Programación 3

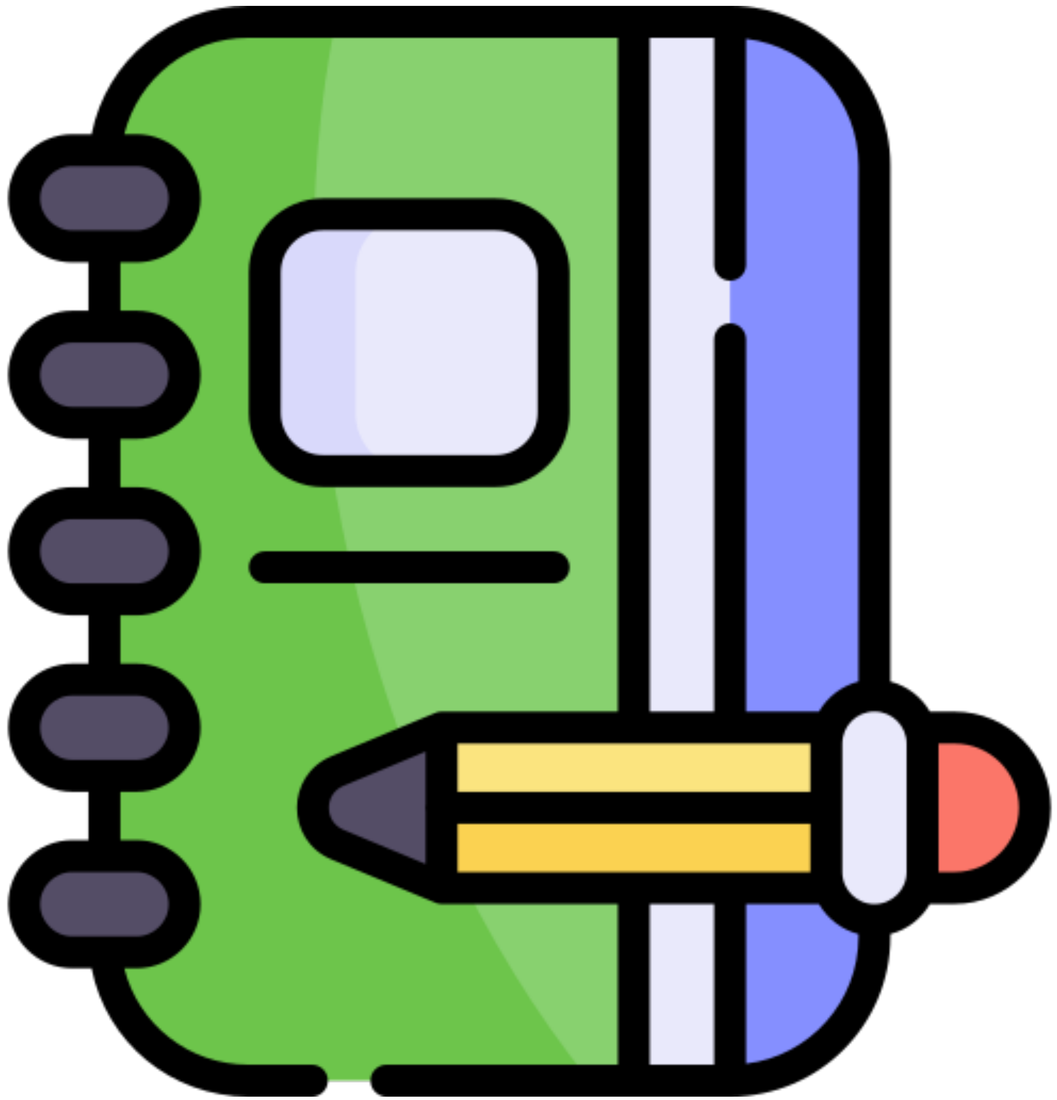
Programación Distribuida

Dr. Andrés Melgar

Programación Distribuida

Agenda

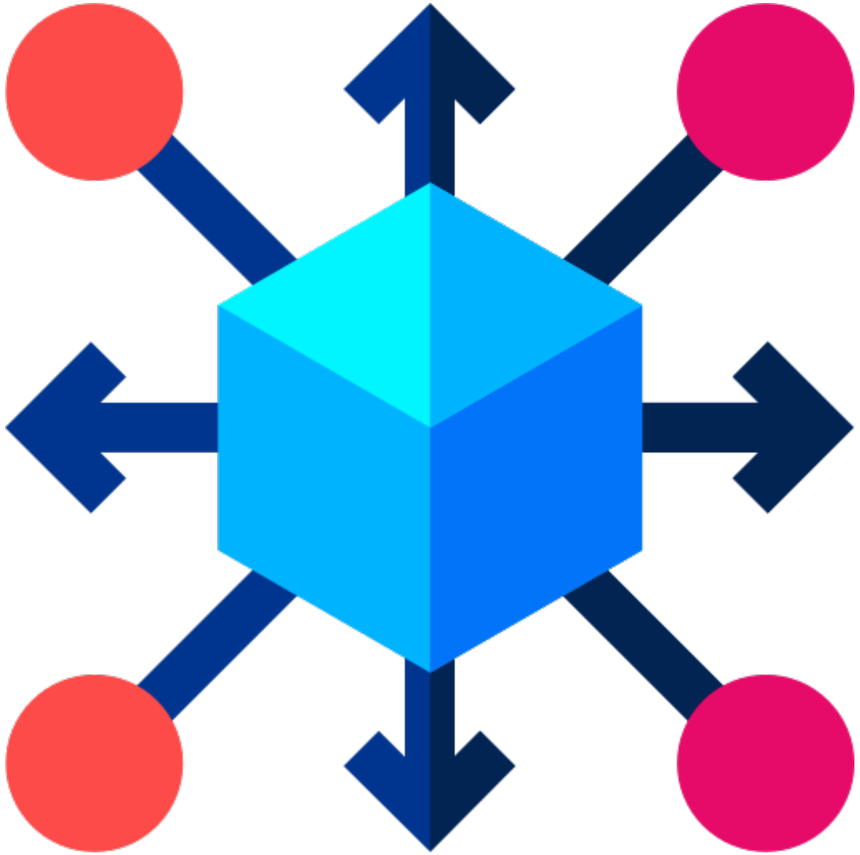
- Introducción a la Programación Distribuida
- Conceptos básicos de Comunicación en Red
- Servicios Web
- Servicios Web con SOAP
- Servicios Web con RESTful
- Arquitectura de nuestro proyecto



