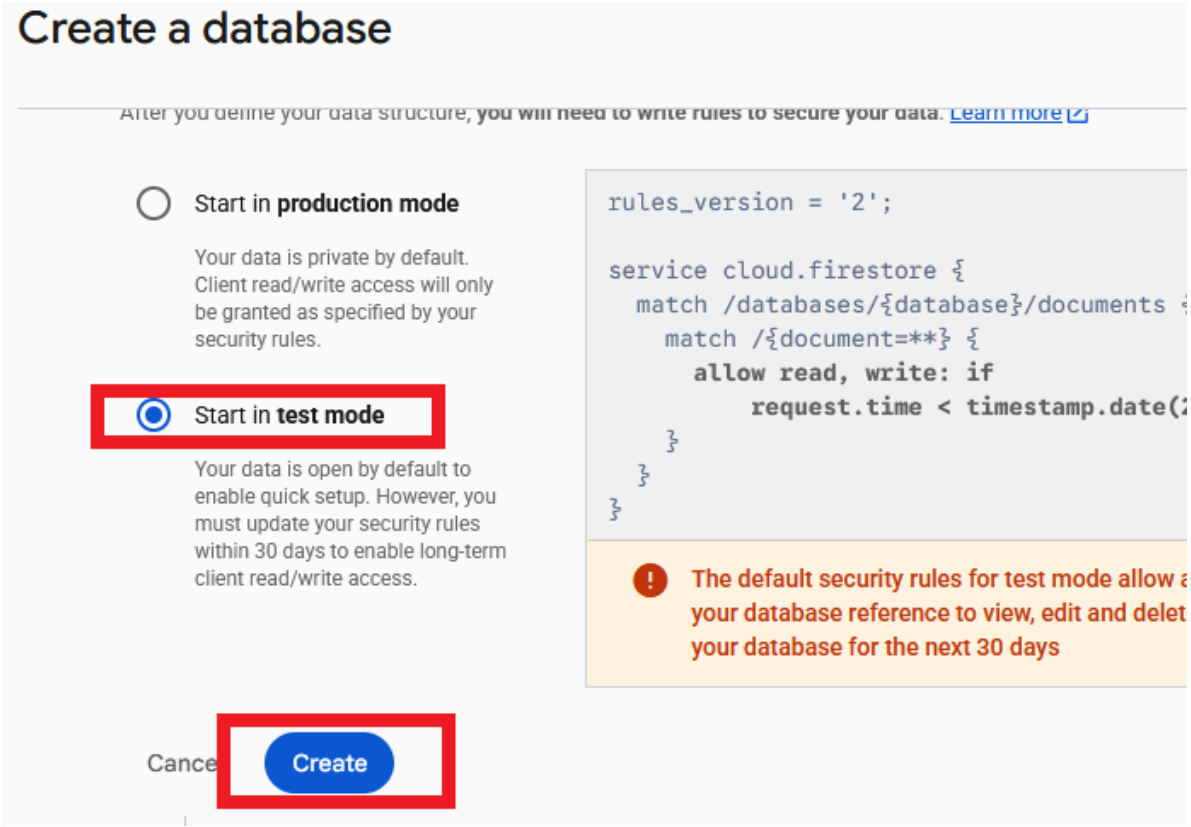




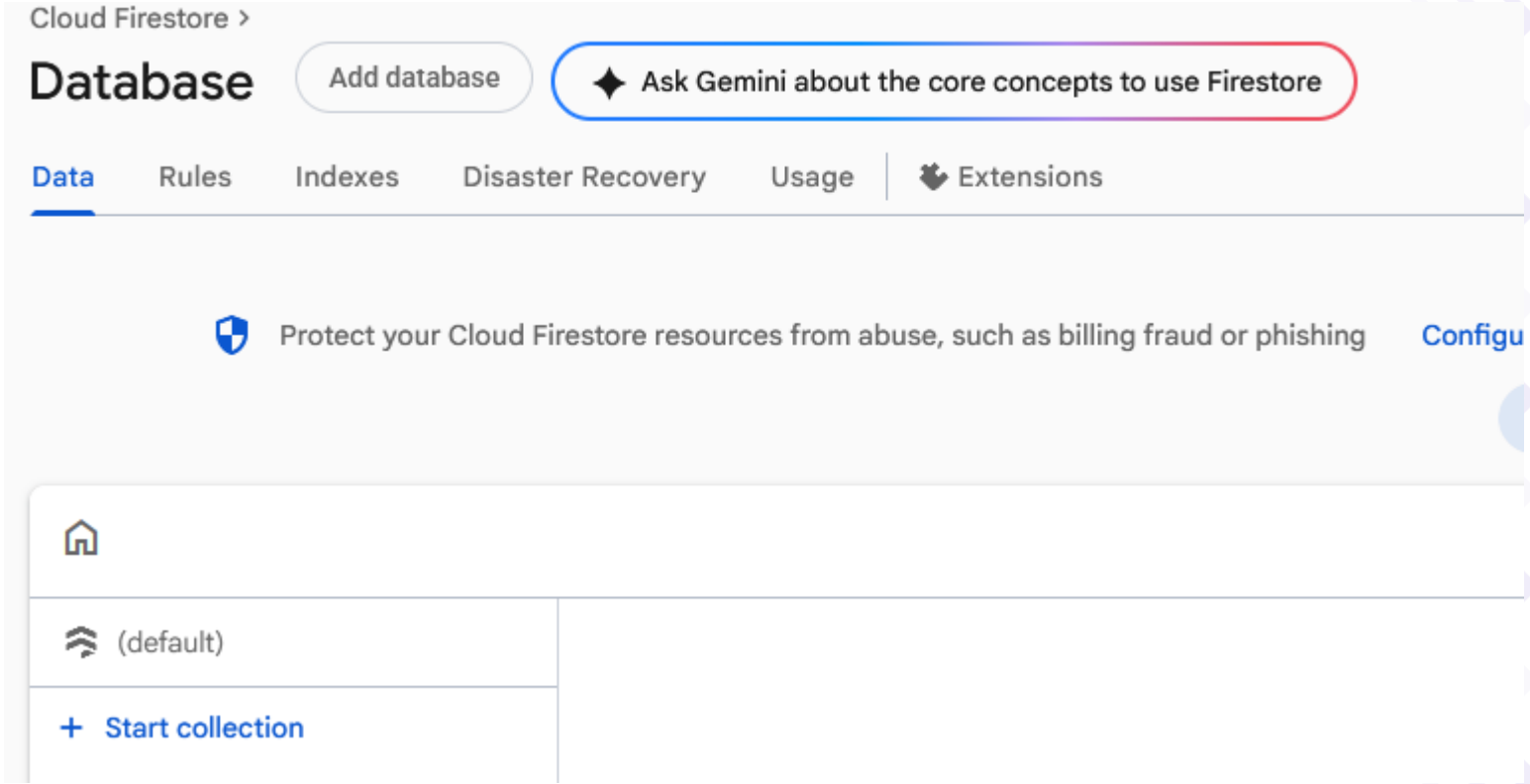




