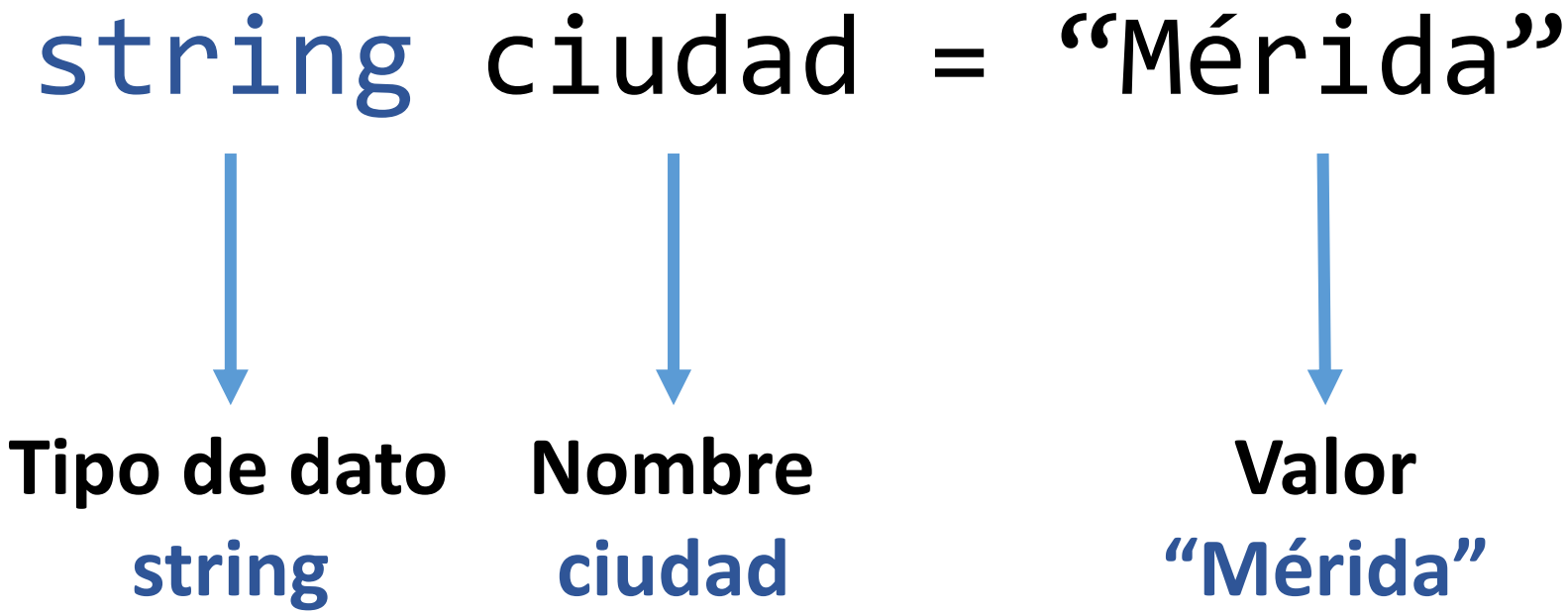
Variable: elemento cuyo valor puede cambiar


string ciudad = “Mérida”
Tipo de dato
string
Nombre
ciudad
Valor
“Mérida”

Tipos de datos

Texto:

- **string** – cadena de caracteres (palabras)
- **char** – un solo carácter

Números:

- **int** – enteros
- **decimal** – decimales

Booleanos:

- **bool** – verdadero o falso

Tipos de datos

Cualquier tipo:

- **var** – cualquier tipo de dato

Cuando no se sabe con certeza el tipo,
resultado de alguna operación

Operaciones aritméticas

P Paréntesis

E Exponentes

M Multiplicación

D División

A Adición

S Sustracción

Operadores relacionales

X == Z	X igual a Z
X != Z	X diferente a Z
X < Z	X menor a Z
X > Z	X mayor a Z
X <= Z	X menor o igual a Z
X >= Z	X mayor o igual a Z

Métodos

```
decimal CalcularIVA(decimal precio){…}
```

Valor de retorno
decimal

Nombre
CalcularIVA

Parámetro
precio

Bloque/ cuerpo

Arreglo: variable que contiene múltiples elementos del mismo tipo

Una vez que se define su tamaño, no se puede modificar

Lista: variable que contiene una colección de elementos del mismo tipo

A diferencia del arreglo, su tamaño es dinámico

Operadores lógicos

AND	X && Z	Verdadero cuando ambos son verdaderos
OR	X || Z	Verdadero cuando por lo menos uno es verdadero
NOT	!X	Modifica al valor opuesto

Trabajar con archivos

La estructura de los directorios varía entre los diferentes sistemas operativos.