Programación Distribuida

Definición

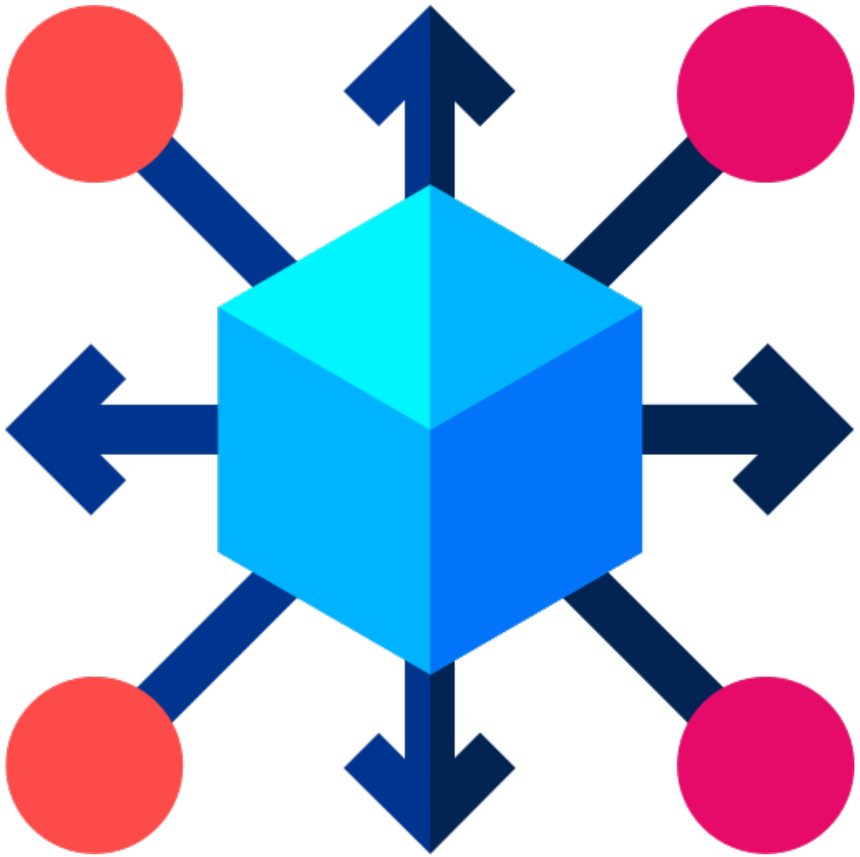
- Es un paradigma de desarrollo en el cual el software está compuesto por **componentes** que se ejecutan en **múltiples computadoras** que se comunican entre sí a través de una **red**.
- En el contexto moderno, se usa en:
  - Microservicios
  - Aplicaciones móviles/web
  - IoT
  - Servicios en la nube



