



Hacienda



Transferencia de Conocimiento

Asesoftware



Agenda

-  1 Presentación
-  2 Cronograma
-  3 Alcance
-  4 Inicio de sesión

PRESENTACIÓN

Juan David Leon Barrera

Ingeniero de Sistemas

jleon@asesoftware.com
304 340 5607

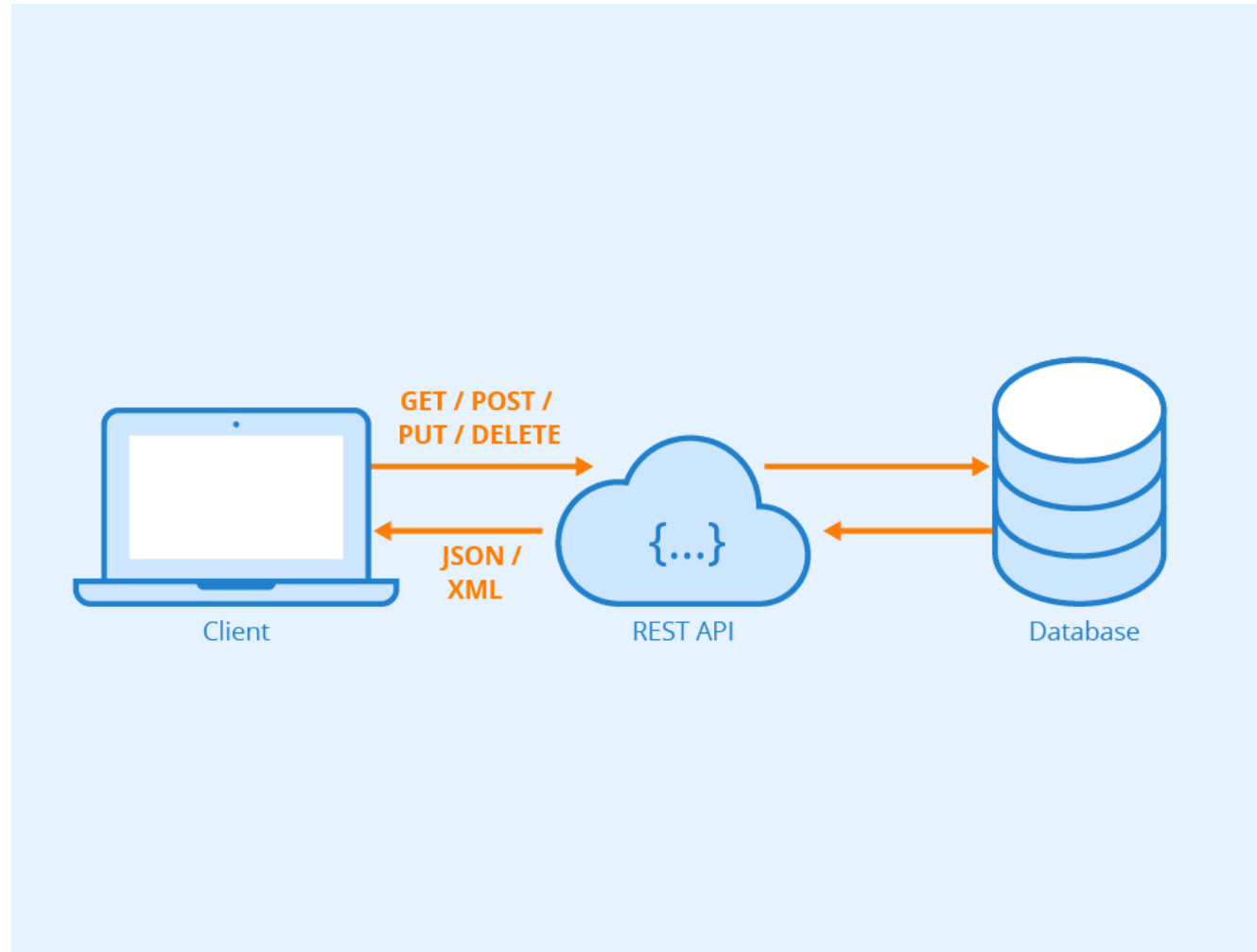


CRONOGRAMA

Sesión	Modulo	Tema	Teoría	Práctica	Tiempo
1	1	Introduccion .NET y C#	1 hora (Modulo 1)	1 Hora (Modulo 1)	2
2	1	Introduccion .NET y C#	1 hora (Modulo 1)	1 Hora (Modulo 1)	2
3	1	Introduccion .NET y C#	1 hora (Modulo 1)	1 Hora (Modulo 1)	2
4	2	Fundamentos C#	1 hora (Modulo 2)	1 hora (Modulo 2)	2
5	2	Fundamentos C#	1 hora (Modulo 2)	1 hora (Modulo 2)	2
6	2	Fundamentos C#	1 hora (Modulo 2)	1 hora (Modulo 2)	2
7	2	Fundamentos C#	1 hora (Modulo 2)	1 hora (Modulo 2)	2
8	3	Programación Orientada a Objetos	1 hora (Modulo 3)	1 hora (Modulo 3)	2
9	3	Programación Orientada a Objetos	1 hora (Modulo 3)	1 hora (Modulo 3)	2
10	3 y 4	Programación Orientada a Objetos - Bases de Datos	1 hora (Modulo 4)	1 hora (Modulo 3)	2
11	4	Bases de Datos	1 hora (Modulo 4)	1 hora (Modulo 4)	2
12	4	Bases de Datos		2 horas (Modulo 4)	2
13	5	Api REST	1 hora (Modulo 5)	1 hora (Modulo 5)	2
14	5	Api REST	1 hora (Modulo 5)	1 hora (Modulo 5)	2
15	5	Api REST		2 horas (Modulo 5)	2



ALCANCE



INTRODUCCIÓN A .NET Y C#



Módulo 1: Introducción a .NET y C# (6 horas)

☐ **Teoría (3 horas)**

- Conceptos básicos de programación
- ¿Que es .NET, C# y Entity Framework Core?
- Introducción a C#: Sintaxis básica

☐ **Práctica (3 horas)**

- Instalación de Visual Studio
- Creación de un proyecto simple (Consola)