Por ejemplo, la ruta al directorio Documents es la siguiente:

- Windows: C:\Users\'usuario'\Documents
- Linux: home\'usuario'\Documents

.NET incluye constantes y métodos para trabajar con rutas de archivos de manera constante independientemente del SO.

Excepciones

Errores que ocurren durante la ejecución de la aplicación.

try

Bloque de código a “probar”; se espera algún tipo de excepción.

catch

Bloque que se ejecuta para tratar con un tipo de excepción.

finally

Bloque que se ejecuta siempre, si ocurre una excepción o no (se utiliza en casos muy específicos).

Referencias y dependencias

using — Palabra clave (keyword) para referenciar un espacio de nombres (namespace) que contiene clases que queremos usar.

Podemos referenciar clases definidas por .NET (.NET Class Library), por nosotros mismos o por un tercero.

nuget es el administrador de paquetes (package manager) de .NET

Abstracción: expresar las características esenciales de un objeto, así como su comportamiento

¿Cuáles son sus características?
¿Qué acciones puede realizar?

Usuario

Características

- ID
- Nombre
- Fecha de Nacimiento
- Correo
- Fecha de Registro ...

Acciones

- Revisar cuenta
- Depositar a cuenta
- Retirar de cuenta
- Transferir a tercero ...

Clase

Una plantilla que define las características y acciones de los objetos de un cierto tipo.

- Las características se representan por medio de propiedades.
- Las acciones por medio de métodos.

Objeto

Un objeto es una entidad concreta basada en una clase.

Es una instancia de una clase.

Es una variable del tipo de la clase.

Constructores

Los constructores son métodos de una clase que nos permiten crear instancias (crear objetos) de la clase.

- Mismo nombre de la clase.
- No definen un valor de retorno.
- No hay un límite para la cantidad de constructores.

Encapsulamiento

Se refiere al ocultamiento del estado (el conjunto de propiedades) de una clase, para que no sea modificado directamente.

Sólo se puede acceder (obtener o modificar) a las propiedades a través de métodos.

De esta manera se protege el estado de la clase.

Modificadores de acceso

Nivel de acceso	Descripción
public	Sin restricción. Cualquier elemento tiene acceso.
private	Sólo tienen acceso los elementos de la clase.
protected	Los elementos de la clase y de clases hijas tienen acceso.
internal	Cualquier elemento dentro del ensamblado (proyecto) tiene acceso; '**public**' a nivel del ensamblado.

Sobrecarga de métodos

La sobrecarga significa que, en una clase, existen dos o más métodos con el mismo nombre.

La diferencia entre ellos es la cantidad y/o el tipo de parámetros.

Clases estáticas

Una clase estática no se puede instanciar; no se pueden crear objetos de ella.

Para acceder a los miembros de una clase estática se debe referenciar a la clase misma.

Herencia

La herencia es un mecanismo mediante el cuál se puede construir una jerarquía de clases.

En esta jerarquía, se definen clases hijas (o subclases) que heredan las propiedades y métodos de su clase padre.

- Usuario
  - Cliente
    - Personal
    - Institucional
  - Empleado
    - Gerente
  - Administrador
    - TI

Herencia

Al heredar las propiedades y métodos de los padres, se posibilita la reutilización de código.

A su vez, cada clase hija define propiedades y métodos propios, que la diferencian del padre.

Conforme se baja en la jerarquía, se desarrollan clases cada vez más específicas.

Sobre escritura de métodos

La sobre escritura ocurre cuando una o más clases hijas implementan un método de la clase padre.

El método es el mismo (mismo nombre y parámetros), pero tiene una implementación distinta en las clases hijas.

Sobre escritura de métodos

La sobre escritura es una aplicación del polimorfismo (muchas formas).

Es la capacidad de los objetos de implementar de manera diferente a un mismo método.

Se produce el mismo efecto u objetivo básico, pero la implementación es distinta.

Clases abstractas

Una clase abstracta no se puede instanciar; sólo sirve como clase base (padre).

Su propósito es proveer una definición común que múltiples clases derivadas pueden compartir.

- Puede definir elementos abstractos.
- Estos elementos deben ser implementados por las clases hijas.

