

Universidad de la Sierra Juárez

inter natura et scientia harmonia ~ Dua ztee-riu lanii taa rhenii lanagua yubiriu

Ixtlán de Juárez



CURSO DE BASE DE DATOS DE NO RELACIONALES

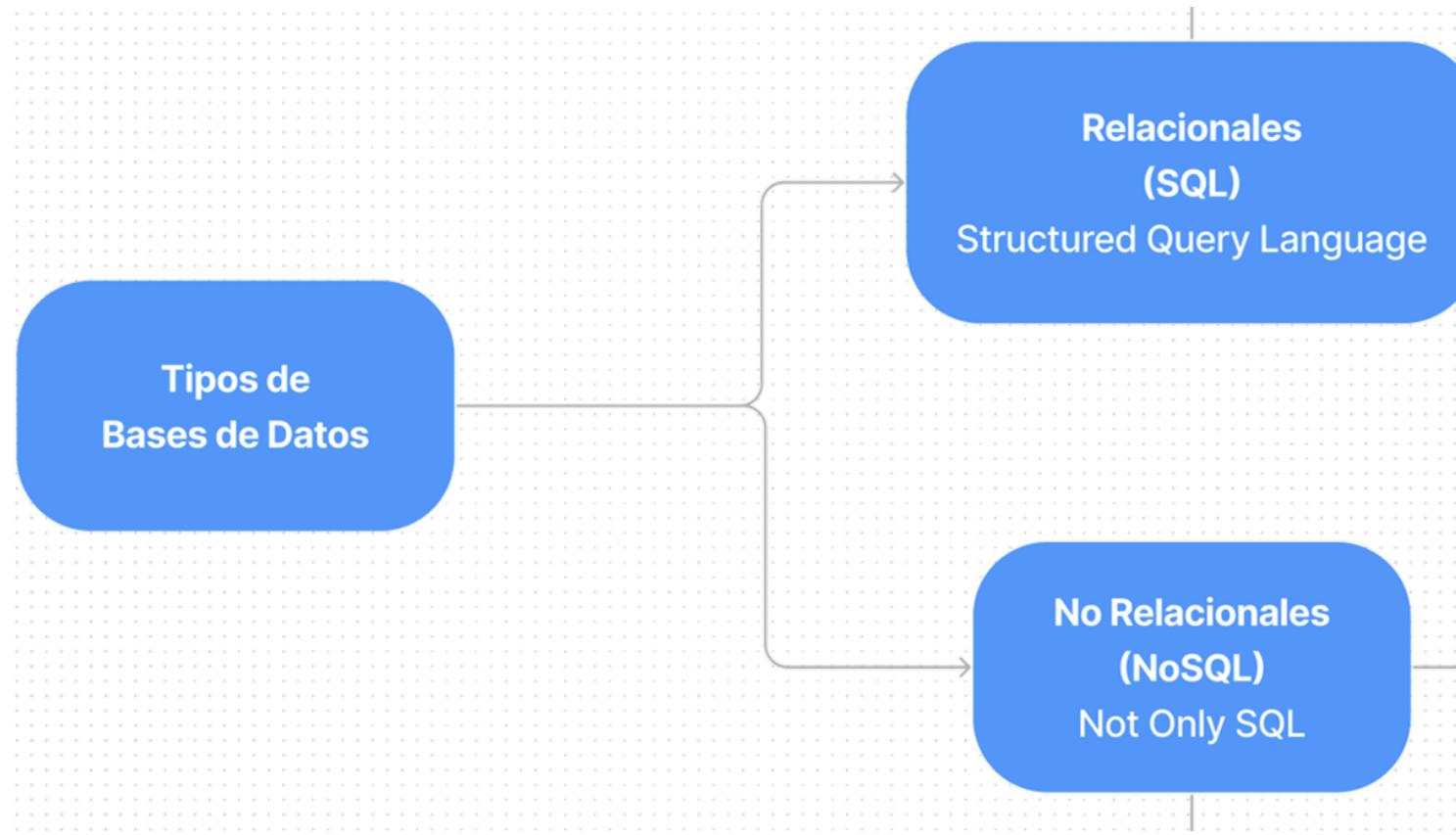
NoSQL
(Not Only SQL)



Presenta:

M.I.S. Celiflora Diaz Jiménez

Introducción



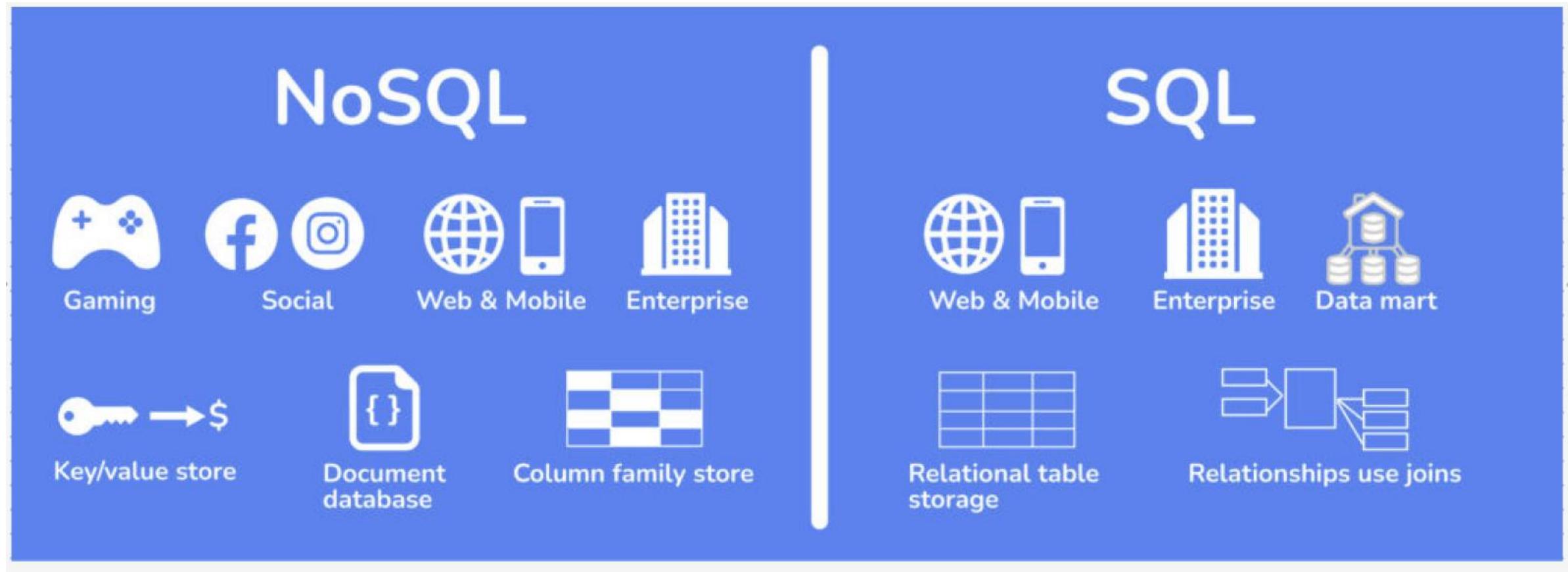
Introducción



- NoSQL (Not Only SQL) es un término que se utiliza para describir un conjunto de sistemas de gestión de bases de datos que utilizan modelos de datos no relacionales, en contraposición a los sistemas de gestión de bases de datos relacionales (RDBMS) tradicionales.