Programación Distribuida

Características

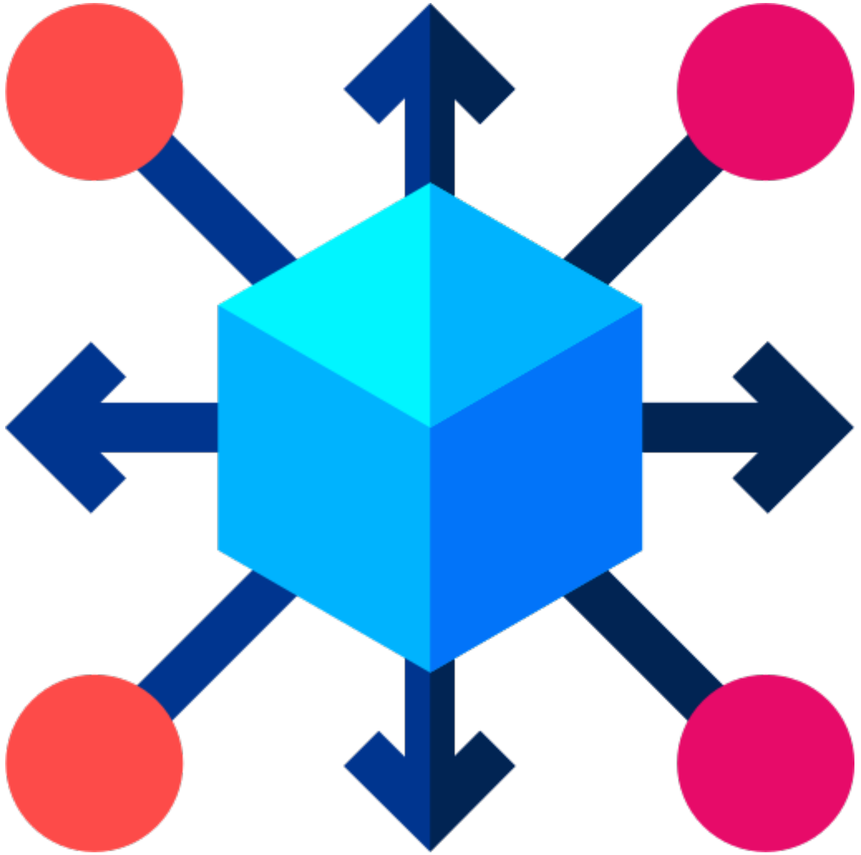
- **Concurrencia**: múltiples procesos se ejecutan en paralelo.
- **Escalabilidad**: capacidad para crecer eficientemente al aumentar nodos.
- **Transparencia**:
  - De ubicación (no se ve dónde está cada recurso)
  - De acceso (la forma de acceder es uniforme)
  - De concurrencia (manejo automático de múltiples usuarios)
  - De replicación y fallos



Programación Distribuida

Comunicación

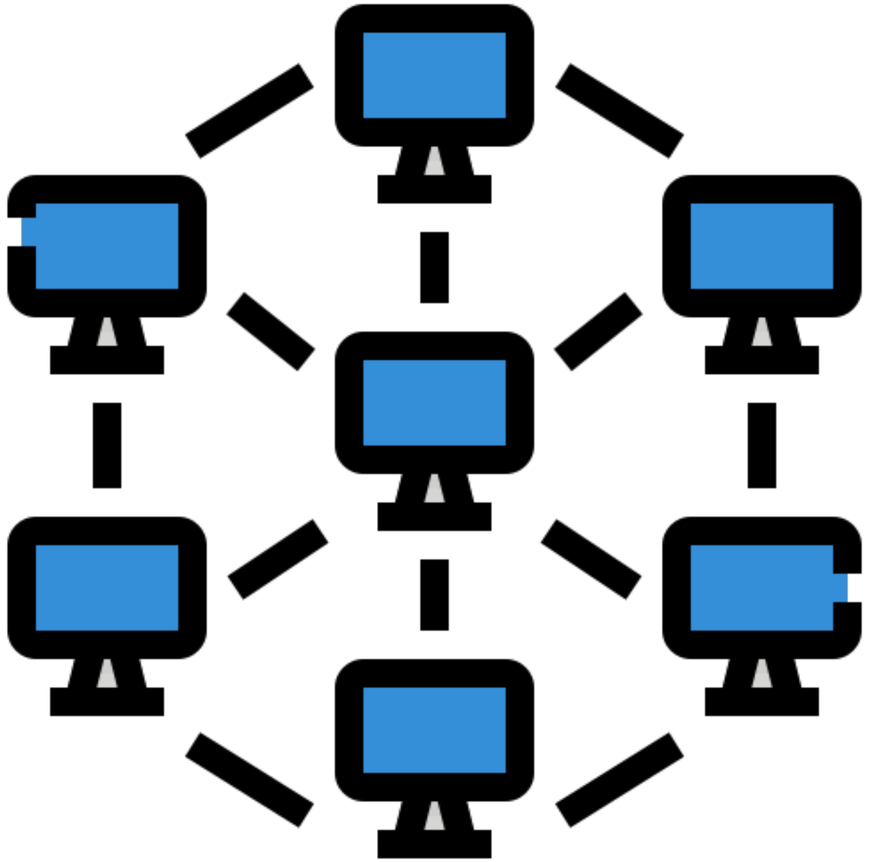
- **Intercambio de mensajes**: base de la comunicación distribuida.
- **Modelos de interacción**:
  - **Cliente-servidor**
  - **Peer-to-peer**: Todos los nodos pueden actuar como clientes y servidores al mismo tiempo.
- **Mecanismos de comunicación**:
  - **RPC (*Remote Procedure Call*)**: Permite que un programa invoque funciones o procedimientos ubicados en otra máquina, como si fueran locales.
  - **RMI (*Remote Method Invocation*)**: Versión orientada a objetos del RPC, específica de Java.
  - ***Sockets***: Permiten la comunicación a bajo nivel entre procesos a través de la red. Requieren que el programador gestione detalles como la conexión, envío y recepción de datos.
  - **Servicios Web**: Permiten la comunicación entre aplicaciones en distintos lenguajes o plataformas usando protocolos estándar (como HTTP).