Interfaces

Una interfaz tampoco se puede instanciar.

Su propósito también es proveer una definición común que múltiples clases pueden compartir.

- Todos sus elementos son abstractos.
- Una clase que implementa a una interfaz debe implementar a todos sus elementos.

Una clase puede heredar de sólo una clase, sea abstracta o no.

Sin embargo, una clase puede implementar múltiples interfaces.

git

Es un sistema de control de versiones; posibilita el almacenamiento de diferentes versiones de un archivo, para que este sea recuperable.

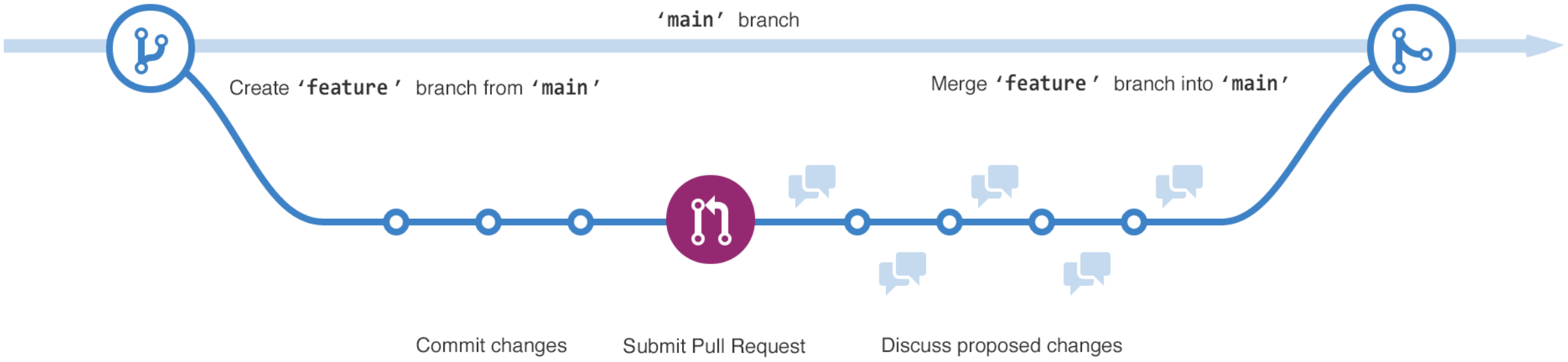
GitHub

Es una plataforma para alojar repositorios git; facilita la colaboración remota entre usuarios.

Flujo de trabajo en GitHub:

1. El repositorio remoto inicial es la rama (branch) main.
2. Creas una rama a partir de main.
3. Trabajas localmente en tu rama, guardando tus cambios (commit).
4. Publicas tu rama (push).
5. Solicitas un “pull request” para que tu rama sea parte de la rama main (merge).
6. Eliminas tu rama.

Flujo de trabajo en GitHub:



Tipos de información

Estructurada: se adhiere a un esquema fijo, toda la información tiene los mismos campos; el esquema es tabular.

- Bases de datos relacionales.

Semi-estructurada: hay un esquema pero no es rígido, no toda la información tiene los mismos campos.

- Bases de datos no-relacionales (documentos JSON).

No estructurada: no hay ningún tipo de esquema, la información no está contenida en campos.

- Documentos, imágenes, archivos de audio y video.

Tablas

En una **tabla**, todos los registros (filas) tienen la misma cantidad de columnas.

Persona

ID	Nombre	Edad
1	Pedro	20
2	Ana	21
3	Mario	19

Tablas

Cada columna en la tabla está definida por un tipo de dato.

Persona (columnas)

Tipo	Nombre
int	ID
varchar	Nombre
int	Edad

Tablas (llaves)

Una **llave primaria (PK)** identifica de manera única a los registros de una tabla.

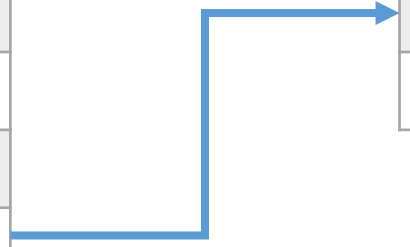
Una **llave foránea (FK)** establece una relación hacia la **llave primaria (PK)** de otra tabla.

Persona (columnas)

Tipo	Nombre
int (PK)	ID
varchar	Nombre
int	Edad
int (FK)	País

País (columnas)

Tipo	Nombre
int (PK)	ID
varchar	Nombre



SQL

SQL es un lenguaje para trabajar con bases de datos relacionales (RDBMS).