Introducción

¿Cuándo sería importante aprender NoSQL?



Introducción



¿Por qué es importante?

- **Escalabilidad:** Permiten crecer rápidamente y manejar grandes volúmenes de datos y tráfico sin perder rendimiento.
- **Flexibilidad del modelo de datos:** No dependen de esquemas rígidos. Se pueden agregar o modificar datos sin cambiar estructuras completas.
- **Alta velocidad:** Ofrecen un rendimiento muy alto gracias a modelos más simples y consultas optimizadas.
- **Alta disponibilidad:** Replican los datos en varios servidores, garantizando funcionamiento continuo incluso si uno falla.
- **Ideales para Big Data:** Manejan eficientemente datos masivos, no estructurados o semiestructurados que no encajan en modelos tradicionales.

Introducción

¿Cuándo usar NoSQL?



- **Big Data:** Ideal para manejar grandes volúmenes de datos no estructurados, como los generados por redes sociales.
- **Carga distribuida:** Escalan horizontalmente, permitiendo agregar más servidores cuando aumenta la demanda en aplicaciones web y móviles.
- **Flexibilidad del esquema:** Permiten modificar o agregar datos sin necesidad de cambiar un esquema rígido, útil en aplicaciones que evolucionan rápido.
- **Alta disponibilidad:** Ofrecen replicación y tolerancia a fallos, lo que las hace adecuadas para aplicaciones críticas que deben estar siempre activas.
- **Procesamiento en tiempo real:** Capaces de analizar grandes cantidades de datos al instante, como en sistemas de detección de fraudes.

Introducción



Historia y evolución de NoSQL

- **1998:** Surge *mSQL*, precursor de MySQL. Aunque MySQL se vuelve muy popular, su modelo relacional y esquemas rígidos limitan el manejo de datos masivos y no estructurados.
- **2000–2007:** El crecimiento de la web y las aplicaciones móviles genera enormes volúmenes de datos no estructurados. Se vuelve evidente la necesidad de modelos más flexibles.
- **2007:** Google publica el artículo sobre **Bigtable**, su base de datos distribuida basada en columnas. Este modelo inspira el desarrollo de las primeras bases de datos NoSQL modernas.

Introducción

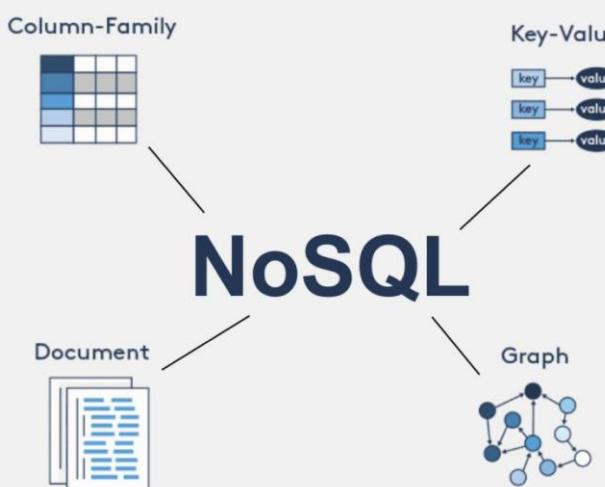


Historia y evolución de NoSQL

- **2009:** Johan Oskarsson acuña el término “**NoSQL**” para describir bases de datos no relacionales y sin SQL como lenguaje principal. Ese mismo año aparece **MongoDB**, que se convierte en uno de los sistemas NoSQL más populares.
- **2010 en adelante:** Surgen sistemas como **Cassandra**, **Couchbase**, **FireEstore**, **HBase**, **Redis**, entre otros. Evolucionan incorporando:
 - ✓ escalabilidad automática,
 - ✓ alta disponibilidad,
 - ✓ integración con la nube,
 - ✓ procesamiento en tiempo real.

Bases de datos NoSQL

Tipos de bases de datos NoSQL



- **Llave–Valor** Almacenan información como pares $llove \rightarrow valor$. Son extremadamente rápidas y simples.
 - Ejemplos: Redis, Riak, DynamoDB.
- **Orientadas a Columnas** Guardan los datos por columnas, permitiendo alta escalabilidad y manejo de grandes volúmenes.
 - Ejemplos: Cassandra, HBase.
- **De Grafos** Representan datos como nodos y relaciones. Ideales para redes sociales y análisis de conexiones.
 - Ejemplos: Neo4j, OrientDB, ArangoDB.
- **De Documentos** Almacenan información en documentos JSON/BSON flexibles y sin esquemas rígidos.
 - Ejemplos: MongoDB, CouchDB, Couchbase.

Bases de datos de llave-valor

(Key-value)



- Almacenan información como pares **llave → valor**, donde la llave es única y permite recuperar el valor asociado.
- Su estructura es **simple y plana**, sin relaciones ni esquemas rígidos, lo que les da **alta velocidad y gran escalabilidad**.
- Permiten operaciones básicas como **insertar, actualizar, eliminar y recuperar** valores de forma muy eficiente.
- Son ideales para **caché, sesiones, metadatos, colas** y cualquier escenario que requiera acceso rápido.
- **Ejemplos:** Redis, Riak, Amazon DynamoDB.

Redis

The Redis website features a dark-themed header with the Redis logo, a search bar, and navigation links for Products, Recursos, Documentos, Precios, Inicia sesión, Reserva una reunión, and Prueba Redis.

Redis para IA
Somos la capa de memoria rápida para chatbots y agentes de IA. **Consigue hoy** mismo herramientas listas para crear apps de IA.

- ✓ Base de datos vectorial
- ✓ Memoria para agentes de IA
- ✓ Búsqueda semántica

Redis LangCache
Reduce la latencia y el coste de los LLM con **caché semántica totalmente gestionada**.

Caché
Obvio.

Redis Insight
Desarrolla, depura y visualiza con nuestra interfaz **gráfica gratuita y herramienta para devs**.

Redis Flex
Almacena en caché 5 veces más, sin coste adicional.

Redis Query Engine
Ejecuta consultas de datos potentes y búsquedas en **tiempo real**.

Redis Data Integration
Sincroniza los datos de tu base existente al instante.

Bases orientadas a columnas

(Column-
Family)



Almacenan los datos **por columnas en lugar de filas**, lo que permite leer solo las columnas necesarias y acelerar las consultas.

Ofrecen **alta compresión y eficiencia**, ya que cada columna puede optimizarse y comprimirse de forma independiente.