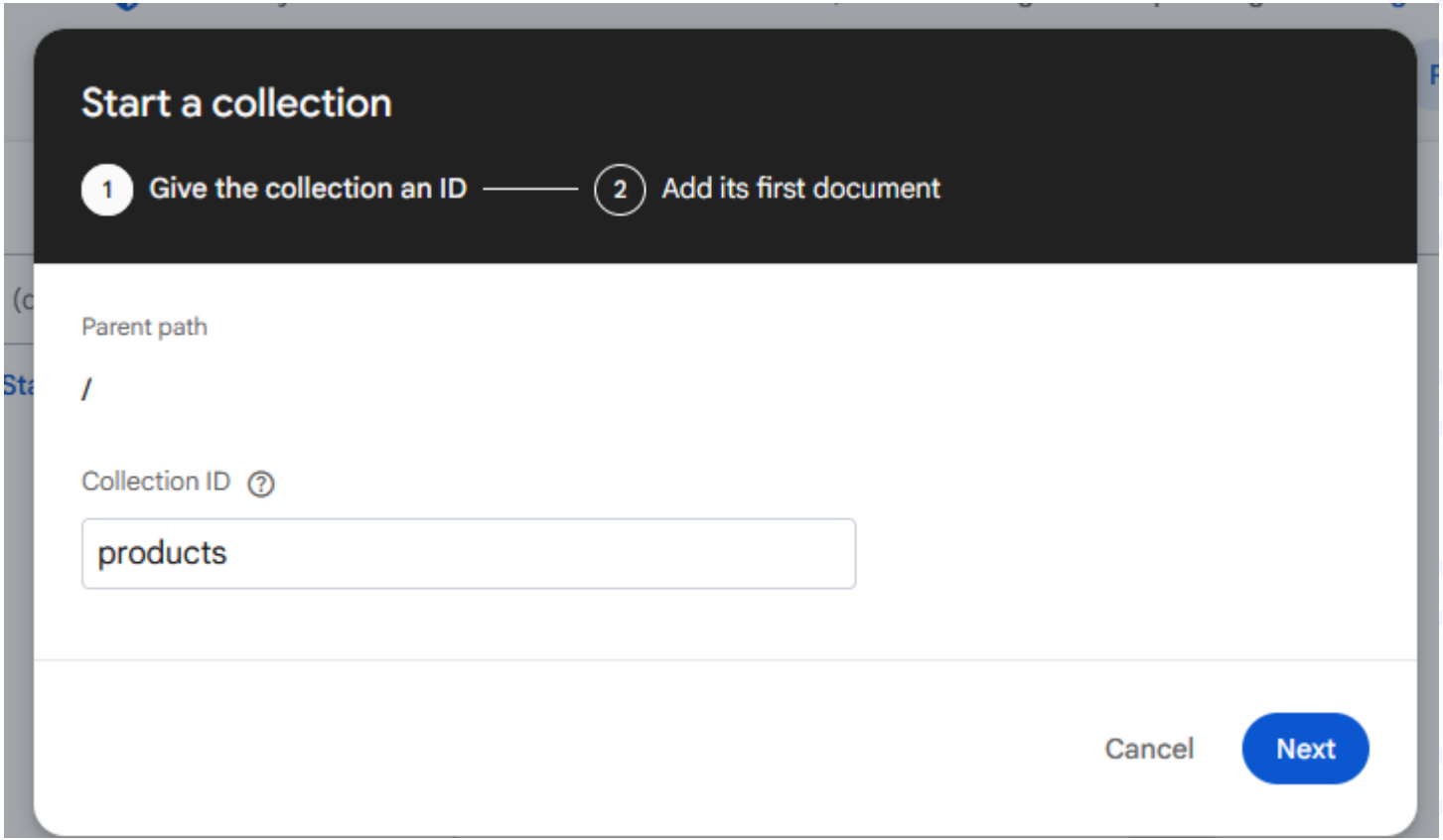
Начинаем в тестовом режиме (ДЛЯ УЧЕБЫ)