Los diferentes sistemas RDBMS tienen sus propias extensiones (o versiones) de SQL:

T-SQL

PL/SQL

pgSQL

SQL (comandos básicos)

Los comandos básicos de SQL entran en las siguientes categorías:

DDL	DML	DCL
• Data Definition Language • Definen objetos en la base de datos • **CREATE, ALTER, DROP**	• Data Manipulation Language • Para obtener y manipular información • **SELECT, INSERT, UPDATE, DELETE**	• Data Control Language • Para administrar los permisos de seguridad • **GRANT, REVOKE, DENY**

Normalización

La normalización consiste en diseñar adecuadamente el esquema de la BD para minimizar la duplicación de datos, reducir el espacio de almacenamiento y mejorar la calidad de la información.

En un esquema de BD “normalizado”:

- Las llaves primarias (PK) y foráneas (FK) se utilizan para definir relaciones.
- La información se obtiene al unir (join) tablas en una consulta.

Normalización

Persona

ID	Nombre	Edad	País
1	Pedro	20	México
2	Ana	21	EEUU
3	Mario	19	México

Persona

ID	Nombre	Edad	País
1	Pedro	20	1
2	Ana	21	2
3	Mario	19	1

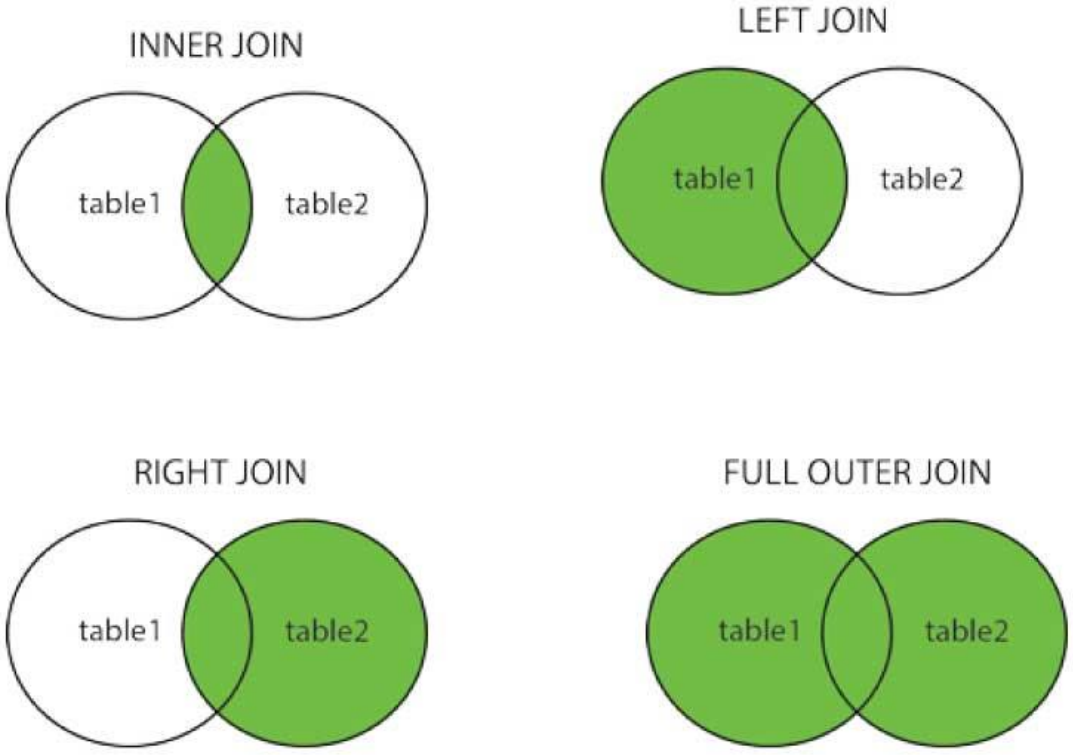
FK → PK

País

ID	Nombre
1	México
2	EEUU
3	Brasil

JOIN

Las sentencias JOIN permiten combinar los datos de dos o más tablas, en base a una columna relacionada entre ellas.