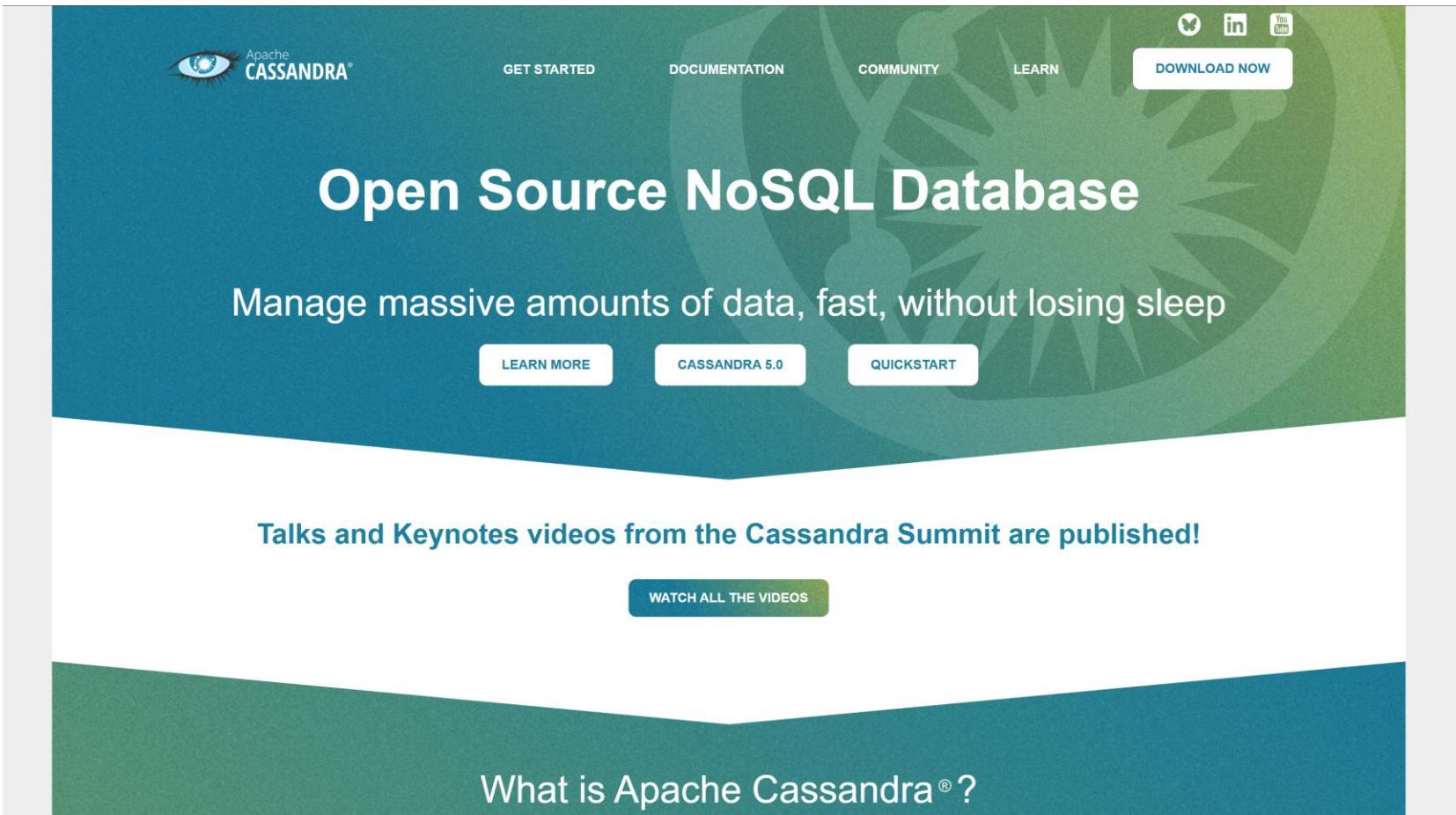
Son ideales para **consultas analíticas y agregaciones** (SUM, COUNT, AVG) en grandes volúmenes de datos.

Permiten **escalabilidad horizontal**, distribuyendo datos entre múltiples nodos para soportar cargas masivas.

Ofrecen **flexibilidad de esquema**, permitiendo agregar o modificar columnas sin afectar el sistema.

Ejemplos: Apache Cassandra, Apache HBase, ScyllaDB.

Apache Cassandra



The screenshot shows the official Apache Cassandra website. At the top, there's a dark header with the Apache Cassandra logo (an eye icon) and the word "CASSANDRA". Below the header, a large green banner features the text "Open Source NoSQL Database" and "Manage massive amounts of data, fast, without losing sleep". It also includes three buttons: "LEARN MORE", "CASSANDRA 5.0", and "QUICKSTART". Above the banner, there are navigation links: "GET STARTED", "DOCUMENTATION", "COMMUNITY", "LEARN", and a "DOWNLOAD NOW" button with social media icons for GitHub, LinkedIn, and YouTube. A callout at the bottom of the banner says "Talks and Keynotes videos from the Cassandra Summit are published!" with a "WATCH ALL THE VIDEOS" button. The bottom of the page has a dark teal footer with the text "What is Apache Cassandra® ?"

Apache CASSANDRA®

GET STARTED DOCUMENTATION COMMUNITY LEARN DOWNLOAD NOW

Open Source NoSQL Database

Manage massive amounts of data, fast, without losing sleep

LEARN MORE CASSANDRA 5.0 QUICKSTART

Talks and Keynotes videos from the Cassandra Summit are published!

WATCH ALL THE VIDEOS

What is Apache Cassandra® ?

Bases orientadas a grafos

(Graph)



- Almacenan los datos como **nodos** (entidades) y **aristas** (relaciones), lo que permite modelar conexiones complejas de forma natural.
- Tanto nodos como relaciones pueden tener **propiedades**, facilitando representar información detallada.
- Están optimizadas para **consultar y analizar relaciones**, siguiendo rutas y patrones de forma muy eficiente.
- Ofrecen **flexibilidad y escalabilidad**, ideales para grandes volúmenes de datos con muchas conexiones.
- Son muy útiles para **descubrir patrones**, hacer **recomendaciones**, analizar **redes sociales** y detectar **fraudes**.
- Utilizan lenguajes de consulta especializados como **Cypher** (Neo4j) o **Gremlin**.
- **Ejemplos:** Neo4j, Amazon Neptune, JanusGraph, ArangoDB.

neo4j

Iniciar sesión en Aura Fogonadura ▾ Compañía ▾ Apoyo

neo4j Productos ▾ Casos de uso ▾ Desarrolladores ▾ Sistemas de IA Aprender ▾ Precios **Contáctenos** **Empieza gratis**