Conceptos básicos de Comunicación en Red

TCP/IP (*Transmission Control Protocol/Internet Protocol*)

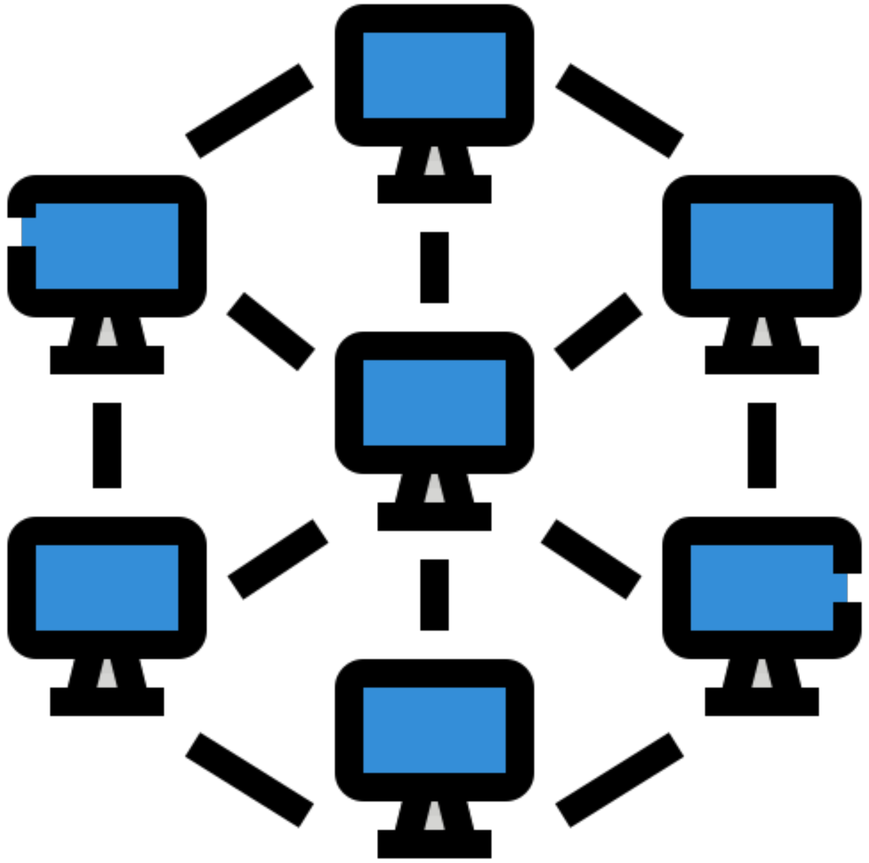
- **IP (*Internet Protocol*)**
  - IP es como si fuera el “sistema de direcciones de Internet".
  - Cada dispositivo conectado a una red tiene una dirección IP única (como la dirección de su casa).
  - IP se encarga de que los paquetes de datos lleguen de un punto A a un punto B, sin preocuparse por el orden o la confiabilidad.
  - Es como el servicio postal que entrega una carta, pero no se asegura de que llegue completa o en el orden correcto si envías varias.
- **¿Para qué sirve?**
  - Identificar el servidor y el cliente en la red.



Conceptos básicos de Comunicación en Red

TCP/IP (*Transmission Control Protocol/Internet Protocol*)

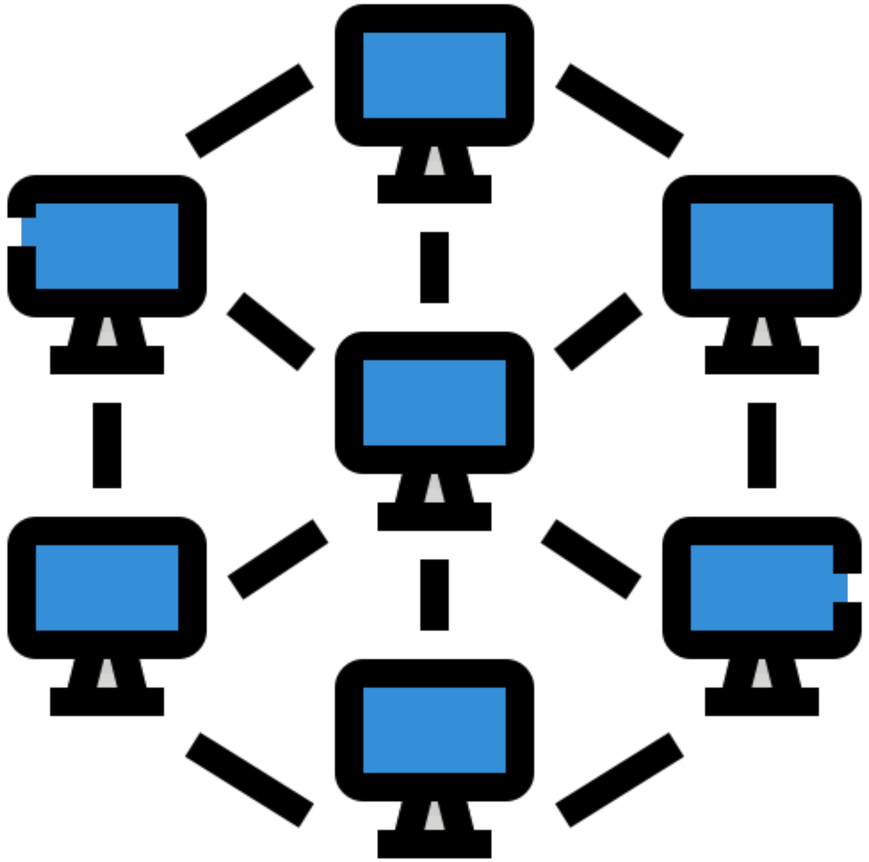
- **TCP (*Transmission Control Protocol*)**
  - TCP es el “asegurador” de la comunicación.
  - Trabaja sobre IP y garantiza que los paquetes de datos se entreguen de forma confiable, ordenada y sin errores.
  - Establece una conexión entre dos puntos, la mantiene mientras dura la comunicación y la cierra cuando finaliza.
  - Es como una conversación telefónica donde ambos interlocutores saben si el mensaje fue recibido y en qué orden.
- ¿Para qué nos sirve?
  - Los servicios web confían en TCP para una comunicación robusta y garantizada.
  - Si un paquete de datos se pierde o llega desordenado, TCP se encarga de retransmitirlo o reordenarlo.