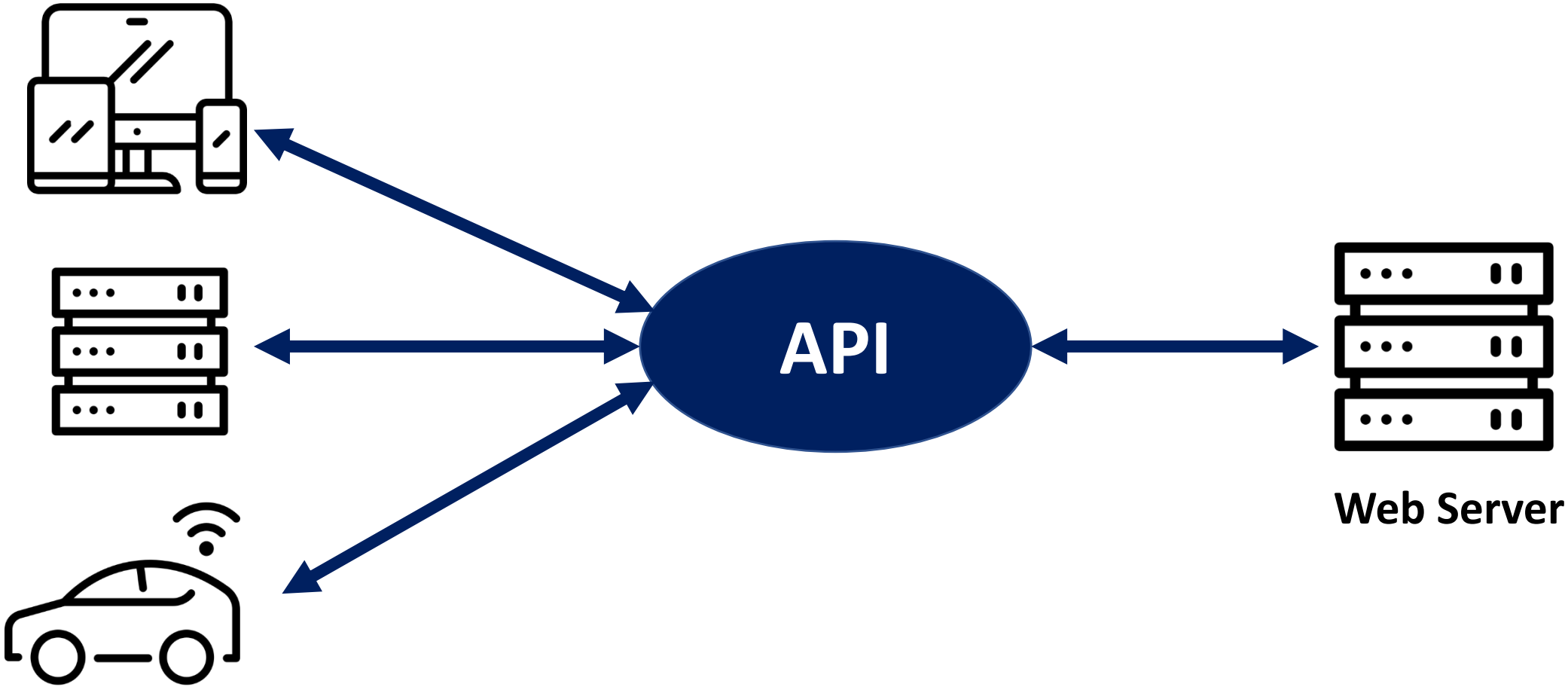
API REST

(Application Programming Interface) Es un componente de software para la comunicación con un servidor web.

Utiliza la arquitectura REST (Representational State Transfer) para el manejo de las solicitudes hacia el servidor.

- Las solicitudes REST utilizan el protocolo HTTP.

API REST



API REST

Los clientes que consumen la API no necesitan saber cómo se procesa la información ni de dónde proviene.

Verbos HTTP

GET: Obtener información del servidor web

POST: Crear un nuevo elemento en el servidor

PUT: Actualizar un elemento existente

DELETE: Eliminar un elemento

API REST

Por lo tanto, una API REST consiste en:

- Una URI
- Verbos HTTP
- Un formato para los datos

GET (text/json)

https:/api.banco.com/cliente

Entity Framework Core

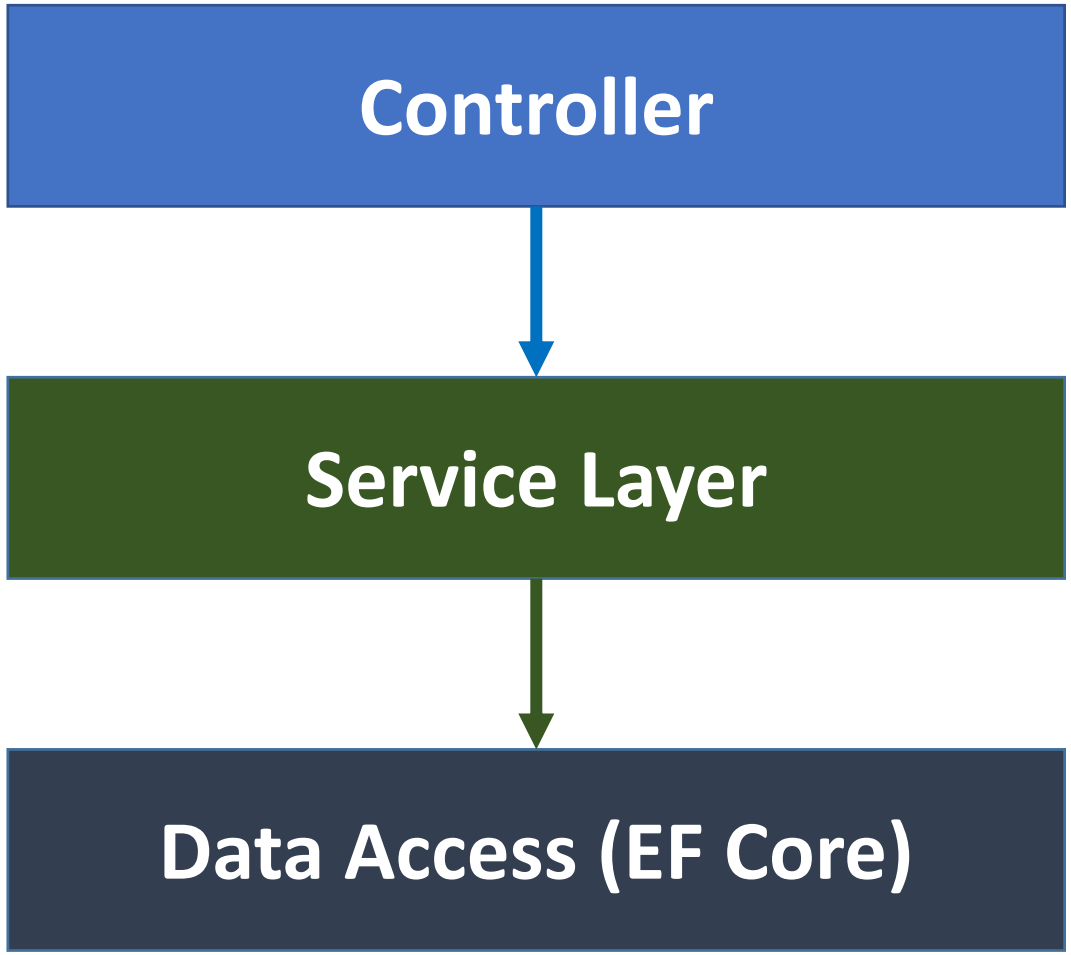
EF Core es un ORM (object-relational mapper). Los ORMs proveen una capa entre nuestro código y la BD.

- Las tablas de la BD se representan como clases en el código.
- Se utiliza LINQ para obtener y modificar información de la BD.
- El enfoque puede ser Code First o Database First.

HTTP Status Codes

Código	Descripción
200	**OK**. La solicitud (**GET**) fue exitosa y la respuesta incluye el recurso solicitado.
201	**Created**. La solicitud (**POST**) fue exitosa: se ha creado un nuevo recurso y se devuelve en el cuerpo de la respuesta.
204	**No Content**. La solicitud (**PUT**) fue exitosa pero la respuesta no tiene ningún contenido.
400	**Bad Request**. El servidor no puede procesar la solicitud debido a una sintaxis inválida en ella (error del cliente).
404	**Not Found**. El servidor no puede encontrar el recurso solicitado (error del cliente).
500	**Internal Server Error**. El servidor encontró una situación inesperada que le impide procesar la solicitud.

Arquitectura de Tres Capas



Operaciones asíncronas

Las operaciones síncronas se ejecutan en secuencia; una debe terminar antes de que se ejecute la siguiente.

Las operaciones asíncronas se ejecutan a la par; no es necesario que una termine para que se ejecute la siguiente.

- Las palabras clave async/await se usan en conjunto.
- La clase Task representa una operación asíncrona.