LA PLATAFORMA DE INTELIGENCIA GRÁFICA LÍDER EN EL MUNDO

Cree aplicaciones inteligentes e IA para Resolución de identidad

Datos preparados para IA por diseño

Empezar a construir **Más información**

Edificio de 300.000 desarrolladores

Más de 80 clientes de Fortune 100

Ecosistema de **más de 170 socios**



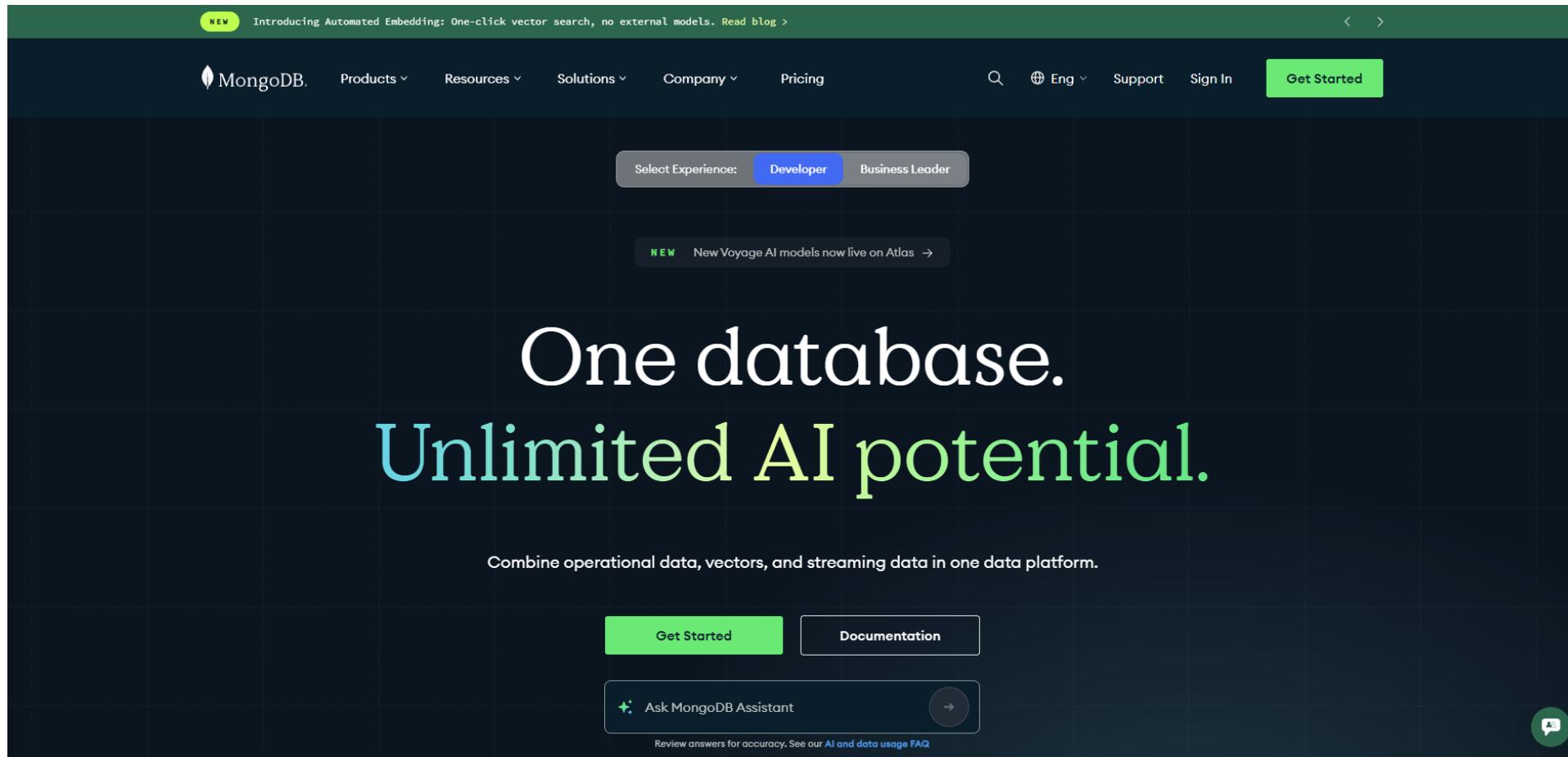
Bases orientadas a documentos

(Document)



- Almacenan la información en **documentos** (JSON, BSON, etc.), cada uno **autocontenido** y con estructura flexible.
- No requieren un **esquema fijo**, permitiendo que cada documento tenga campos distintos y se adapte a cambios fácilmente.
- Ofrecen **consultas eficientes** mediante índices y permiten búsquedas rápidas dentro de documentos complejos.
- Diseñadas para **escalar horizontalmente**, agregando nodos para manejar grandes volúmenes de datos.
- Admiten **replicación** para alta disponibilidad y tolerancia a fallos.
- Se integran fácilmente con múltiples lenguajes y frameworks.
- **Ejemplos:** MongoDB, Couchbase, CouchDB, Amazon DocumentDB, Firebase

MongoDB



The screenshot shows the MongoDB homepage with a dark background. At the top, there's a green header bar with a "NEW" badge and the text "Introducing Automated Embedding: One-click vector search, no external models. Read blog >". Below the header is a navigation bar with links for MongoDB, Products, Resources, Solutions, Company, Pricing, a search icon, language selection ("Eng"), Support, Sign In, and a prominent green "Get Started" button. A dropdown menu titled "Select Experience" offers options for "Developer" (which is selected) and "Business Leader". A banner at the bottom of the header area announces "New Voyage AI models now live on Atlas →". The main visual features large, bold text: "One database." above "Unlimited AI potential." in a light blue color. Below this, a subtitle reads "Combine operational data, vectors, and streaming data in one data platform." Two buttons are present: a green "Get Started" button and a white "Documentation" button. At the bottom, there's a "Ask MongoDB Assistant" input field with a blue arrow icon, a note about AI and data usage FAQ, and a small circular icon with a speech bubble and "AI" text.

NEW Introducing Automated Embedding: One-click vector search, no external models. Read blog >

MongoDB Products Resources Solutions Company Pricing

Select Experience: **Developer** Business Leader

NEW New Voyage AI models now live on Atlas →

One database. Unlimited AI potential.

Combine operational data, vectors, and streaming data in one data platform.

Get Started Documentation

Ask MongoDB Assistant

Review answers for accuracy. See our [AI and data usage FAQ](#)

Firebase: Plataforma Integral para Aplicaciones Modernas



Origen de Firebase

Fundación como startup (2011)

Nace para simplificar el desarrollo con servicios backend en la nube.

Adquisición por Google (2014)

Se integra al ecosistema de desarrollo de Google, ampliando capacidades.

Plataforma actual serverless

Conjunto de herramientas para desarrollar, gestionar y escalar apps móviles y web.





¿Qué es Firebase y para qué sirve?

Firebase es una plataforma de desarrollo de aplicaciones web y móviles mantenida por Google.

Firebase es un *backend as a service* cuyo objetivo es proporcionar un conjunto de servicios de backend, como autenticación, análisis de datos, almacenamiento de datos, notificaciones, bases de datos.

