React Native + Firebase

De Cero a tu Primera Aplicación Móvil Profesional

Tutorial paso a paso con código completo y ejemplos visuales

Tabla de Contenidos

1. Introducción: Lenguajes y *Frameworks* usados

En este tutorial se enseña a construir una aplicación móvil en *React Native* + *Firebase* desde cero, tratando de ser claro, visual y sin tener en cuenta un nivel de experiencia avanzado.

> **¿Qué se va a construir?**
>
> Una **aplicación móvil en *React Native* con *TypeScript* que se conecta con *Firebase***, gestiona datos en tiempo real y muestra información en una interfaz visual escalable.

Al finalizar este tutorial se aprenderá a

✅ Construir una app móvil funcional en *React Native* con *TypeScript*

✅ Configurar una conexión con *Firebase Firestore*

✅ Un modelo de datos (entidad Usuario) completo

✅ Operar CRUD *(Create, Read, Update, Delete)* básicamente para leer datos de la base de datos

✅ Construir una interfaz visual

✅ Gestionar arquitectura en capas

> **¿Por qué estas tecnologías?**

Componente	Tecnología	Versión	Propósito
Lenguaje	TypeScript	5.x	Tipado estático y mejor experiencia de desarrollo
***Framework* UI**	*React Native*	0.74.x	Crear interfaces móviles nativas multiplataforma
Plataforma	Expo	51.0.0	Simplificar desarrollo y *deployment*
Base de Datos	Firebase Firestore	10.x	Base de datos NoSQL en tiempo real en la nube

Tabla 1: *Stack* tecnológico completo del proyecto

Flujo de Trabajo

2. Librerías ORM utilizadas

¿Qué es ORM?

ORM (*Object-Relational Mapping*) es una técnica de programación que permite convertir datos entre sistemas incompatibles usando programación orientada a objetos (POO).

🔍 **Es decir…**

En lugar de escribir consultas SQL como:

```sql
SELECT * FROM users WHERE age > 18;
```

Se puede usar código orientado a objetos como:

```
User.find({ age: { $gt: 18 } })
```

¿Porque se necesita un ORM?

El ORM actúa como un puente entre la base de datos y el código de la aplicación, proporcionando las siguientes ventajas:

Ventaja	Descripción
Abstracción	Se trabaja con objetos en lugar de consultas SQL directas
Productividad	Reduce significativamente la cantidad de código repetitivo
Sostenibilidad	El código es más fácil de leer y mantener a largo plazo
Seguridad	Protege automáticamente contra ataques de inyección SQL
Portabilidad	Facilita cambiar de base de datos sin reescribir todo el código
Validación	Permite validar datos antes de guardarlos en la BD

Para Bases de Datos NoSQL (No Relacionales) como es el caso de Firebase, no se emplea tablas ni fila, ni columnas como SQL.

En su lugar, se organizan los datos en **colecciones** (equivalente a tablas) y cada una de ellas tiene **documentos** (equivalente a filas) que se almacenan en formato JSON.

Pero entonces, ¿que librería se usa que represente ese puente?

2.1. Librería utilizada en este proyecto

En la creación de la app en React Native el puente entre la misma y Firebase es el mismo SDK de Firebase:

> El **Firebase SDK** reemplaza al ORM tradicional proporcionando métodos para interactuar con Firestore de manera programática, aportando lo siguiente:
>
> 1. Abstracción en consultas CRUD
> 2. Consultas en tiempo real
> 3. Sincronización automática de datos

De esta manera se comparan los ORM tradicionales con el Firebase SDK objetivo en cuanto a las operaciones abstractas.

Aspecto	ORM SQL	Firebase SDK -> No SQL
Relaciones	Claves foráneas y JOINs	Referencias y datos embebidos
Consulta típica	`User.find({ where: { edad: MoreThan(18) } })`	`getDocs( query(collection(db, 'users'), where('edad', '>', 18)) )`

> 📦 **Firebase SDK v10.x:**

Módulos principales:

- `firebase/app` → Inicialización de Firebase
- `firebase/firestore` → Operaciones con la base de datos en Firestore
- `firebase/auth` → Autenticación de usuarios (opcional)
- `firebase/storage` → Almacenamiento de archivos (opcional)

Métodos clave de la base de datos (Firestore):

- `collection()` → Referencia a una colección
- `doc()` → Referencia a un documento específico
- `getDocs()` → Obtener múltiples documentos
- `getDoc()` → Obtener un documento único
- `addDoc()` → Crear un nuevo documento
- `updateDoc()` → Actualizar un documento existente
- `deleteDoc()` → Eliminar un documento

2.2. Instalación del Firebase SDK

Para instalar el Firebase SDK en el proyecto, se ejecuta el siguiente comando en la terminal:

```
npm install firebase
```

Esta librería incluye todos los módulos necesarios para trabajar con Firebase y su base de datos en Firestore, ademas de otros servicios que se verán mas adelante, como autenticación o *storage*.

3. Configuración y levantamiento de la base de datos

En esta sección se explica cómo configurar Firebase desde cero, crear un proyecto, habilitar Firestore y preparar la base de datos para la aplicación móvil.