INTRODUCCION

- Lenguaje líder en plataforma Windows.
- Parte principal de sus herramientas de desarrollo: Visual Studio y .NET
- Engloba las características mas provechosas de otros lenguajes.
- Permite crear aplicaciones de escritorio, web, móviles, videojuegos...
- Sintaxis fácil de entender



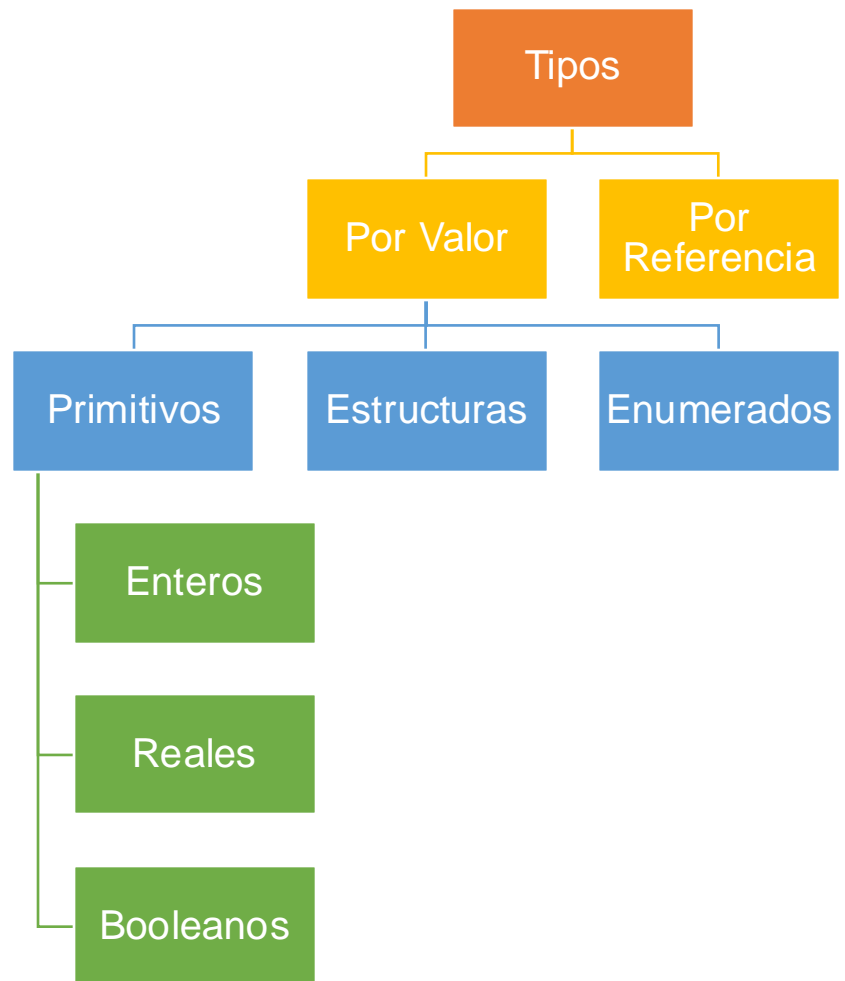
INDICE TIOBE



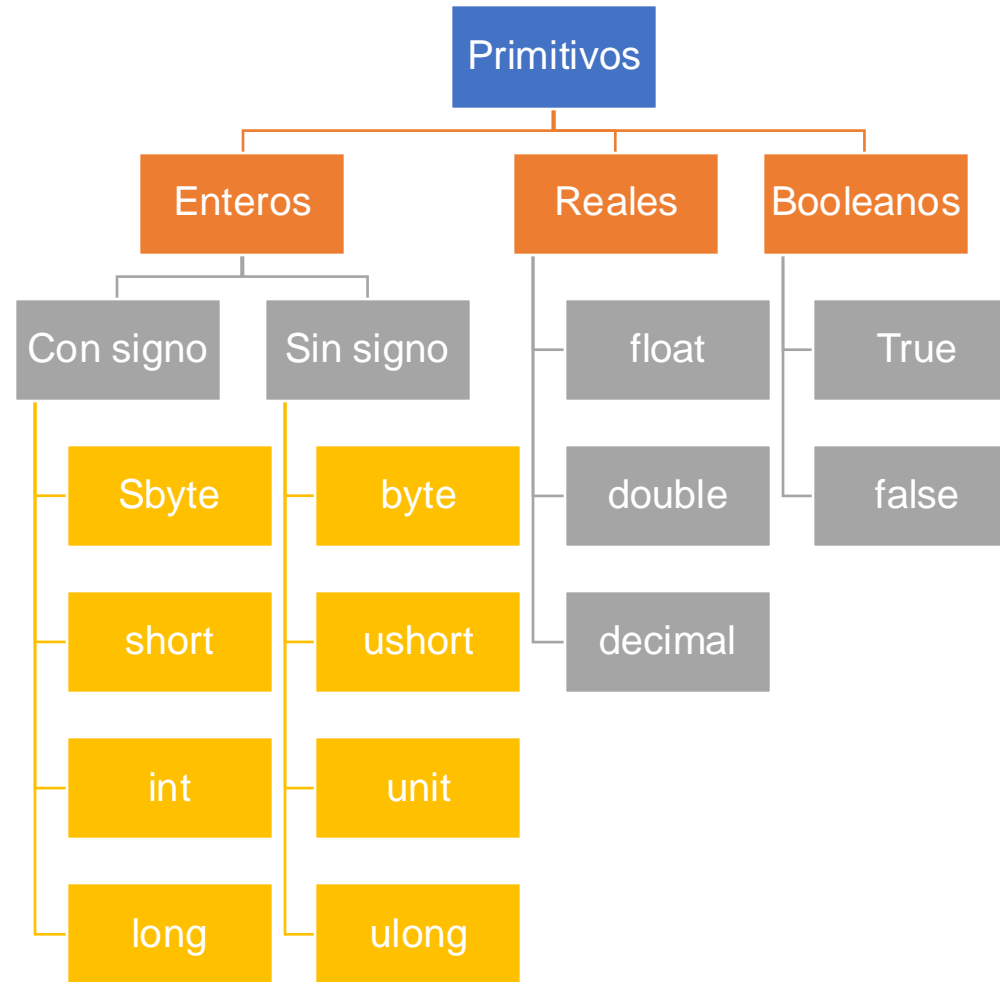
<https://www.tiobe.com/tiobe-index/>



SINTAXIS – TIPOS DE DATOS

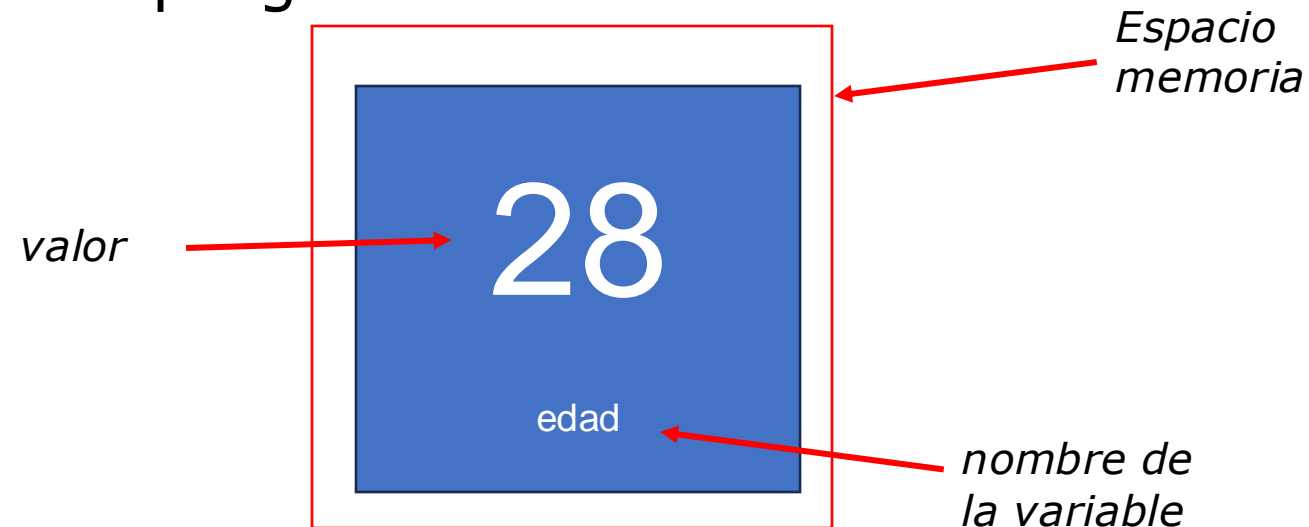


SINTAXIS – TIPOS DE DATOS



Variables ¿Que son?