Propósito y enfoque de Firebase



Simplificación del backend



Manejo de datos en tiempo real



Análisis integrado

Servicios de: Base de datos en tiempo real y Firestore



Características

Realtime Database ofrece sincronización instantánea y estructura basada en JSON.

Firestore utiliza documentos y colecciones, facilitando la organización y consultas complejas.



Capacidades de consulta

Firestore permite consultas avanzadas, filtrado y ordenación.

Realtime Database tiene capacidades de consulta más limitadas.



Escalabilidad

Firestore está diseñado para escalar automáticamente y soportar grandes volúmenes de datos.

Realtime Database es adecuado para proyectos pequeños y medianos.



Soporte offline

Firestore ofrece sincronización offline robusta y persistencia local.

Realtime Database también soporta offline, pero con menos funcionalidades avanzadas.

Servicios de Hosting, almacenamiento y otras herramientas integradas



Firebase Hosting

Despliegue rápido y seguro de sitios web con SSL automático y CDN global.



Firebase Storage

Almacenamiento de archivos multimedia, seguro y escalable para apps modernas.



Cloud Functions

Ejecución de código backend sin servidores, automatización y lógica personalizada.



Monitoreo

Seguimiento en tiempo real del rendimiento y errores de la aplicación.

Casos de uso en aplicaciones móviles

Chats en tiempo real

Mensajería instantánea con presencia, entrega confiable y latencia baja.

Sincronización offline

Datos que persisten y se reconcilian al recuperar conexión.

Autenticación segura

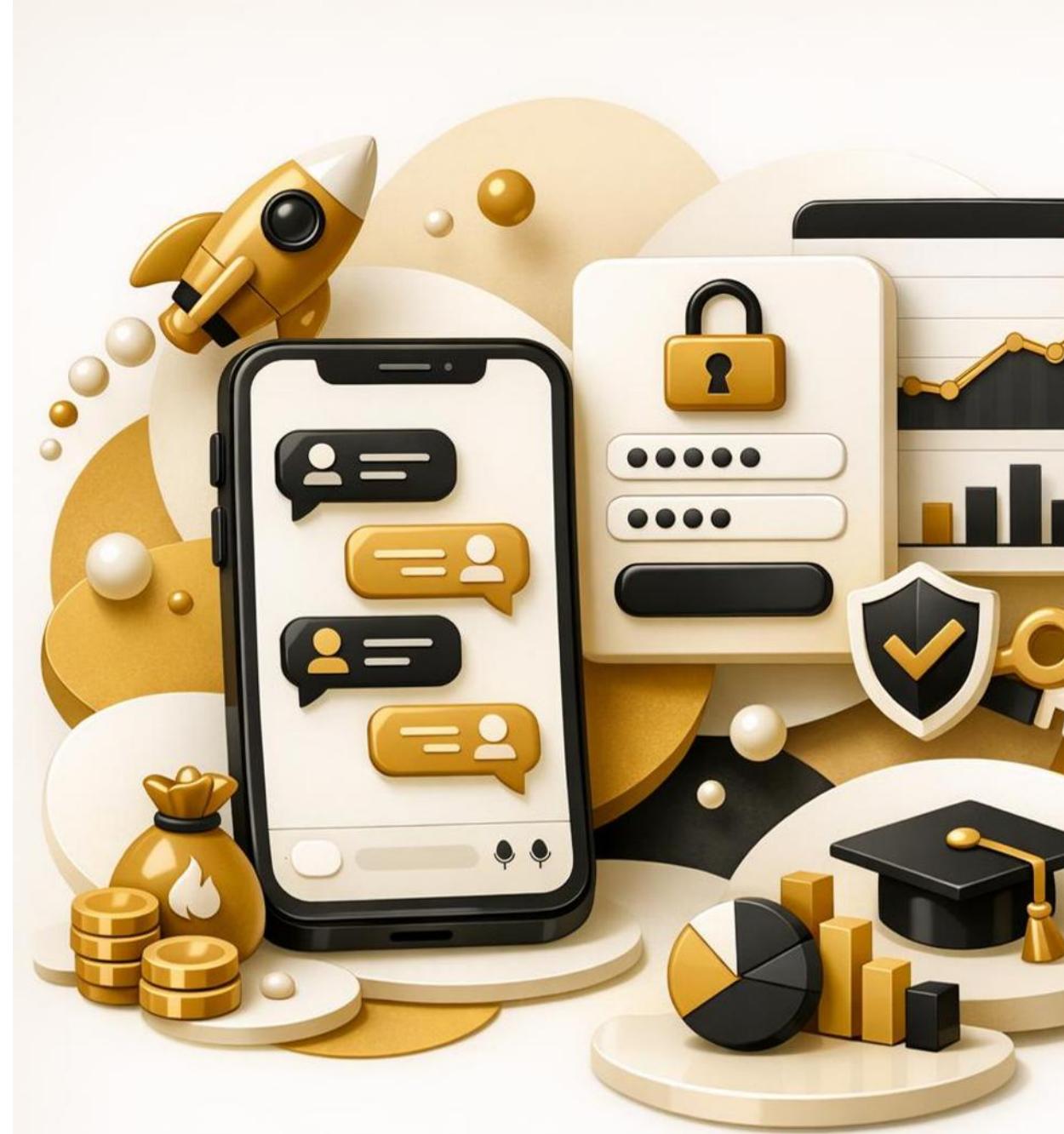
Login con correo, redes sociales o SSO con reglas robustas.

Analítica accionable

Eventos, embudos y audiencias para mejoras continuas.

Sectores e impacto

De fintech a educación, backend serverless acelera releases.



Integración en aplicaciones web

SPA con backend en tiempo real

Firestore sincroniza datos al instante, ideal para contenido dinámico y colaboración.

Hosting y CI/CD integrados

Despliegues rápidos y pipelines automatizados simplifican el ciclo de entrega.

Autenticación moderna y UX

Firebase Authentication gestiona usuarios con seguridad y mejora la experiencia.





- Cloud Firestore es una base de datos flexible y escalable para el desarrollo en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud.

Conceptos clave

Colecciones

Son contenedores de información.

Agrupan documentos del mismo tipo.

Equivalente a una “tabla” en bases relationales.

- **Ejemplos de colecciones:**
 - usuarios
 - productos
 - alumnos
 - tareas

Documentos

Conceptos clave

Cada documento representa un registro.

Tiene un **ID único**.

Vive dentro de una colección.

Conceptos clave

Campos

Son los datos almacenados dentro de un documento.

Se componen de:

- nombre del campo
- Valor
- **Tipos de datos comunes:**
- Texto (string)
- Número
- Booleano
- Arreglos
- Objetos

Conceptos clave

Subcolecciones

Una colección dentro de un documento.

Permiten organizar información más compleja.