Conceptos básicos de Comunicación en Red

HTTP (*Hypertext Transfer Protocol*)

- **Definición**
  - HTTP es un protocolo de aplicación (capa superior del modelo TCP/IP) para la transmisión de hipertexto.
  - Permite a los navegadores web solicitar páginas web y a los servidores web enviarlas.
- **Modelo Petición-Respuesta**: HTTP funciona con un modelo simple:
  - un cliente envía una petición a un servidor,
  - y el servidor envía una respuesta al cliente.
- Este es el protocolo sobre el que se construye gran parte de la web moderna, y es el pilar fundamental de los servicios web RESTful, aunque también es el protocolo de transporte subyacente para SOAP.



Conceptos básicos de Comunicación en Red

Serialización

- Es el proceso de convertir un objeto (o un grafo de objetos) en un formato que puede ser almacenado, transmitido o reconstruido más tarde.
- Básicamente, tomamos la estructura de datos en memoria y la transformamos en una secuencia lineal de bytes (o un formato de texto estructurado).
- **¿Por qué lo necesitamos?**
  - Para enviar datos a través de la red, debemos “aplanar" nuestros objetos en un formato que pueda ser transmitido bit a bit.

Conceptos básicos de Comunicación en Red

Serialización

```
public class Producto {
    private int id;
    private String nombre;
    private BigDecimal precio;
    ...
}
```

```
{"id": 1,
"nombre": "Laptop",
"precio": 1200.50}
```


```
<Producto>
    <Id>1</Id>
    <Nombre>Laptop</Nombre>
    <Precio>1200.50</Precio>
</Producto>
```

Conceptos básicos de Comunicación en Red

Deserialización

- Es el proceso inverso a la serialización.
- Toma la secuencia de bytes (o el formato de texto estructurado) que se recibió a través de la red y lo convierte de nuevo en un objeto funcional en la memoria del programa receptor.
- **¿Por qué lo necesitamos?**
  - Para que el programa receptor pueda trabajar con los datos recibidos como objetos nativos de su lenguaje.

Servicios Web

Definición

- Son **interfaces** que permiten que diferentes aplicaciones se comuniquen entre sí a través de una red (generalmente Internet) usando estándares abiertos.
- Actúan como una **“puerta” de acceso** a funciones o datos que ofrece una aplicación, de forma que otros sistemas puedan invocar funciones remotamente como si fueran locales.
- Un servicio web permite que un sistema exponga **funcionalidades** para que otros sistemas las consuman, sin importar en qué lenguaje están programados.

Servicios Web

Características

- **Interoperabilidad**: Funciona entre sistemas escritos en diferentes lenguajes y plataformas.
- **Uso de HTTP**: La mayoría se comunica sobre el protocolo HTTP.
- **Formato de intercambio**: Usan formatos como XML o JSON para enviar datos.
- **Basados en estándares**: Siguen especificaciones abiertas como WSDL, XML, JSON, HTTP, etc.
- **Desacoplamiento**: El cliente no necesita conocer la implementación interna del servicio.