Espacio en la memoria (RAM) del ordenador donde se almacenará un valor que podra cambiar durante la ejecución del programa.



SQL



TEMARIO

- Definición de bases de datos.
- Diferencia entre bases de datos relacionales y no relacionales.
- Elementos básicos de una base de datos relacional:
- Tablas
- Filas (Registros)
- Columnas (Campos)
- Claves primarias y foráneas
- Relación entre tablas: **uno a uno, uno a muchos, muchos a muchos.**
- Normalización de bases de datos: qué es y por qué es importante.

PAGINAS DE APOYO



<https://postgres.new>



<https://sqlbolt.com/>

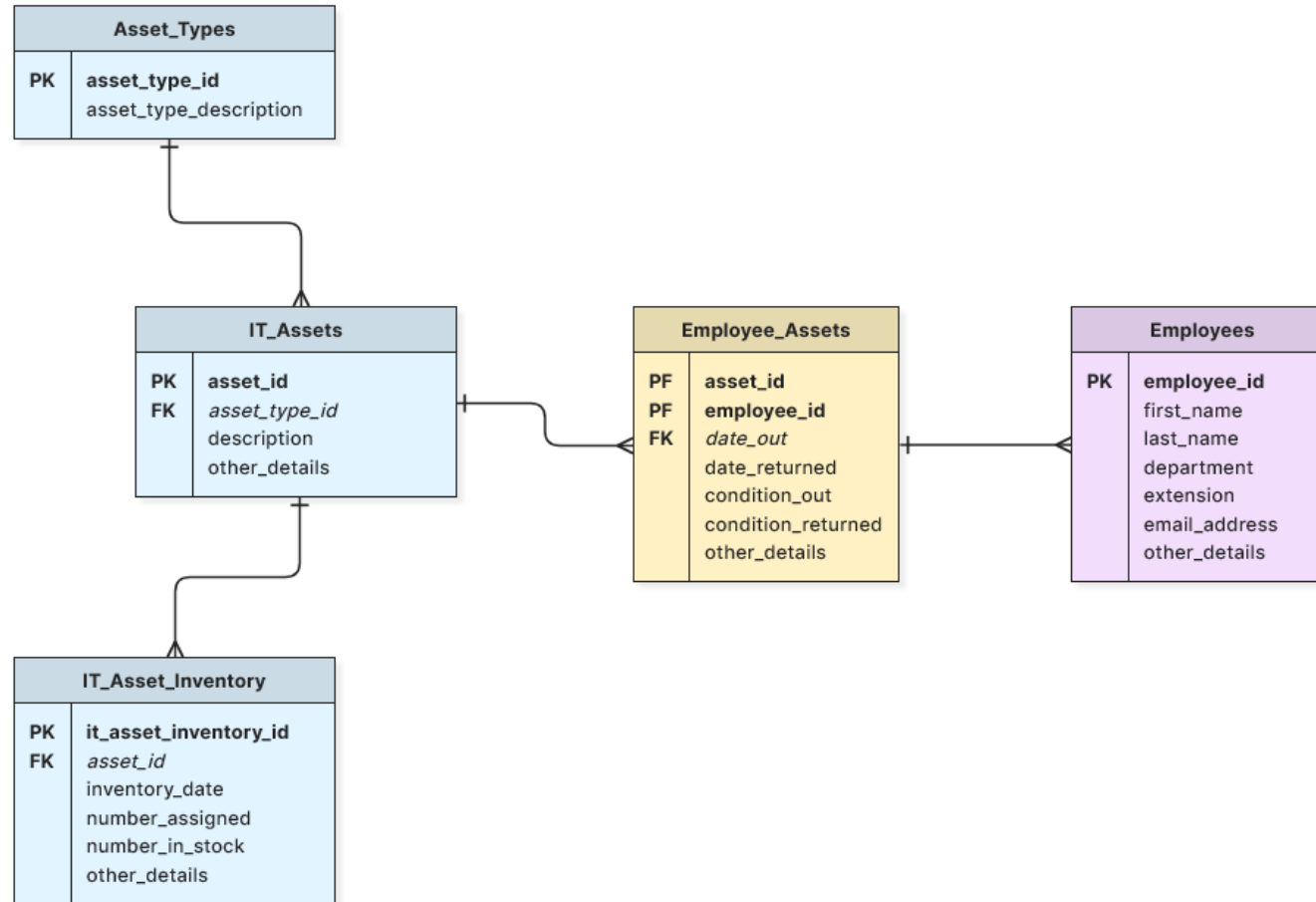
¿Qué es una Base de Datos?