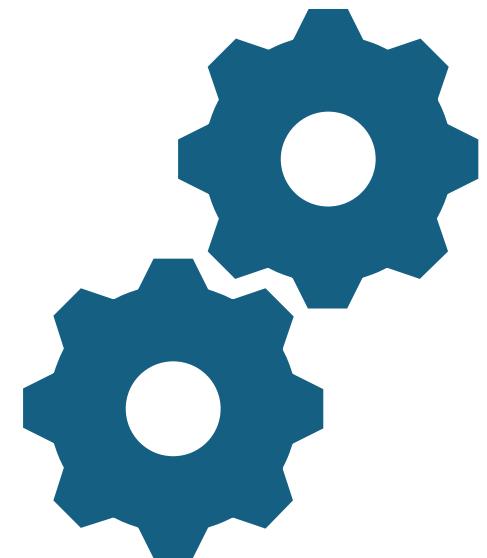
Ejemplo conceptual en Firestore

```
usuarios (colección)
  └── usuario1 (documento)
      ├── nombre: "Juan Pérez"
      ├── edad: 22
      └── correo: "juan@gmail.com"
```

Ejemplo con subcolección

```
usuarios
└ usuario1
  └ nombre: "Juan"
  └ materias (subcolección)
    └ materia1
      └ nombre: "Bases de Datos"
      └ calificacion: 90
```

Configuración del Proyecto Firebase



Crear un proyecto en Firebase Console

Primero necesita crear una cuenta en Firebase

1. Firebase es un servicio de Google, por lo que se requiere una cuenta de Google.
2. Abrir el navegador.
3. Ir a:
<https://firebase.google.com>
4. Dar clic en “Ir a la consola”.
5. Iniciar sesión con una cuenta de Google.

Nota :

Si ya tienes una cuenta de Gmail, no necesitas crear una nueva.



Hola celi

Te damos la bienvenida a Firebase

Comenzar



Para comenzar, configura un
proyecto de Firebase

Integra productos de Firebase para potenciar tu
app

Prueba una app de ejemplo



Prueba una app de planificación de viajes
potenciada por IA

Implementa una app de ejemplo que use Firestore, Authentication
y una entrada multimodal con Firebase AI Logic. Explora el código
en Firebase Studio



Crea una app de Flutter potenciada por IA

Implementa una app de ejemplo que muestre cómo funcionan la
API de Gemini Live, las instrucciones multimodales y la creación
de imágenes con Nano Banana en Flutter



Explorar nuestro proyecto de demostración de solo lectura

Ver



Paso 2: Crear un nuevo proyecto en Firebase

1. En la Firebase Console, hacer clic en “Agregar proyecto”.
2. Escribir el nombre del proyecto.
Ejemplo:
curso-nosql-firebase
3. Hacer clic en Continuar.
4. Firebase preguntará si deseas habilitar Google Analytics:
Para este curso: No es necesario
Selecciona Desactivar.
5. Clic en Crear proyecto.
6. Esperar unos segundos y hacer clic en Continuar.
7. Listo, el proyecto ha sido creado.

 Crear un proyecto

Comencemos con el nombre de tu proyecto[?]

Nombre del proyecto

Curso de prueba

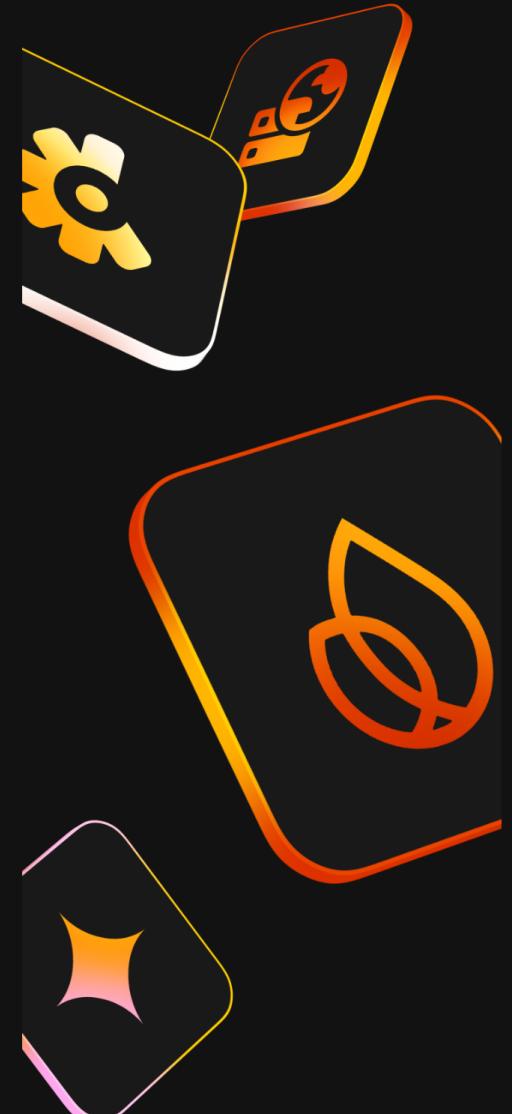
 curso-de-prueba-ef614

Acepto las [condiciones de Firebase](#)

Únete al [Programa para desarrolladores de Google](#) para enriquecer tu experiencia como desarrollador con acceso a asistencia de IA, recursos de aprendizaje, insignias de perfil y mucho más.

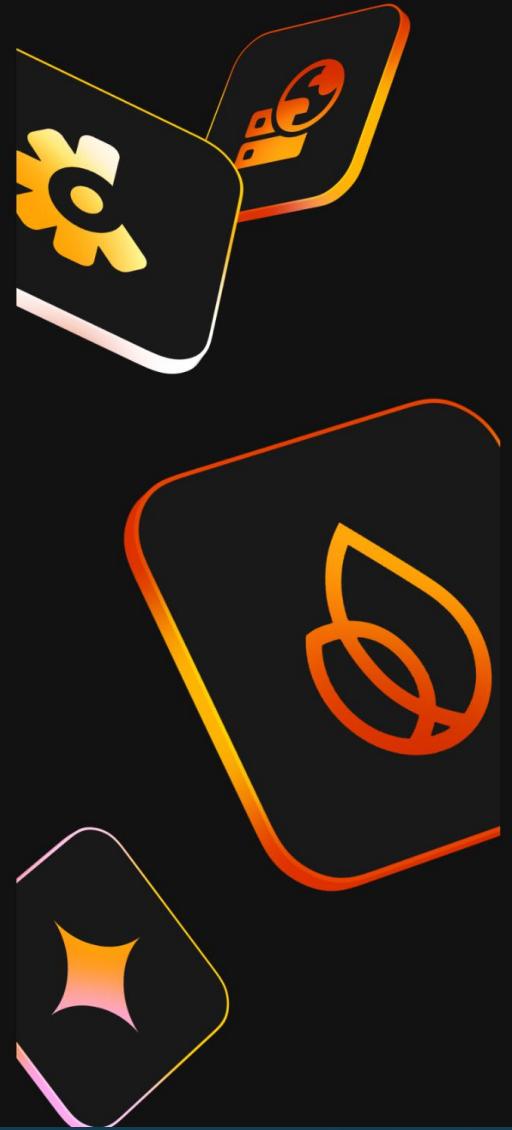
¿Ya tienes un proyecto de Google Cloud?
[Agregar Firebase al proyecto de Google Cloud](#)

Continuar





Estamos preparando tu proyecto. Espera
un momento.