Servicios Web

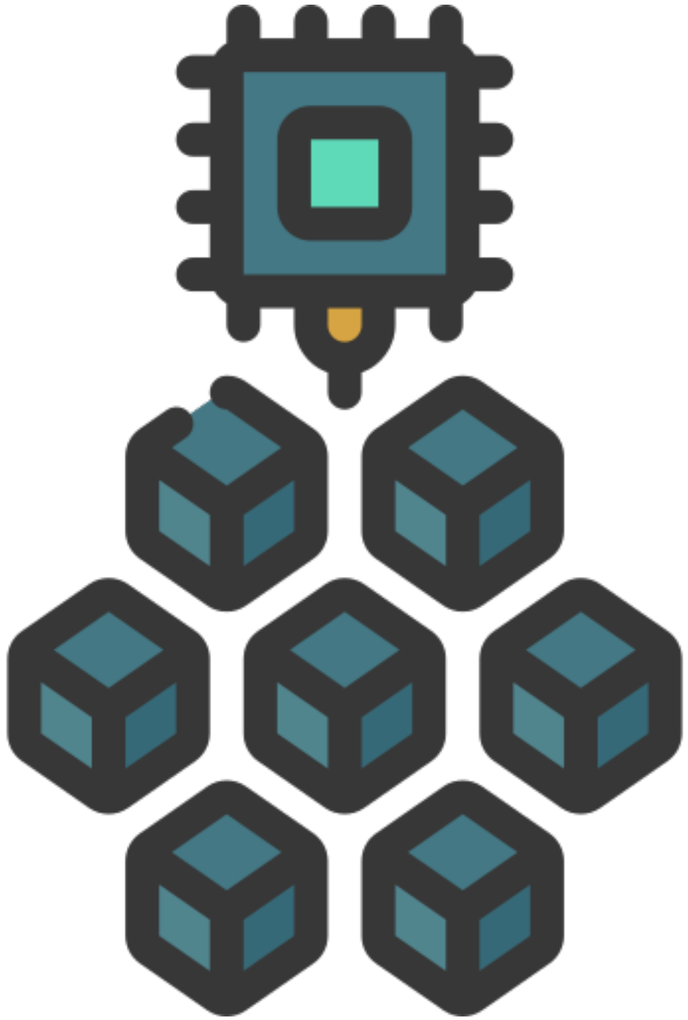
¿En donde se usan?

- Integración entre sistemas internos (ej. ERP + CRM).
- Publicación de APIs para terceros (ej. Google Maps, Stripe).
- Comunicación entre microservicios.
- Conexión entre front-end (por ejemplo una app web en C#) y back-end en Java.

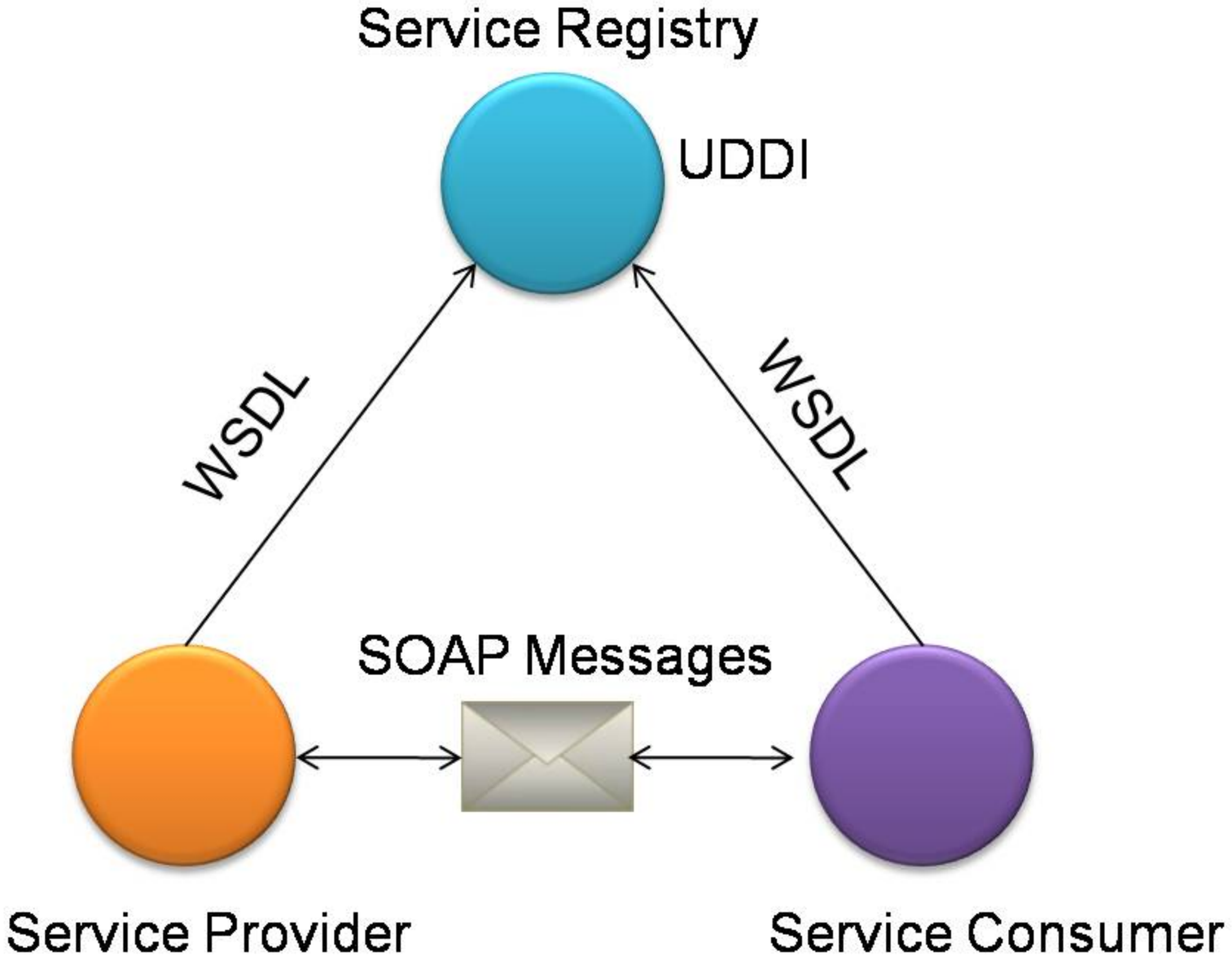
SOAP (*Simple Object Access Protocol*)