- **Definición:**

Una base de datos es un sistema organizado para recopilar, almacenar y gestionar datos. Las bases de datos permiten que los datos sean fácilmente accesibles, gestionados y actualizados.

- **Características principales:**

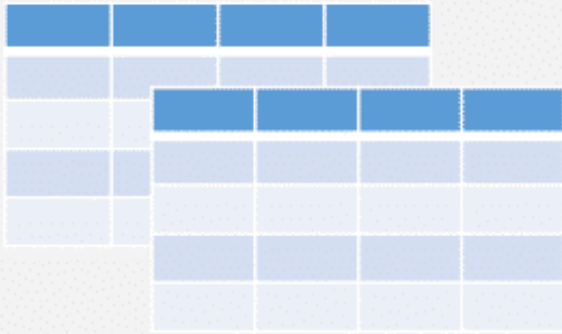
- Almacenan grandes cantidades de datos.
- Permiten realizar búsquedas rápidas y consultas.
- Facilitan la actualización y la integridad de los datos.
- Pueden estar estructuradas (relacionales) o no estructuradas (no relacionales).



- **Bases de Datos Relacionales (RDBMS):**
 - Utilizan tablas para almacenar datos.
 - Los datos están estructurados en filas y columnas.
 - Se usan para manejar datos estructurados y relacionados (Ej.: MySQL, PostgreSQL).
- **Bases de Datos No Relacionales (NoSQL):**
 - No usan tablas; pueden almacenar datos en documentos, claves-valor o grafos.
 - Son más flexibles en cuanto a la estructura de los datos.
 - Son ideales para datos no estructurados o semi-estructurados (Ej.: MongoDB, Cassandra).

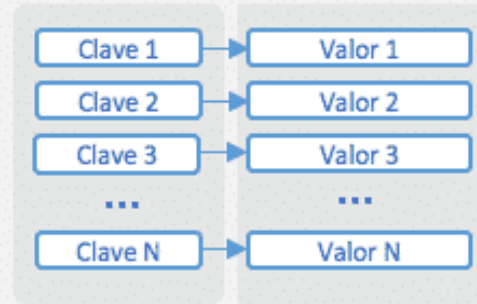
Base de datos SQL

Relacional



Base de datos NoSQL

Clave-Valor



Documental



Grafos



Elementos Básicos de una Base de Datos Relacional

- Tablas:**

- Son la estructura principal donde se almacenan los datos.
- Compuestas por filas y columnas.

- Filas (Registros):**

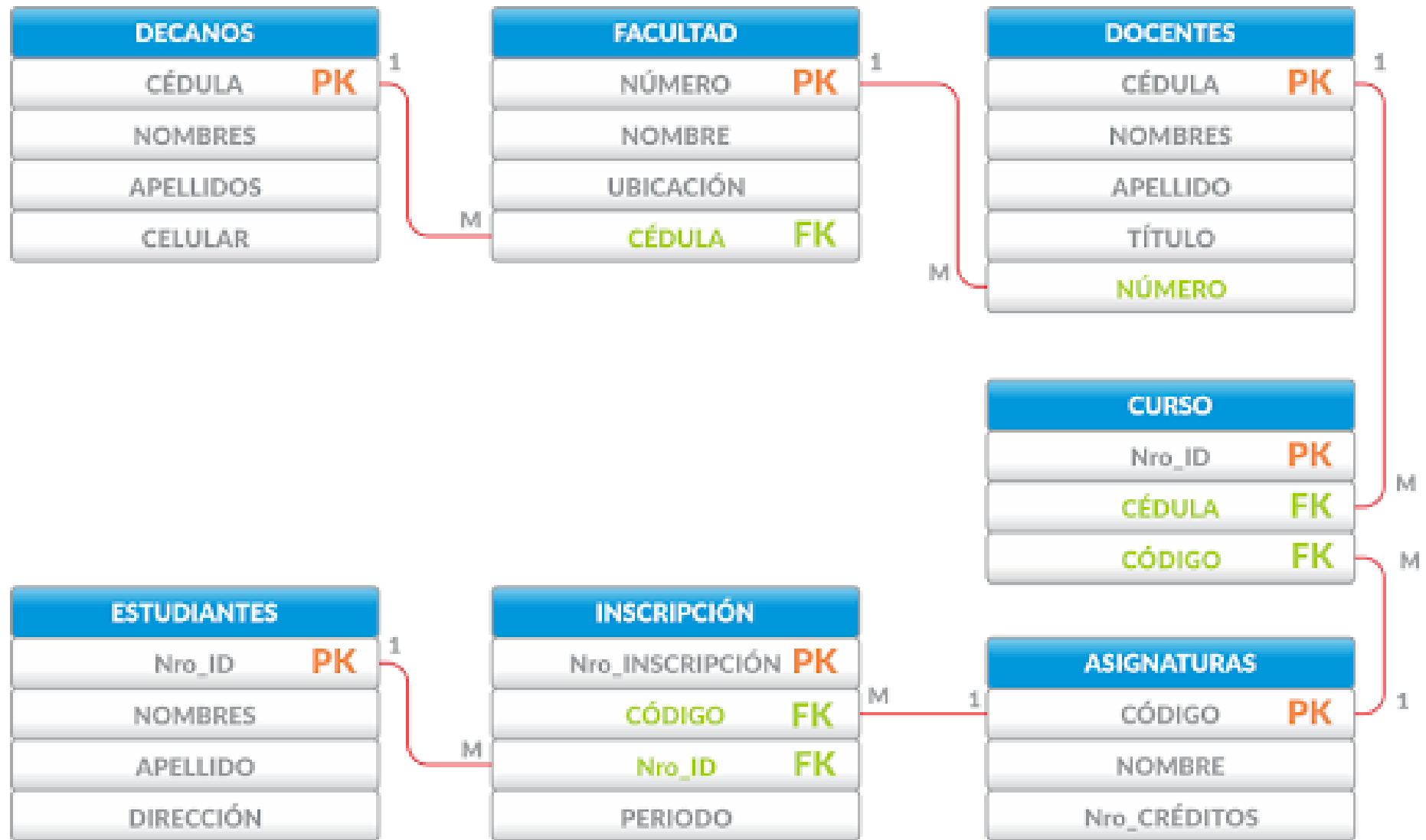
- Representan una entrada única de datos (un registro).
- Cada fila tiene valores correspondientes a las columnas.