Paso 3: Activar Firestore (Base de datos)

1. En el menú izquierdo, seleccionar Build → Firestore Database.
2. Clic en Crear base de datos.
3. Seleccionar el modo de inicio:
 - Elegir Modo de prueba (solo para aprendizaje).
4. Clic en Siguiente.
5. Elegir la ubicación del servidor (por ejemplo: us-central).
6. Clic en Habilitar.
7. Resultado:
Firestore queda activo y listo para usarse.

← → ⌂ console.firebaseio.google.com/u/0/project/curso-de-prueba-62e1a.firebaseio?hl=es-419

Personal Posgrado UNSI Buscadores Conacyt bd requisitos Contaby estado de larte IA Sci-hub Importado IMPORTANCIA Ingles Libros Todos los marcadores

Firebase curso de prueba ▾

Descripción gen... |

Accesos directos a proyectos

Firestore Database

Categorías de producto

Compilación Ejecución Analytics IA

Cloud Firestore

Actualizaciones en tiempo real, consultas potentes, escalado automático y compatibilidad con MongoDB

[Crear base de datos](#) [Preguntarle a Gemini](#)

Más información

Herramientas de desarrollo relacionadas

[Firebase Studio](#)

Spark Sin costo (USD 0) Actualizar

¿Cómo empiezo? Ver los documentos

¿Cuánto costará Cloud Firestore? Ver los precios

Introducing Cloud Firestore

Ver más t... Compartir

Crea una base de datos

1 Seleccionar edición

Edición Standard

Motor de consultas simples con indexación automática

Admite operaciones principales

Edición Enterprise

Motor de consultas avanzadas con indexación autoadministrada.

Admite operaciones principales, de canalizaciones y de MongoDB

¿No sabes cuál es la edición ideal para ti? [Comparar ediciones](#)

Siguiente

2 ID y ubicación de la base de datos

3 Configurar

Firestore – Edición Standard (puntos principales)

Base de datos de documentos con SDKs (Software Development Kit) para muchos lenguajes.

Actualización en tiempo real y funcionamiento sin conexión.

Alta disponibilidad en configuraciones regionales y multirregionales.

Escalado automático y modelo sin servidores.

Soporta Firestore en modo nativo con operaciones Core:

- Lecturas

- Escrituras

- Consultas básicas

Firestore – Edición Enterprise (puntos principales)

Motor de consultas más avanzado con capacidades exhaustivas.

Índices opcionales y totalmente personalizables (no se crean automáticamente).

Mayor control para desarrolladores y capacidades ampliadas.

Ofrece dos modos de operación:

x Crea una base de datos

1 Seleccionar edición

2 ID y ubicación de la base de datos

ID de la base de datos

(default)

Ubicación

northamerica-south1 (Mexico) ▾

ⓘ La configuración de ubicación es el lugar donde se almacenarán tus datos de Cloud Firestore

! No podrás cambiar la ubicación después de configurarla

Más información ↗

Siguiente

3 Configurar

Crea una base de datos

 ID y ubicación de la base de datos

 3 Configurar

Después de definir la estructura de tus datos, **debes crear reglas para protegerlos.** [Más información](#) 

Iniciar en **modo de producción**

Tus datos son privados de forma predeterminada. El acceso de lectura/escritura de los clientes solo se otorgará como se indica en tus reglas de seguridad.

Comenzar en **modo de prueba**

Para permitir una configuración rápida, los datos se abren de forma predeterminada. Sin embargo, debes actualizar las reglas de seguridad en un plazo de 30 días para habilitar el acceso de lectura/escritura a largo plazo para los clientes.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
        request.time < timestamp.date(2026, 3, 5);
    }
  }
}
```

 **Las reglas de seguridad predeterminadas del modo de prueba permiten que cualquier persona con acceso a la referencia de tu base de datos pueda ver, editar y borrar todos los datos de esta durante los siguientes 30 días**

Cancelar

Crear

Operaciones CRUD en Firebase

¿Qué es CRUD?

- CRUD representa las **operaciones básicas de cualquier sistema**:

Operación	Significado	Ejemplo
C	Create	Registrar usuario
R	Read	Mostrar usuarios
U	Update	Editar usuario
D	Delete	Eliminar usuario

Paso a paso: crear datos **SOLO** desde la consola

-
- **Paso 1: Entrar a Firestore**
 - Firebase Console
 - Proyecto
 - Build → Firestore Database
 - Pestaña Datos

Paso 2: Crear una colección

-
- Clic en **Iniciar colección**
 - Nombre de la colección:
 - usuarios
 - Clic en Siguiente

Paso 3: Crear un documento

- ID del documento:
 - Automático → dejar vacío
 - Manual → usuario1
- Agregar campos:
 - nombre → string → "Juan"
 - edad → number → 22
 - correo → string → "juan@gmail.com"
- Clic en **Guardar**

curso de prueba

Cloud Firestore >

Inicia una colección

Asignar un ID a la colección — 2 Agregar el primer documento

Ruta superior del documento

/usuarios

ID de documento ?

Z3EOiB9v2kPt4IIHA91Q

Campo	Tipo
nombre	= string

String

juan

+ Agregar campo

Cancelar Guardar

This screenshot shows the Google Cloud Platform (GCP) Firestore interface. The user is creating a new collection named 'usuarios'. A single document is being added with the ID 'Z3EOiB9v2kPt4IIHA91Q'. The document contains a single field 'nombre' of type string with the value 'juan'. The interface includes a sidebar with navigation links like 'Home', 'Collections', 'Documents', and 'Functions'. The main area has tabs for 'Datos' (Data) and 'Protección' (Protection). The bottom right corner features a large blue 'Guardar' (Save) button.

The screenshot shows the Google Cloud Firestore interface. At the top left, there's a navigation bar with a home icon, three dots, and a document ID: Z3E0iB9v2kPt4l... On the top right, there's a "Más funciones en Google Cloud" dropdown menu.

The main area is a grid of cards. The first card, titled "(default)", contains a "+ Iniciar colección" button and a "usuarios" section with a "usuarios" button and a right arrow. The second card, titled "usuarios", contains a "+ Agregar documento" button and a document ID: Z3E0iB9v2kPt4l... followed by a right arrow. The third card, titled "Z3E0iB9v2kPt4l... (document)", has a list of fields:

- correo: "juan@gmail.com"
- edad: "22"
- nombre: "juan"

¿Qué Sí puedes hacer solo desde la consola?

Desde Firestore Database puedes:

Crear la base de datos

- Elegir modo prueba
- Definir ubicación

Crear colecciones

- Definir nombre de la colección

Crear documentos

- ID automático
- ID manual

Agregar campos

- Texto
- Número
- Booleano
- Arreglos
- Objetos

Editar documentos

- Cambiar valores
- Agregar o eliminar campos

Eliminar documentos o colecciones

¿Qué NO se puede hacer solo con la consola?

No puedes:

- Automatizar procesos
- Conectar una app real
- Ejecutar lógica compleja
- Hacer consultas dinámicas avanzadas
- Para eso sí necesitas código.

¿Cómo se visualizan los datos en Firestore?

- Firestore **no usa tablas** como las bases de datos relacionales.
- Los datos se muestran en forma **jerárquica**, similar a carpetas.
- La vista principal es un **árbol de datos**.