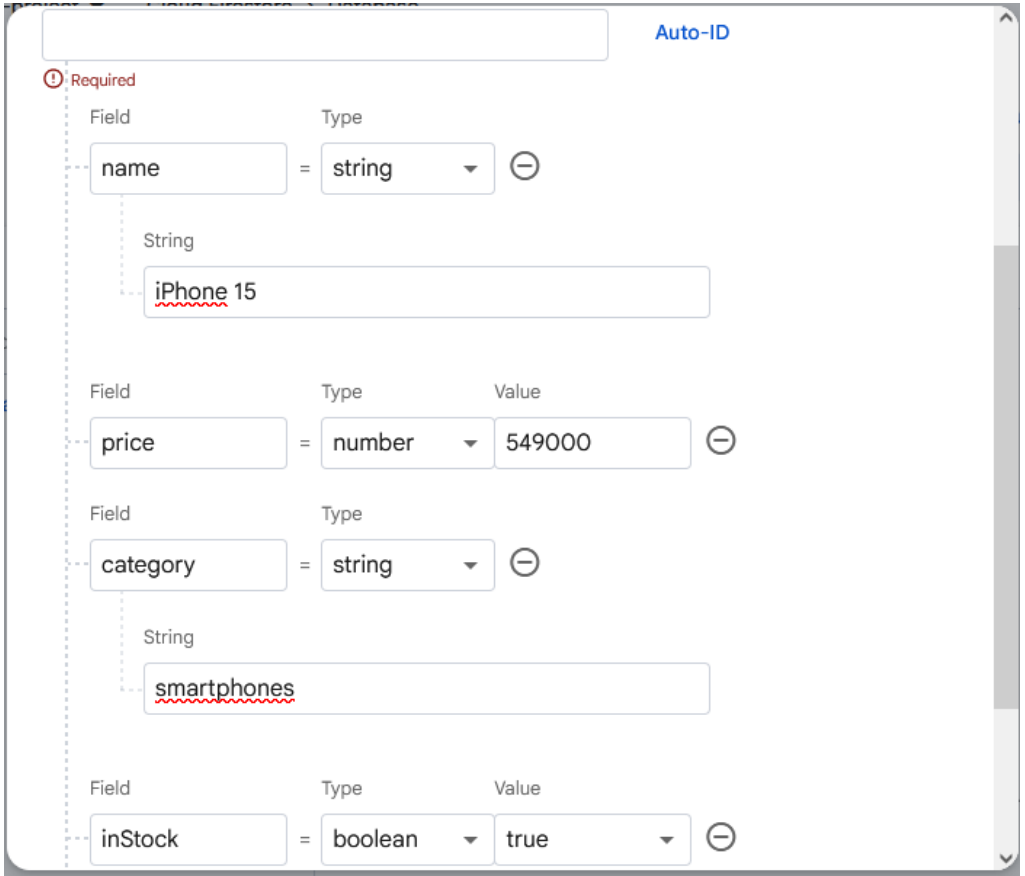
Шаг 4: Проверка созданной базы



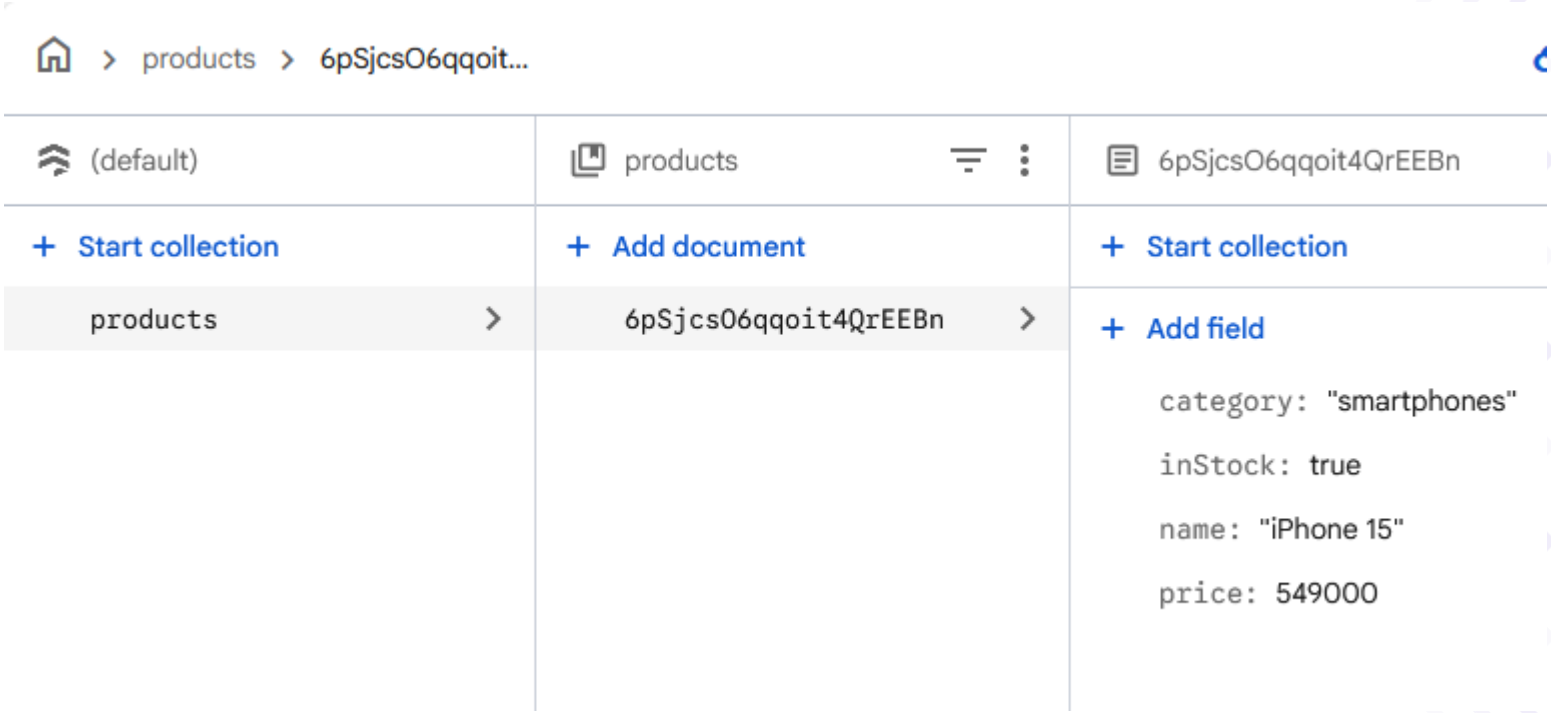