> 📝 **Requisitos previos**
>
> - Una cuenta de Google (*Gmail*)
> - Conexión a Internet

3.1. Paso 1. Crear cuenta en Firebase

- Acceder a https://firebase.google.com
- Hacer clic en el botón **«Comenzar»** o **«Get Started»**
- Iniciar sesión con la cuenta de Google
- Aceptar los términos y condiciones de Firebase

3.2. Paso 2: Crear un nuevo proyecto en Firebase

En Firebase Console, hacer clic en **«Agregar proyecto»** o **«Add project»**. Posteriormente, aparecerá un asistente de configuración con 3 pasos

---

Luego, configurar el nombre del proyecto:

Campo	Valor sugerido
Nombre del proyecto	mi-app-mobile
ID del proyecto	mi-app-mobile-xxxxx (generado automáticamente)
Región	Seleccionar la más cercana

Hacer clic en **«Continuar»**

En el *dashboard* del proyecto, buscar la sección **«Comienza agregando Firebase a tu app»** Hacer clic en el ícono **</> (Web)**

No seleccionar Firebase Hosting

Hacer clic en «Registrar app»

Luego, se obtienen las credenciales de configuración: Aparecerá un código de configuración similar a este:

```
firebaseConfig = {
  apiKey: "AIzaSyXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  authDomain: "mi-app-mobile-xxxxx.firebaseapp.com",
  projectId: "mi-app-mobile-xxxxx",
  storageBucket: "mi-app-mobile-xxxxx.appspot.com",
  messagingSenderId: "123456789012",
  appId: "1:123456789012:web:abcdef123456"
};
```

> ⚠️ **IMPORTANTE: Guardar credenciales**
>
> COPIAR Y GUARDAR estas credenciales en un archivo de texto temporal. Se necesitarán más adelante para configurar la aplicación.
>
> Estas credenciales son únicas para el proyecto y no se pueden recuperar fácilmente después.

Hacer clic en **«Continuar a la consola»**

3.3. Paso 3: Habilitar Firestore Database

> 🗄️ **Creación de la base de datos**

En el menú lateral izquierdo de Firebase Console, buscar la opción ***«Firestore Database»*** Hacer clic en ***«Firestore Database»*** Se visualizará una pantalla de introducción

---

Hacer clic en el botón **«Crear base de datos»** o **«Create *database*»**

Seleccionar **«Iniciar en modo de prueba»** o ***«Start in test mode»***. Hacer clic en **«Siguiente»**

> ⚠️ **Nota de seguridad**

El modo de prueba deja la base de datos abierta por 30 días. Esto es solo para desarrollo. Antes de publicar la aplicación en producción, se deben configurar reglas de seguridad adecuadas.

Elegir la ubicación del servidor Firestore

💡 **Recomendación**

Seleccionar la ubicación geográfica más cercana a donde se encuentran los usuarios de la aplicación para reducir la latencia.

Hacer clic en **«Habilitar»** o **«Enable»**

3.4. Paso 4: Crear la primera colección

En la pantalla de Firestore, hacer clic en **«Iniciar colección»** o **«Start *collection*»**

Hacer clic en **«Siguiente»**

Agregar el primer documento: Ahora se debe agregar el primer documento a la colección users:

Luego, se agregan estos campos al documento:

Campo	Tipo	Valor de ejemplo
nombre	string	Juan Pérez
email	string	juan.perez@email.com
edad	number	25

3.5. Paso 5: Verificar reglas de seguridad

🔒 **Reglas de seguridad de Firestore**

Las reglas de seguridad controlan quién puede leer y escribir datos en Firestore.

En el menú superior de Firestore, hacer clic en la pestaña **«Reglas»** o ***«Rules»*** Se visualizará el editor de reglas

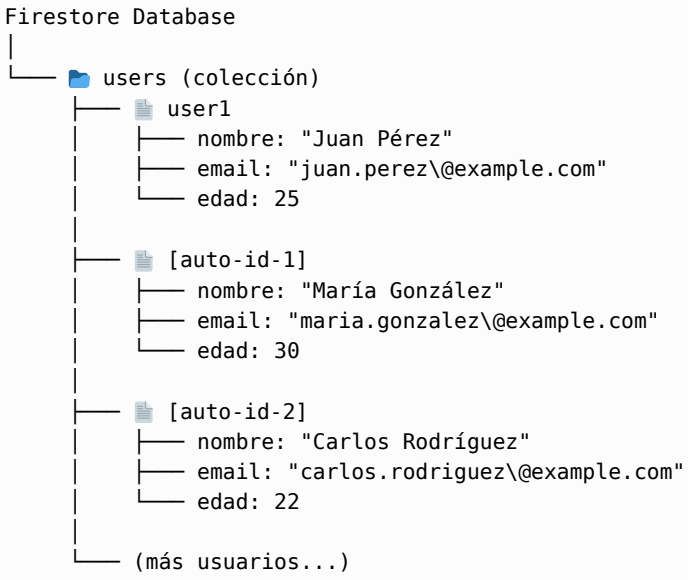
Si se configuró en modo de prueba (recomendado), las reglas serán similares a:

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.time < timestamp.date(2025, 2, 1);
    }
  }
}
```

`allow read, write:` Permite operaciones de lectura y escritura
`if request.time < ...` Solo hasta la fecha especificada (30 días)

Estructura final de la base de datos:

Estructura en Firestore:



4. Archivos yml

En proyectos tradicionales con bases de datos SQL, es común utilizar archivos YAML (.yml) para configurar conexiones a bases de datos, variables de entorno y otros parámetros.

Sin embargo, en este proyecto específico con React Native, Expo y Firebase, NO se requieren archivos YAML.

🔍 ¿Por qué no se necesitan archivos YAML?

- Firebase se configura mediante código JavaScript/TypeScript, no archivos YAML
- Expo utiliza archivos JSON (app.json y eas.json) para su configuración

4.1. Archivos de configuración usados

Aunque no se usan archivos .yml o .yaml, el proyecto requiere los siguientes archivos de configuración:

Archivo	Formato	Propósito
app.json	JSON	Configuración principal de Expo
package.json	JSON	Dependencias y scripts del proyecto
tsconfig.json	JSON	Configuración de TypeScript
firebase.config.ts	TypeScript	Credenciales de Firebase

5. “Hola Mundo”