```
usuarios  (colección)
├── aK92Ld3  (documento)
│   ├── nombre: "Juan"
│   ├── correo: "juan@gmail.com"
│   ├── rol: "alumno"
│   └── edad: 20
|
└── Bx81Pa9
    ├── nombre: "Ana"
    ├── correo: "ana@gmail.com"
    └── rol: "admin"
```

`usuarios`

`└ userId123`

`├ nombre: "Luis"`

`├ correo: "luis@mail.com"`

`└ cursos`

`└ curso01`

`├ nombre: "NoSQL"`

`├ progreso: 80`

Visualización desde el código (consultas)

- Cuando consultas Firestore **no ves tablas**, sino objetos/JSON.

```
const snapshot = await getDocs(collection(db, "usuarios"));

snapshot.forEach(doc => {
    console.log(doc.id, doc.data());
}):
```

¿Qué es JSON?

- JSON significa **JavaScript Object Notation**.
 - Es un formato de texto ligero.
 - Se usa para almacenar y transferir datos.
 - Muy común en aplicaciones web y móviles.
-
- Utiliza llaves {} para objetos y corchetes [] para arreglos.
 - Independiente del lenguaje, funciona en Python, PHP, Java, entre otros.

Estructura básica de un JSON

Estructura básica de un JSON

- Un JSON está formado por:
- Claves
- Valores

Ejemplo:

— json

```
{  
  "usuario": "Juan",  
  "edad": 30,  
  "activo": true,  
  "hobbies": ["leer", "programar"]  
}
```

Crear un proyecto web en Firebase

- **Paso 1:**

En Firebase Console → **Configuración del proyecto**

Clic en </> **Agregar app (Web)**

Nombre de la app (ejemplo):

crud-firebase

Registrar app

Firebase mostrará un **objeto de configuración**

Firebase

prueba ▾ Configuración de proyecto

Configuración de proyecto

General Cloud Messaging Integraciones Cuentas de servicio Privacidad de los datos Usuarios y permisos Alertas

Tu proyecto

Nombre del proyecto	prueba
ID del proyecto ⓘ	prueba-709f0
Número del proyecto ⓘ	560562155578

Entorno

Este parámetro de configuración permite personalizar tu proyecto para las diferentes etapas del ciclo de vida de la app

Tipo de entorno	Sin especificar
-----------------	-----------------

Tus apps

No hay apps en tu proyecto

Selecciona una plataforma para comenzar

Herramientas de desarrollo relacionadas

[Firebase Studio](#)

Spark
Sin costo (USD 0 al mes) [Actualizar](#)

- **Paso 2: Inicializar Firebase en JavaScript**
- Archivo: firebase.js

```
// Importar funciones necesarias
import { initializeApp } from "https://www.gstatic.com/firebasejs/12.8.0.firebaseio.js";
import { getFirestore } from "https://www.gstatic.com/firebasejs/12.8.0/firebase-firestore.js";

// Configuración del proyecto
const firebaseConfig = {
    apiKey: "AIzaSyDbWS_B0U4iBumTNxUcvXjE5VMN8npi3tE",
    authDomain: "prueba-709f0.firebaseio.com",
    projectId: "prueba-709f0",
    storageBucket: "prueba-709f0.firebaseiostorage.app",
    messagingSenderId: "560562155578",
    appId: "1:560562155578:web:91ae3ccea2684e2f5cc503"
};

// Inicializar Firebase
const app = initializeApp(firebaseConfig);

// Inicializar Firestore
export const db = getFirestore(app);
```

- **Paso 2: Crear un documento con ID automático**

```
window.crearUsuarioDesdeHTML = async function () {
  await addDoc(collection(db, "usuarios"), {
    nombre: "Juan Pérez",
    edad: 22,
    correo: "juan@gmail.com"
  });

  alert("Usuario registrado en Firestore");
};
```

- **Paso 2: Crear un documento con ID automático**

```
window.crearUsuarioDesdeHTML = async function () {
  await addDoc(collection(db, "usuarios"), {
    nombre: "Juan Pérez",
    edad: 22,
    correo: "juan@gmail.com"
  });

  alert("Usuario registrado en Firestore");
};
```

- **Paso 3: Llamar la función**

READ – Leer datos

Leer todos los documentos de una colección

Importar funciones

- import { collection, getDocs } from
"https://www.gstatic.com/firebasejs/10.7.1.firebaseio.js";

- **Paso 2: Leer usuarios**

```
window.obtenerUsuariosDesdeHTML = async function () {
  const resultado = document.getElementById("resultado");
  resultado.textContent = "";

  const querySnapshot = await getDocs(collection(db, "usuarios"));

  querySnapshot.forEach((doc) => {
    const data = doc.data();
    resultado.textContent +=
      `ID: ${doc.id}\n` +
      `Nombre: ${data.nombre}\n` +
      `Edad: ${data.edad}\n` +
      `Correo: ${data.correo}\n\n`;
  });
};
```

- **UPDATE – Actualizar datos**

Actualizar un usuario existente

Necesitamos el ID del documento.

Importar funciones

- `import { doc, updateDoc } from
 "https://www.gstatic.com/firebasejs/10.7.1.firebaseio-firestore.js";`

- **Actualizar campos**

```
window.actualizarDesdeHTML = async function () {
  const idUsuario = document.getElementById("idActualizar").value;
  const edadNueva = document.getElementById("edadNueva").value;

  if (!idUsuario || !edadNueva) {
    alert("Faltan datos");
    return;
  }

  await updateDoc(doc(db, "usuarios", idUsuario), {
    edad: Number(edadNueva)
  });

  alert("Usuario actualizado correctamente");
};
```

- **Paso 3: Llamar la función**
- actualizarUsuario("ID_DEL_DOCUMENTO")

- **DELETE – Eliminar datos**

Eliminar un documento

- **Importar función**

- import { deleteDoc } from
"https://www.gstatic.com/firebasejs/10.7.1.firebaseio-firebase.js";

- **Paso 2: Eliminar usuario**

```
window.eliminarDesdeHTML = async function () {
  const idUsuario = document.getElementById("idEliminar").value;

  if (!idUsuario) {
    alert("Ingresa el ID del usuario");
    return;
  }

  await deleteDoc(doc(db, "usuarios", idUsuario));

  alert("Usuario eliminado correctamente");
};
```

Cambiar reglas

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

- <https://github.com/celidiaz/NoSQL.git>