- Columnas (Campos):**

- Son los atributos o características de los datos.
- Cada columna tiene un tipo de dato específico (ej., texto, número, fecha).

- Claves Primarias y Foráneas:**

- **Clave Primaria:** Identificador único de cada fila.
- **Clave Foránea:** Columna que establece una relación con otra tabla.



- **Relación Uno a Uno (1:1):**

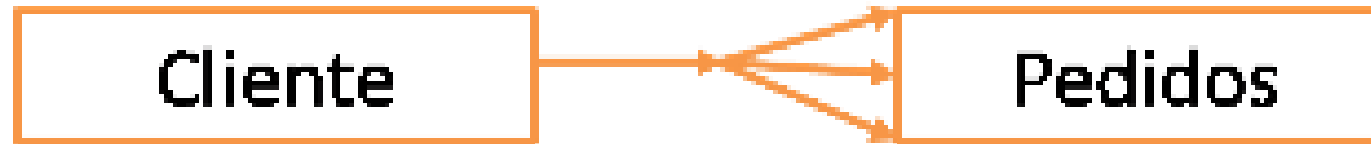
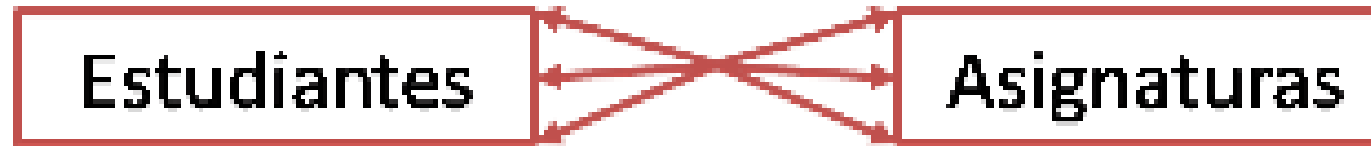
- Una fila en una tabla está asociada a una sola fila en otra tabla.
- Ejemplo: Cada persona tiene una única dirección.

- **Relación Uno a Muchos (1-M):**

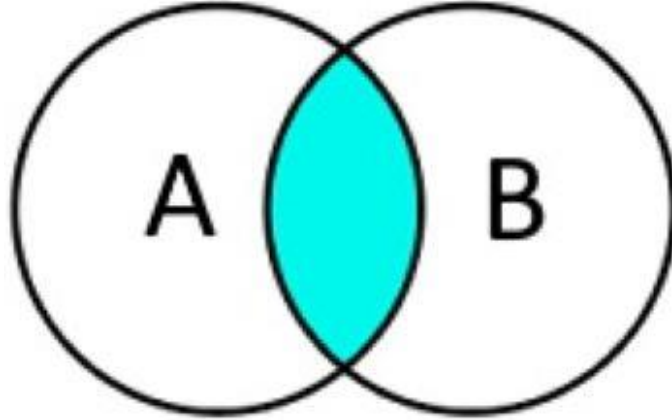
- Una fila en una tabla puede estar asociada a varias filas en otra tabla.
- Ejemplo: Un cliente puede tener múltiples pedidos.

- **Relación Muchos a Muchos (M-M):**

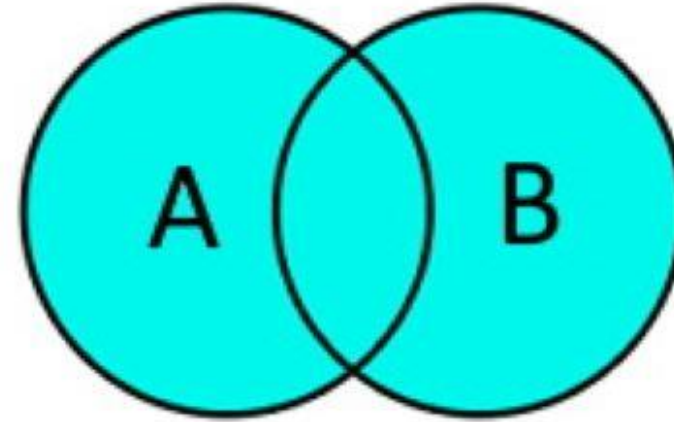
- Varias filas de una tabla pueden estar asociadas a varias filas de otra tabla.
- Ejemplo: Estudiantes y cursos, donde un estudiante puede tomar varios cursos y un curso puede tener varios estudiantes.