Definición

- **SOAP es un protocolo.**
  - Esto significa que es un conjunto de reglas bien definidas para la comunicación.
  - Su propósito principal es permitir que los programas en diferentes sistemas operativos y lenguajes de programación se comuniquen utilizando XML sobre un protocolo de red como HTTP.
- No es solo un protocolo de transporte, sino un **“sobre”** para mensajes.
  - Piensen en él como una forma estandarizada de empaquetar información estructurada para el intercambio entre aplicaciones.
- Nació de la necesidad de un estándar para la comunicación entre aplicaciones distribuidas que fuera más robusto y formal que los enfoques anteriores, especialmente para entornos empresariales críticos.



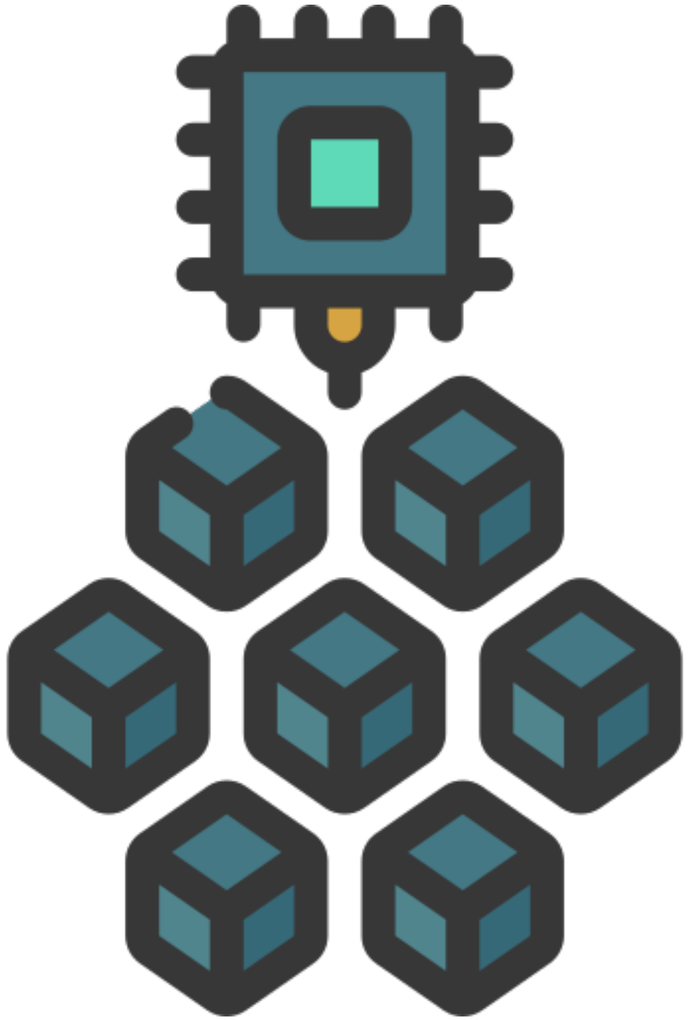
SOAP (*Simple Object Access Protocol*)



SOAP (*Simple Object Access Protocol*)

WSDL (*Web Services Description Language*)

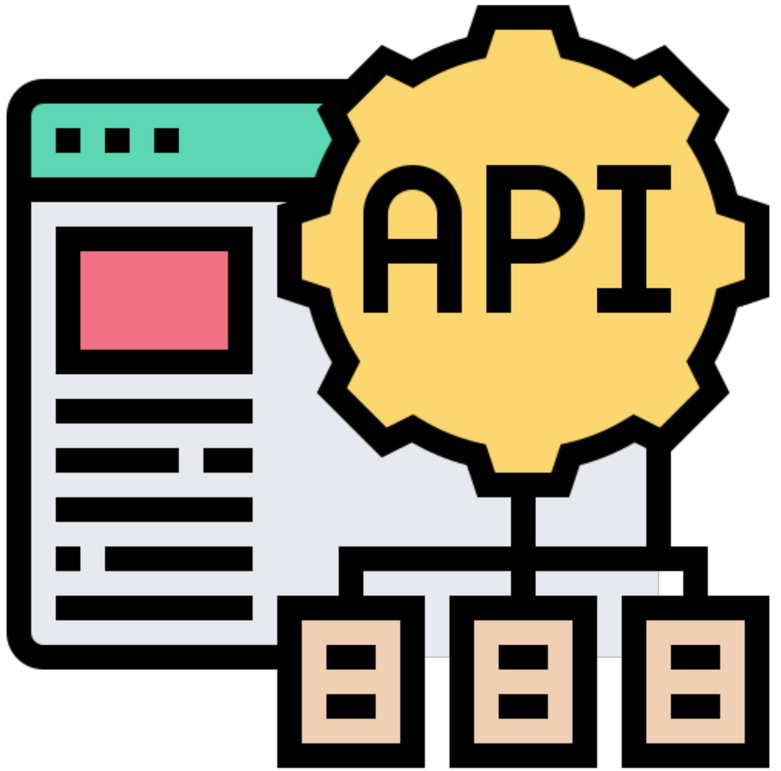
- Es un lenguaje basado en XML para describir servicios web.
- Es un contrato formal que especifica qué operaciones ofrece un servicio web, cómo se llaman esas operaciones, qué parámetros esperan y qué tipo de datos devuelven.
- Documento XML que describe el contrato del servicio (operaciones, entradas, salidas, tipos de datos). Permite a los clientes generar proxies automáticamente.