Шаг 5: Добавление первого товара



Добавляем первый документ



Результат:

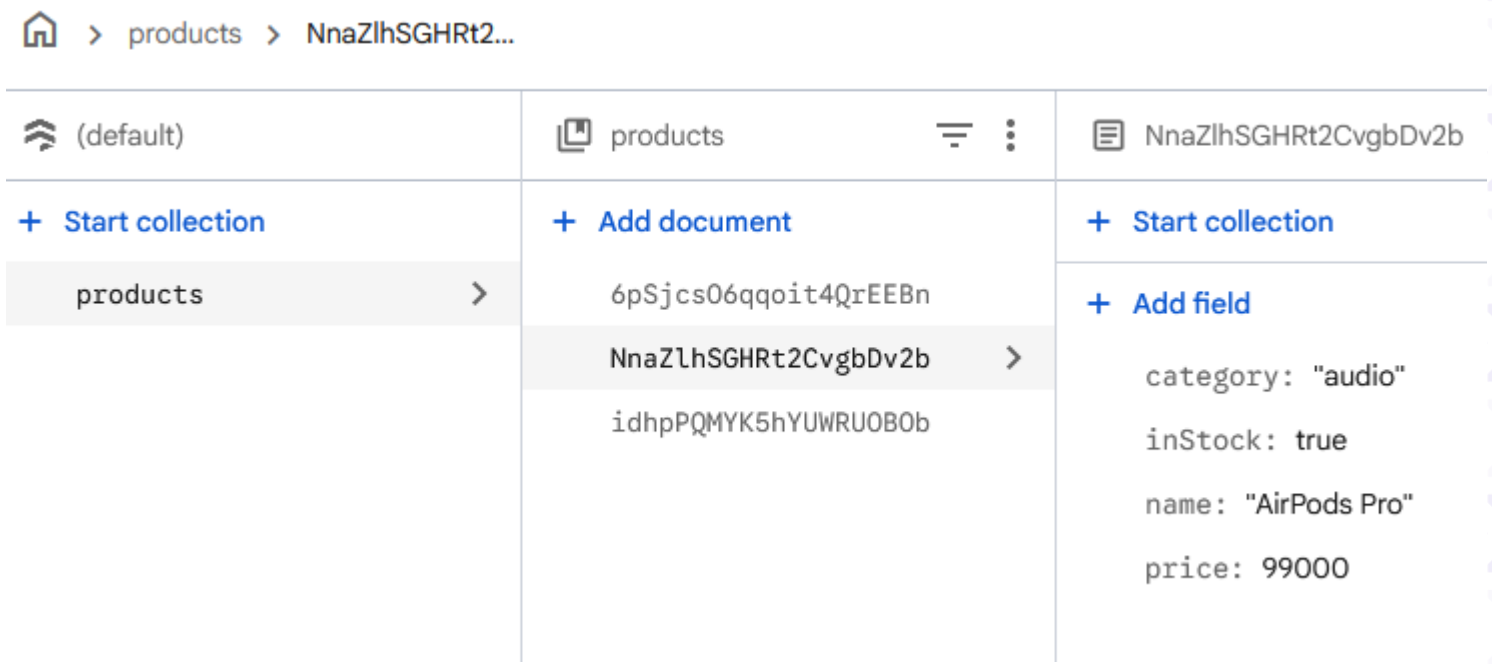


Добавляем ещё товары:

MacBook Pro, цена: 1999, категория: laptops

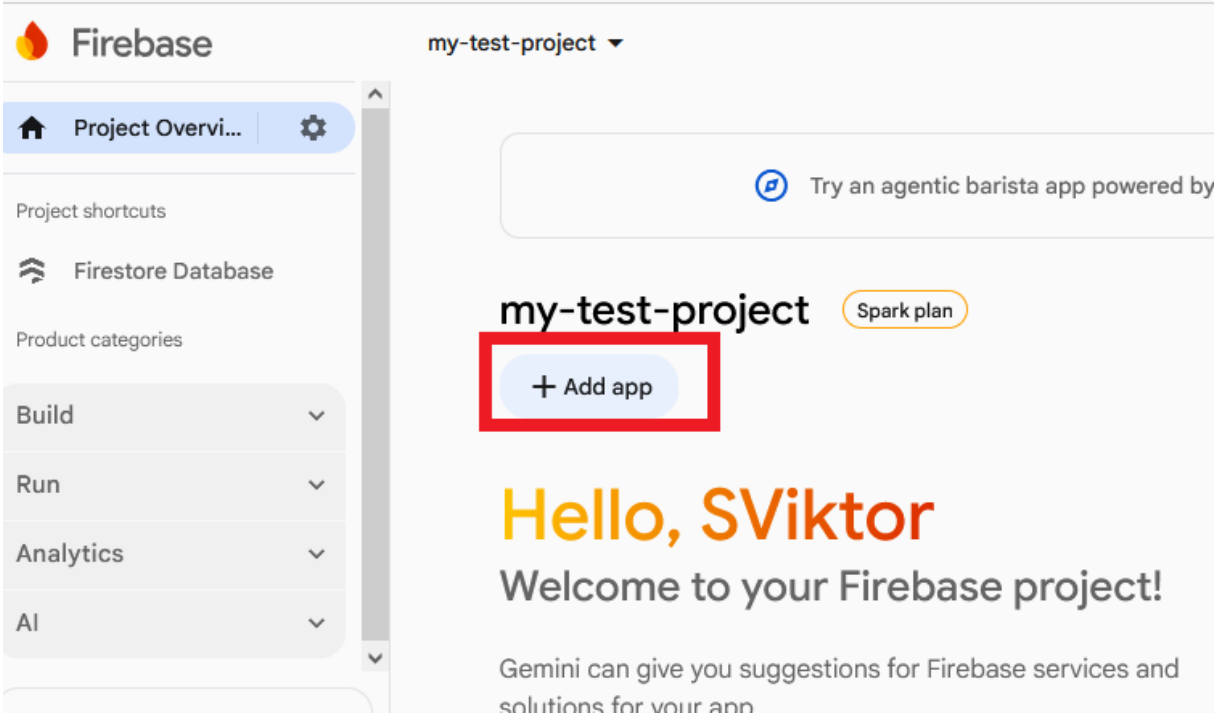
AirPods Pro, цена: 249, категория: audio

Результат:

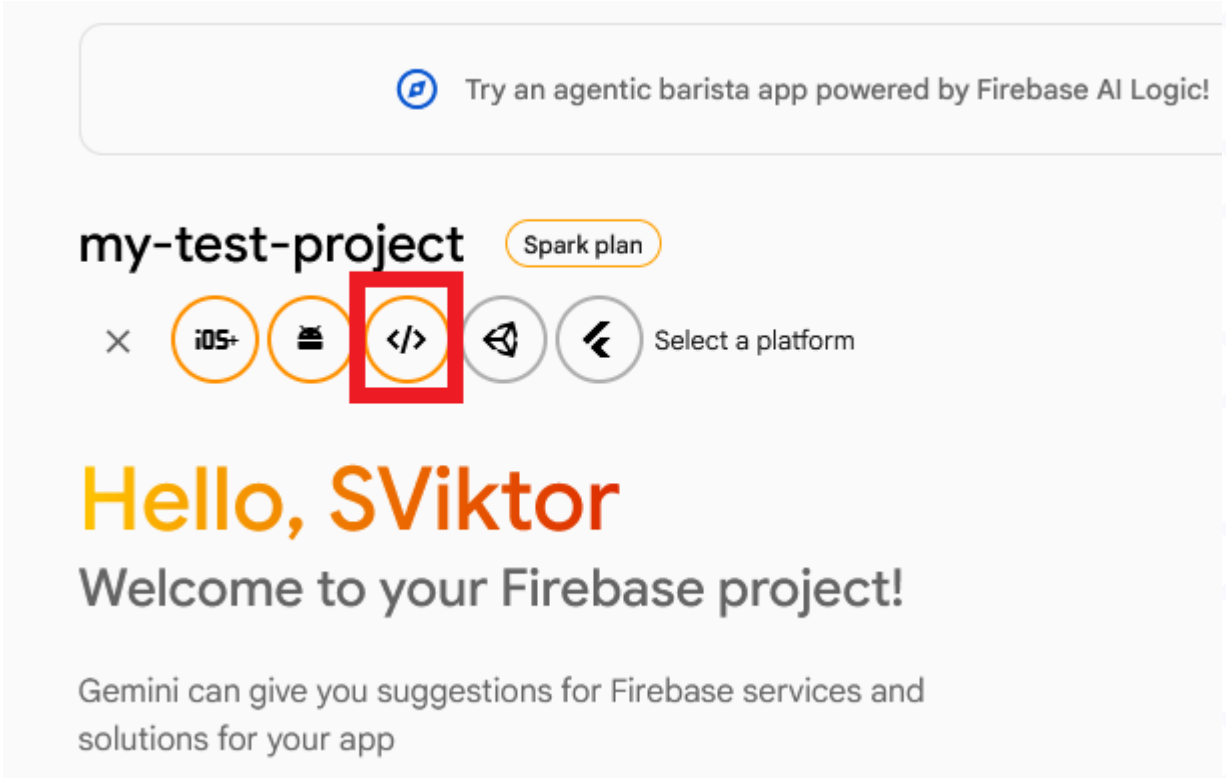


Шаг 7: Получение конфигурации для кода

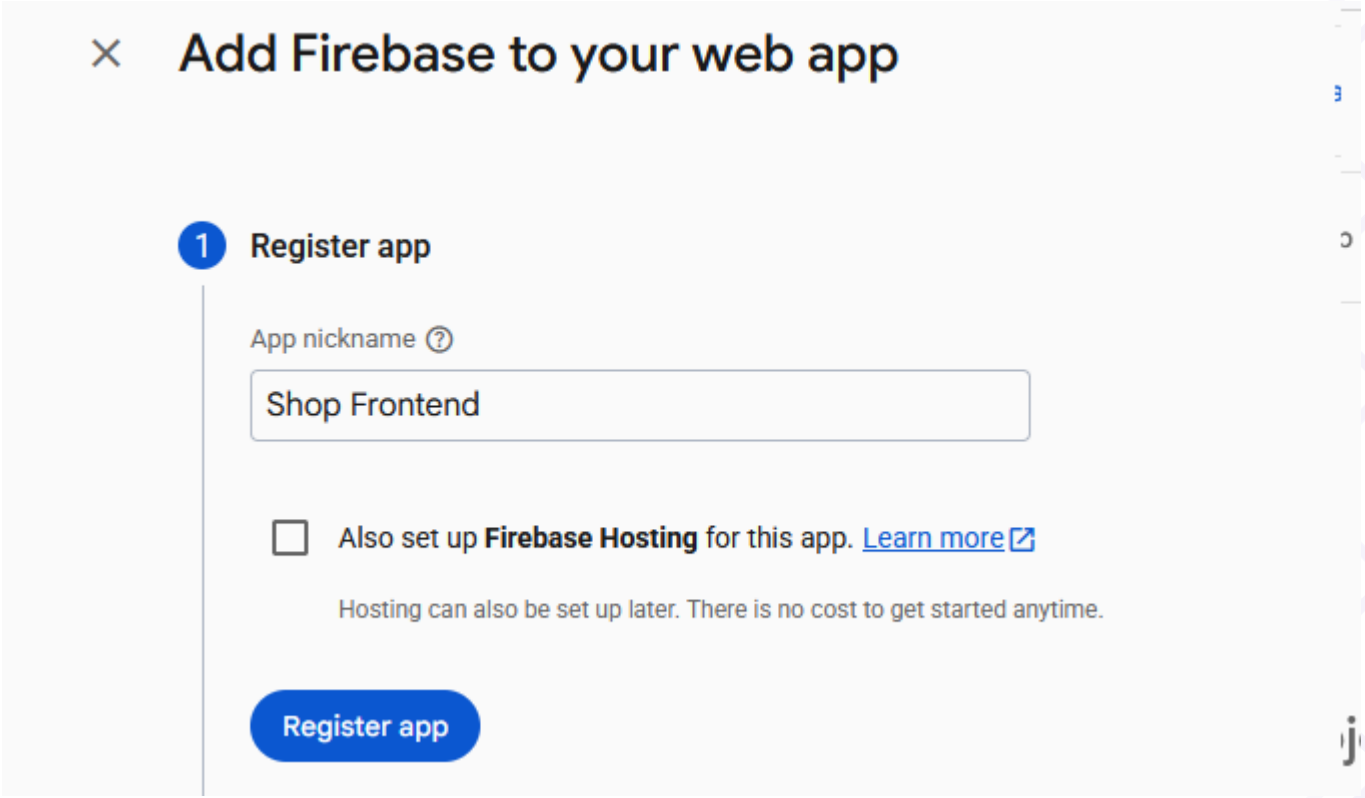
Возвращаемся на главную => Выбираем проект => добавить приложение



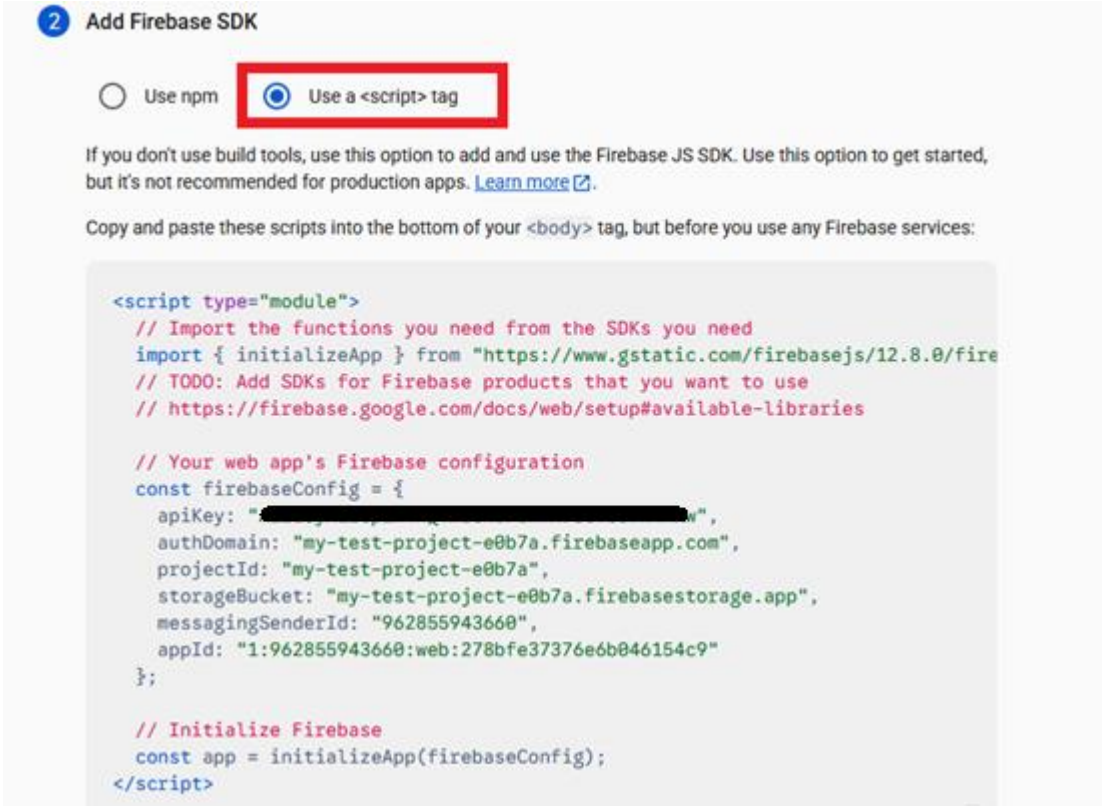
Выбираем веб-приложение:



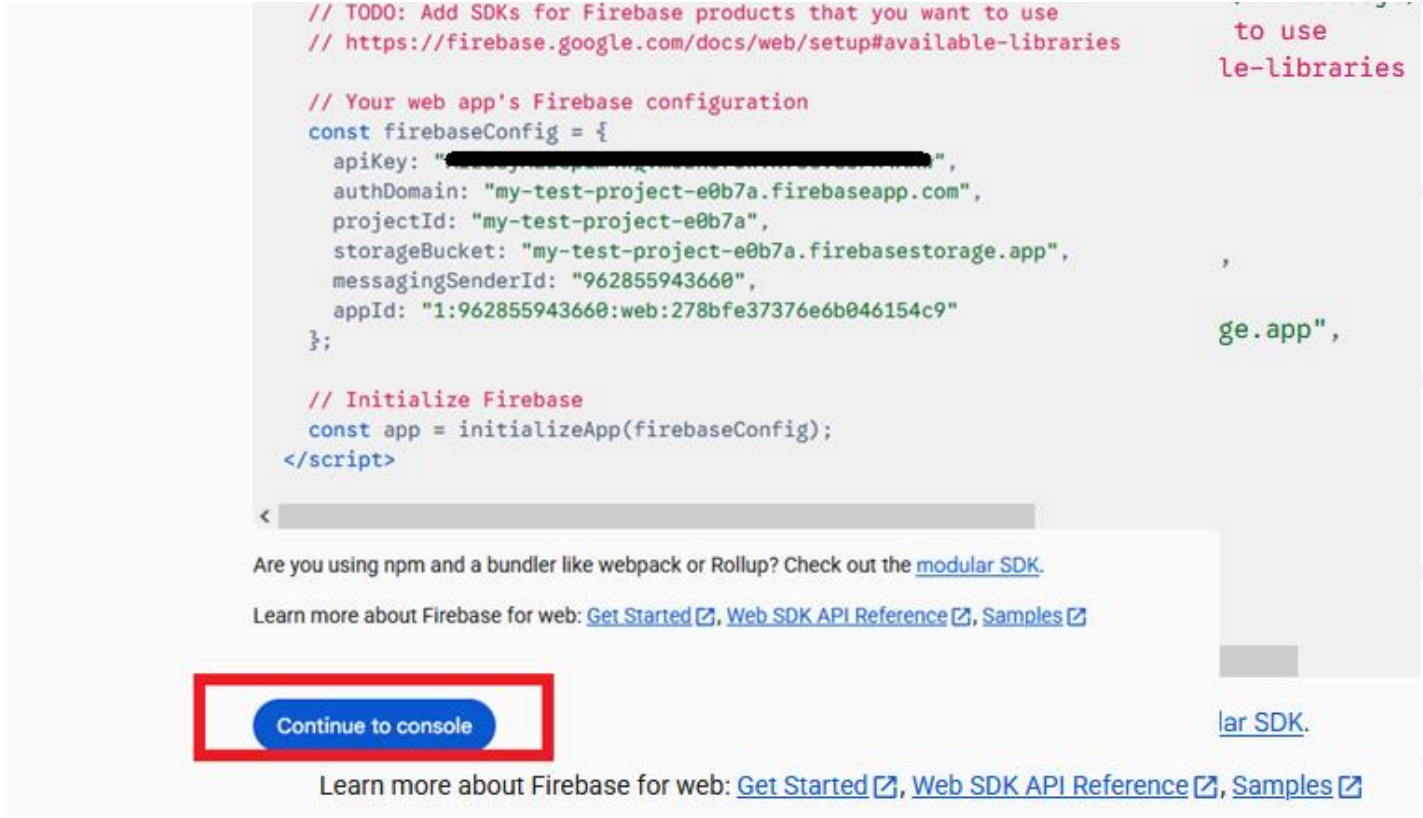
Вводим название приложения



Копируем и сохраняем код в отдельный файл или блокнот



Завершаем настройку



Измените правила (для разработки)

1. Firebase Console → Ваш проект
2. Слева в меню: ▶ Build (Сборка)
3. Развернуть: Firestore Database
4. Появится подменю:
   - └──── Data (Данные)
   - └──── Rules (Правила) ← ВОТ ОНО!
   - └──── Indexes (Индексы)

my-test-project

Cloud Firestore >

Database

Add database

Ask Gemini about the core concepts to use Firestore

Data | Rules | Indexes | Disaster Recovery | Usage | Extensions

Guard your data with rules that define who has access to it and how it is structured