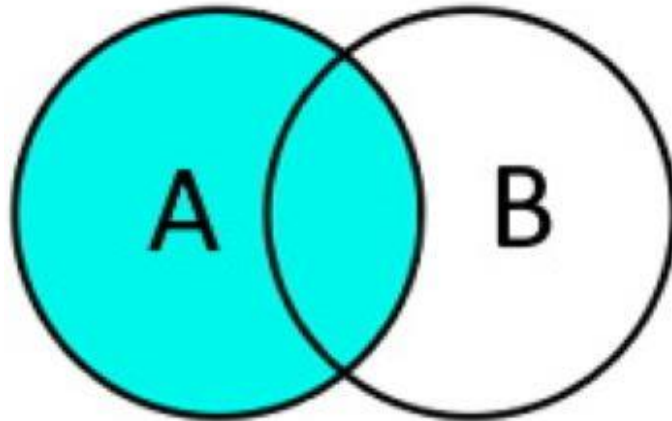
INNER JOIN



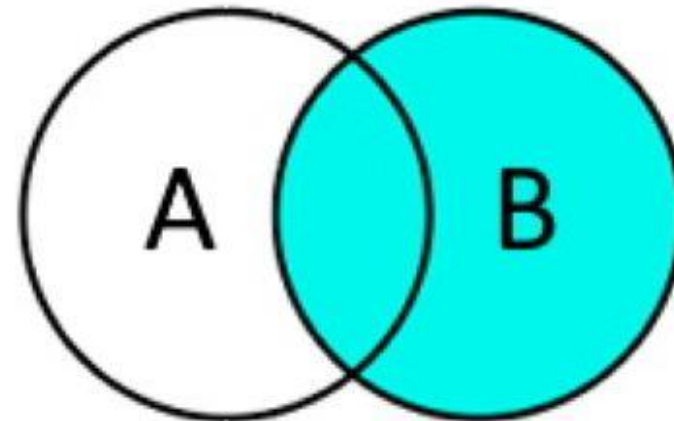
FULL JOIN



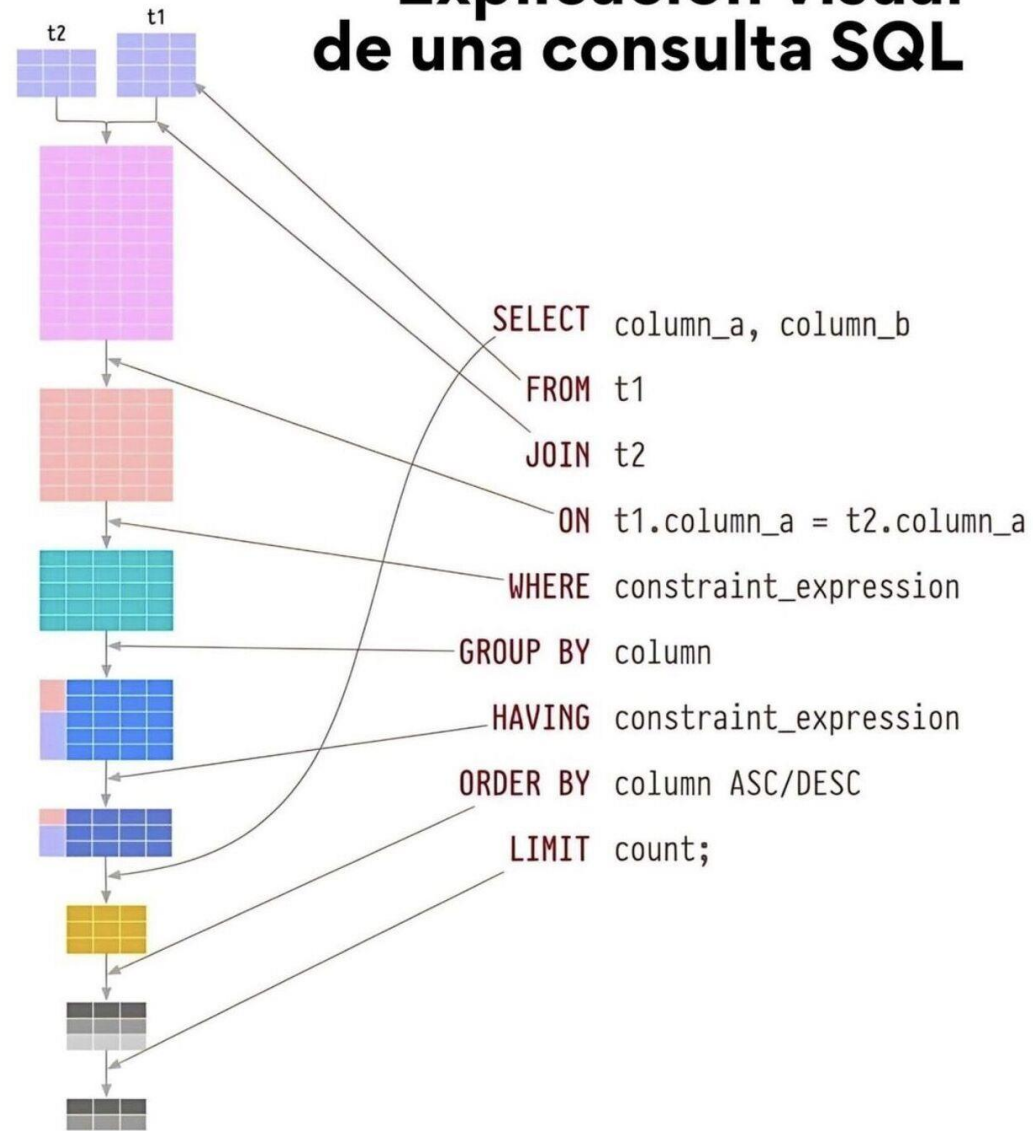
LEFT JOIN



RIGHT JOIN



Explicación visual de una consulta SQL



API REST



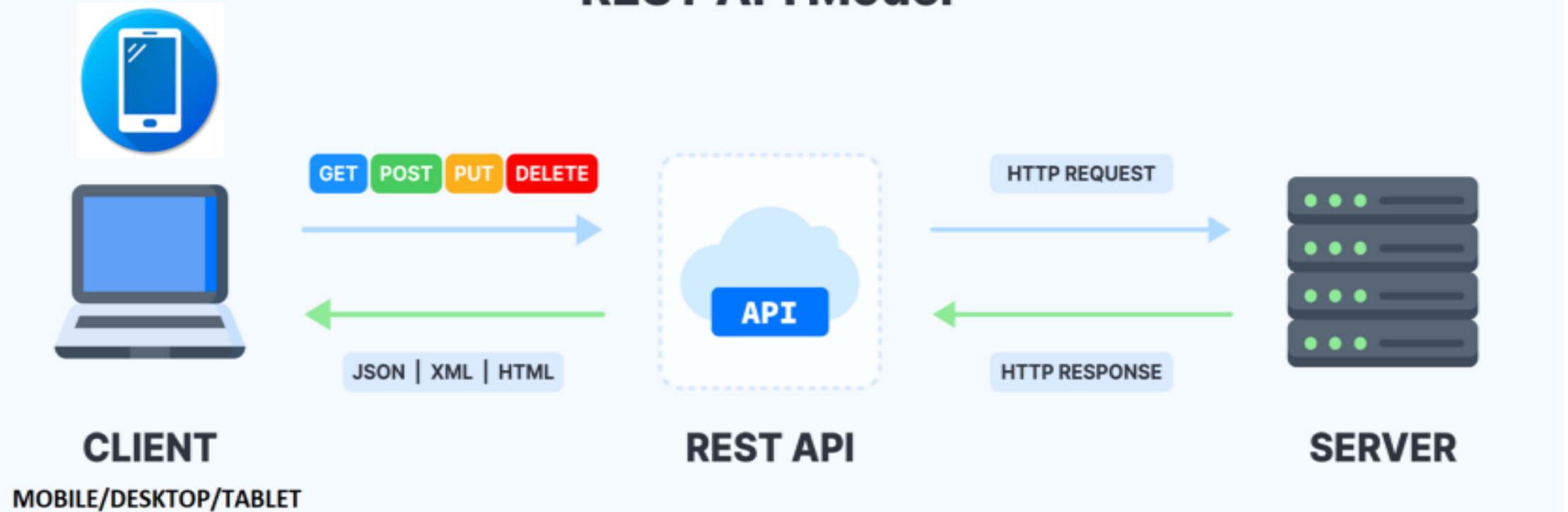
TEMARIO

- ¿Qué es un API?
- Swagger
- DBContext – ORM
- Ejemplos reales
- Subir a Produccion

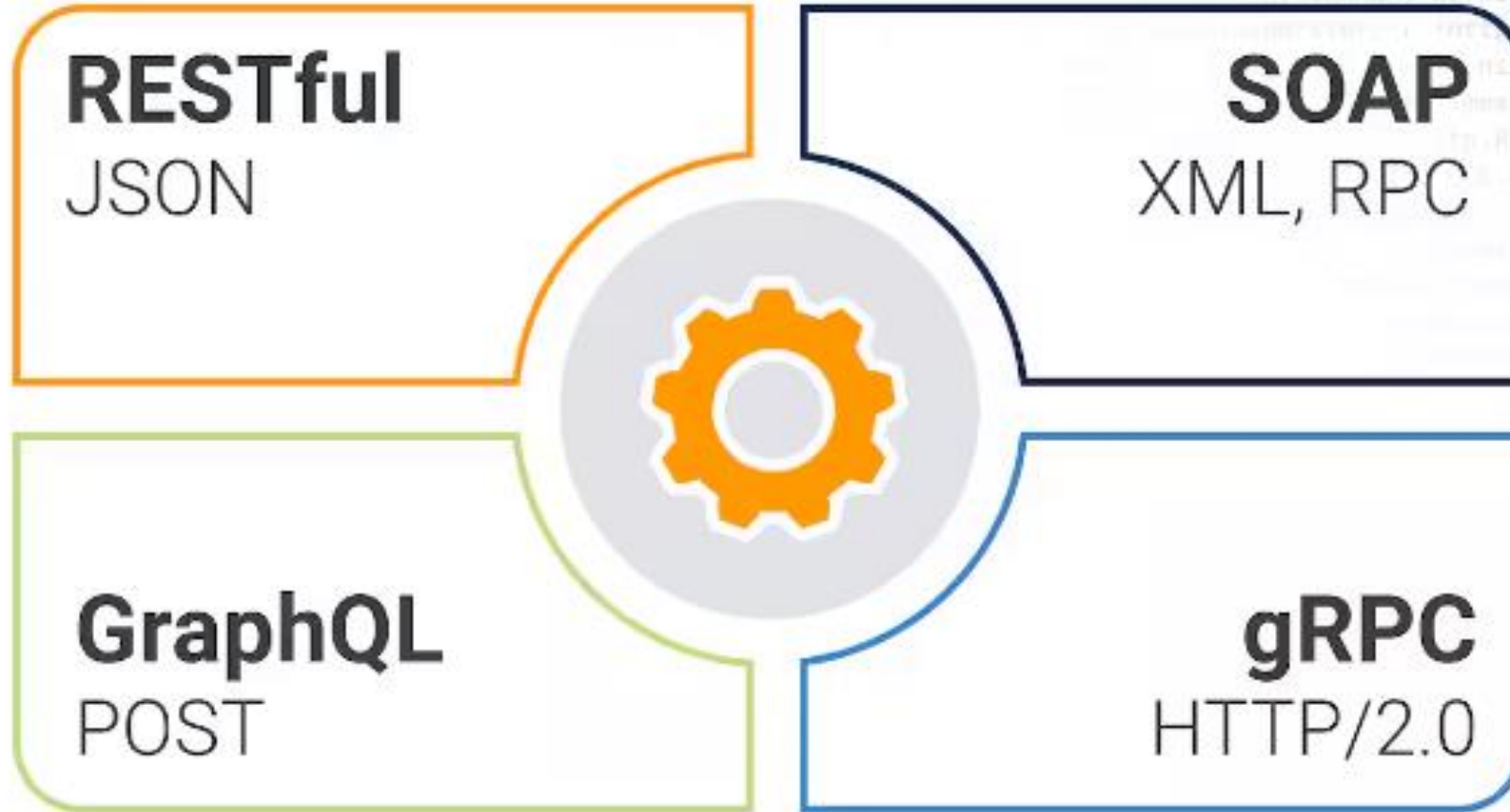
¿Qué es un API?

- Un **API** (siglas de **Interfaz de Programación de Aplicaciones**, en inglés **Application Programming Interface**) es un conjunto de definiciones y protocolos que permiten que diferentes programas o sistemas se comuniquen entre sí. En términos sencillos, un API es un intermediario que facilita la interacción entre aplicaciones o servicios, permitiendo que se intercambien datos o funcionalidades sin necesidad de que el usuario final intervenga directamente.

REST API Model



BY: ANKUSH AGNIHOTRI



Swagger

- **Swagger** es un conjunto de herramientas de código abierto que facilita el diseño, desarrollo, documentación y consumo de **APIs RESTful**. Su objetivo principal es hacer que las APIs sean más fáciles de crear, entender y usar.