Заменить

```
1  rules_version = '2';
2
3  service cloud.firestore {
4    match /databases/{database}/documents {
5      match /{document=**} {
6        allow read, write: if false;
7      }
8    }
9  }
```

Замените существующие правила на:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if true; // Разрешаем ВСЁ
    }
  }
}
```

5. Как подключить Firebase к веб-приложению на JavaScript?

Три способа подключения:

- CDN (самый простой, для новичков)
- npm модули (для современных проектов)
- ES модули (самый современный)

CDN (Content Delivery Network) — сеть доставки контента. Позволяет подключать библиотеки прямо из интернета без скачивания.

Шаг 1: Подключение Firebase SDK

Подключаем Firebase SDK через CDN:

```
<script src="https://www.gstatic.com/firebasejs/12.8.0/firebase-app-compat.js"></script>
<script src="https://www.gstatic.com/firebasejs/12.8.0/firebase-firestore-compat.js"></script>
```

Важно:
Используем -compat версии (совместимость)
Подключаем ДВА скрипта: firebase-app и firebase-firestore
Код Firebase пишем после подключения SDK

Документация:
https://firebase.google.com/docs/web/modular-upgrade?hl=ru

Шаг 2: Конфигурация Firebase

Копируем из Firebase Console:

```
const firebaseConfig = {
    apiKey: "AIzaSyD...",
    authDomain: "ваш-проект.firebaseapp.com",
    projectId: "ваш-проект-id",
    storageBucket: "ваш-проект.appspot.com",
    messagingSenderId: "123456789012",
    appId: "1:123456789012:web:abc123def456"
};
```

Где взять:

- Firebase Console → Ваш проект
- Нажмите "Add app" → "Web"
- Скопируйте объект firebaseConfig

Шаг 3: Инициализация Firebase

Инициализируем Firebase приложение
const app = firebase.initializeApp(firebaseConfig);

Получаем доступ к Firestore
const db = firebase.firestore();
console.log("✅ Firebase подключен!");

Что происходит:

- firebase.initializeApp() — создаёт приложение Firebase
- firebase.firestore() — даёт доступ к базе данных

db — это наш "мост" к Firestore

Шаг 4: Выполнение запроса к Firestore

Метод get() для коллекции:
Получаем все документы из коллекции 'products'

```
const querySnapshot = await db.collection('products').get();
```

Что такое QuerySnapshot:

- Результат запроса к Firestore
- Содержит все найденные документы
- Имеет методы для работы с документами

Шаг 5: Работа с результатами

Преобразование документа:

```
querySnapshot.forEach((doc) => {
    // doc.id — ID документа (автоматически сгенерированный)
    // doc.data() — объект с данными документа

    console.log("ID документа:", doc.id);
    console.log("Данные документа:", doc.data());
});
```

Вывод в консоль браузера:

```
// Простой вывод в консоль
querySnapshot.forEach((doc) => {
    const product = doc.data();
    console.log(`${product.name}: $${product.price}`);
});
```

Вывод на HTML-страницу:

```
const container = document.getElementById('products');

querySnapshot.forEach((doc) => {
    const product = doc.data();

    // Создаём HTML для каждого товара
    container.innerHTML += `
        <div class="product">
            <h3>${product.name}</h3>
            <p>Цена: $${product.price}</p>
        </div>
    `;
});
```

Пример:

```
<body>
  <h1>Товары из Firebase</h1>
  <div id="output"></div>
  <!--  Подключение Firebase SDK через CDN -->
  <script src="https://www.gstatic.com/firebasejs/12.8.0/firebase-app-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/12.8.0/firebase-firestore-compat.js"></script>

  <script>
    // Конфигурация (замените на свою!)
    const firebaseConfig = {
      apiKey: "AIzaSyALEGp1m4WQVmJEJ7JWvK93cvSOFh4ARw",
      authDomain: "my-test-project-e0b7a.firebaseapp.com",
      projectId: "my-test-project-e0b7a",
      storageBucket: "my-test-project-e0b7a.firebasestorage.app",
      messagingSenderId: "962855943660",
      appId: "1:962855943660:web:278bfe37376e6b046154c9",
    };
```

```
// Подключение
const app = firebase.initializeApp(firebaseConfig);
const db = firebase.firestore();

console.log("Firestore подключен!");
```

```
    // Загрузка данных
    db.collection("products")
      .get()
      .then((snapshot) => {
        const output = document.getElementById("output");
        snapshot.forEach((doc) => {
          const data = doc.data();
          output.innerHTML += `<p>${data.name}: $${data.price}</p>`;
        });
      })
      .catch((error) => {
        console.error("Ошибка:", error);
        document.getElementById("output").innerHTML = "Ошибка загрузки";
      });
  </script>
</body>
```

Вывод по лекции:

- Поняли Firebase — облачный бэкенд для фронтендеров
- Создали проект в Firebase Console
- Настроили Firestore (документную БД)
- Подключили к HTML через CDN
- Вывели данные из облака на страницу

Контрольные вопросы:

- Что такое Firebase и зачем он фронтендеру?
- Чем Firestore отличается от SQL-баз?
- Как получить конфигурацию для подключения Firebase?
- Какие шаги для подключения Firebase к HTML через CDN?
- Как получить все товары из коллекции 'products'?
- Опишите иерархию Firestore на примере магазина

хекслет колледж

@HEXLY.KZ