<https://swagger.io/>


Platzi Fake Store API 1.0 OAS3


products

GET /api/v1/products 


POST /api/v1/products 


GET /api/v1/products/{id}  


PUT /api/v1/products/{id} 


DELETE /api/v1/products/{id} 

users

GET /api/v1/users 

POST /api/v1/users 

GET /api/v1/users/{id} 

PUT /api/v1/users/{id} 

- El patrón **OMR** es una técnica que permite mapear las **entidades del mundo real** (representadas como objetos en el código) a las **tablas de una base de datos relacional** y viceversa. Es una forma de conectar el modelo de objetos orientado a clases de un lenguaje de programación (como C#) con las estructuras tabulares de las bases de datos relacionales (como MySQL, SQL Server, PostgreSQL, etc.).

¿QUÉ ES UN ORM?



SQL

Para **hacer consultas** a una base de datos un programador **necesita escribir SQL**.



ORM

Un **ORM** **abstrae la base de datos** para que el programador haga consultas en el **lenguaje en el que está programando**, **sin necesitar SQL**.



LOS ORM MÁS POPULARES SON:



¡NUEVO CURSO!

Aprende SQL para, diseñar, crear y administrar información en:



ed.team/cursos/sql



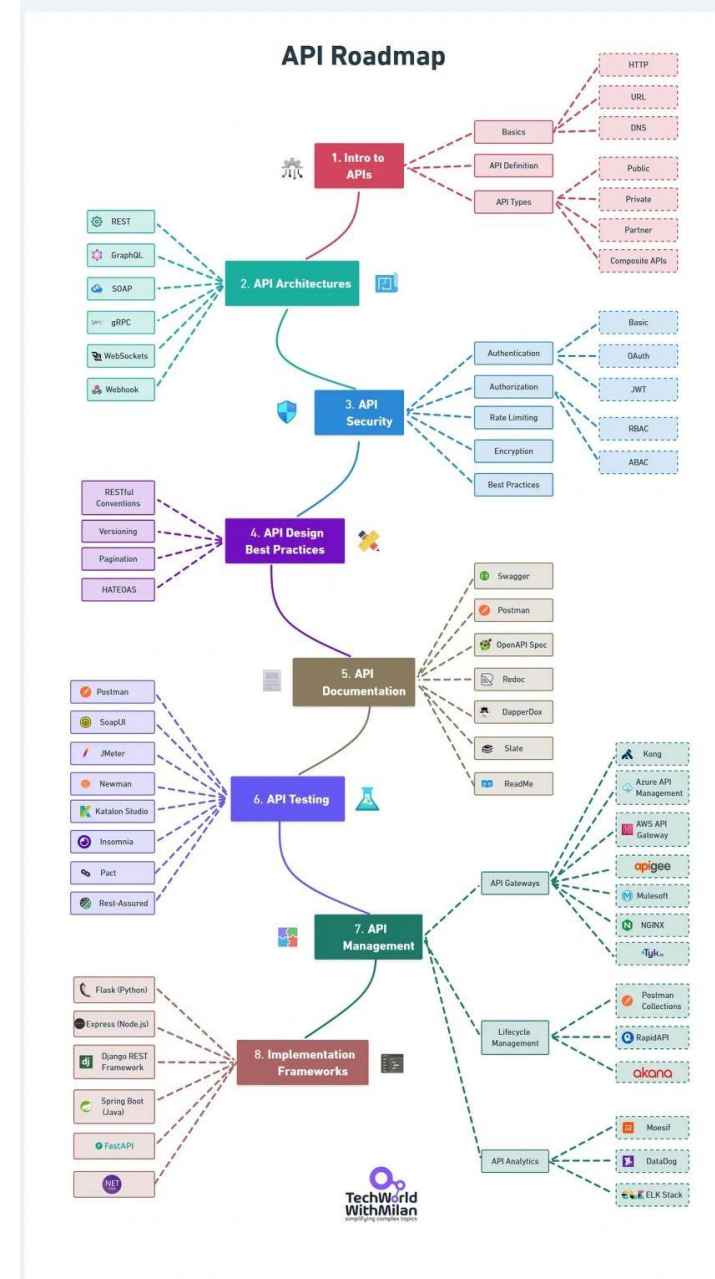
Hacienda

DbSet

- En **Entity Framework** (EF), un **DbSet<T>** es una clase que representa una colección de **entidades** de tipo T, donde T es una clase que define el modelo de datos (por ejemplo, una entidad o tabla en la base de datos). Un DbSet<T> se usa para realizar operaciones **CRUD** (Crear, Leer, Actualizar, Eliminar) sobre las entidades de la base de datos y se asocia directamente con una tabla en la base de datos.



https://www.linkedin.com/posts/sina-riyahi_backend-csharp-efcore-activity-7263124498361569281-ELJ1?utm_source=share&utm_medium=member_android





POSTMAN



Hacienda

Ejemplos Reales



<https://pokeapi.co/>



{ APIs }



<https://api.nasa.gov/>



<https://developer.marvel.com/>



<https://newsapi.org/>



Hacienda

TEMARIO

- Hacer Base de Datos con <https://postgres.new/>
- Api Rest
- Migraciones
- Seeders
- Request
- Ofuscación

Seeders

En el contexto de C# y desarrollo de aplicaciones, un **seeder** (o **datos de siembra**) se refiere a un proceso o clase que se utiliza para poblar una base de datos con datos iniciales o de prueba. Generalmente, se utiliza para insertar registros en las tablas de una base de datos de manera automática, a fin de tener datos con los cuales trabajar en el desarrollo o durante las pruebas de la aplicación.



<https://github.com/bchavez/Bogus>

JWT:

1. Instalar **Microsoft.AspNetCore.Authentication.JwtBearer**
2. Instalar **Microsoft.AspNetCore.Mvc.NewtonsoftJson**
2. Agregar servicio en **Startup.cs**
3. Crear 2 modelos: **User.cs**, **Sesion.cs**
4. Agregar SecurityKey en **appsettings.json**
5. Crear clase para encriptar y desencriptar contraseña
6. Crear **AuthController.cs**
7. Configurar endpoints con data annotation

GlobalExceptionHandler:

1. Crear Clase personalizada **GlobalExceptionHandler.cs**
2. Agregar servicio en el **Startup.cs**

Servicios Personalizados:

1. Crear interfaz del modelo **IProfesorServices.cs**
2. Crear Servicio del modelo **ProfesorServices.cs**
3. Crear dependencias personalizadas **DependencyInjection.cs**
4. Agregar servicio a **Startup.cs**





Hacienda