RESTful

REST (*Representational State Transfer*)

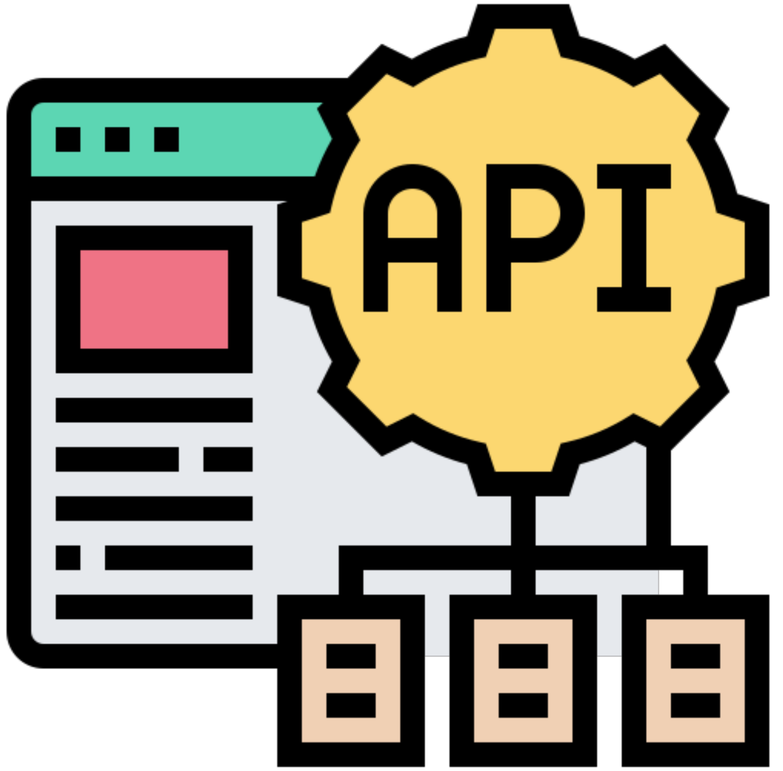
- REST no es un protocolo. Es un estilo arquitectónico para sistemas distribuidos hipermedia.
- Fue definido por Roy Fielding en su tesis doctoral en el año 2000, analizando los principios que hicieron exitosa a la *World Wide Web*.
- REST se basa en los principios de la Web. La idea es que los servicios web RESTful se comporten como recursos de la web, siendo accesibles a través de URIs y manipulables mediante los métodos estándar de HTTP.
- Piensen en cómo interactúan con un sitio web: hacen clic en enlaces (URLs), envían formularios, etc. REST busca aplicar esa misma simplicidad y escalabilidad a la comunicación entre aplicaciones.



RESTful

Principios REST (*Representational State Transfer*)

- **Cliente-Servidor:** Clara separación de las preocupaciones entre la interfaz de usuario (cliente) y el almacenamiento de datos (servidor). Cada uno puede evolucionar de forma independiente.
- **Stateless (Sin estado)**: Cada solicitud del cliente al servidor debe contener toda la información necesaria para que el servidor la entienda y la procese. El servidor no debe almacenar ninguna información de sesión sobre el cliente entre las solicitudes. Esto mejora la escalabilidad y la fiabilidad.
- **Cacheable**: Las respuestas a las solicitudes deben poder ser almacenadas en caché (cacheables) o no. Si una respuesta es cacheable, el cliente y la infraestructura intermedia pueden reutilizar esa respuesta para solicitudes futuras, mejorando el rendimiento y la escalabilidad.
- **Uniform Interface (Interfaz Uniforme)**: Este es quizás el principio más crucial de REST. Simplifica y desacopla la arquitectura al tener una forma consistente de interactuar con todos los recursos:



RESTful

Verbos HTTP

Método	Uso	Ejemplo
GET	Obtener un recurso	`GET /productos/1`
POST	Crear un recurso	`POST /productos`
PUT	Actualizar un recurso	`PUT /productos/1`
DELETE	Eliminar un recurso	`DELETE /productos/1`
PATCH	Actualización parcial (opcional)	`PATCH /productos/1`

Arquitectura de SW (estilo arquitectónico monolítico por capas)

Presentación

Front-End

Microsoft .NET

Lógica de Negocio

Dominio

Persistencia

Back-End

Microsoft .NET

BussinessObject

+regla_de_negocio()

Patrón DAO

(patrón de interacción con BD Data Mapper)

Model

-campo1
-campo2

Patrones:

- Data Transfer Object (diseño)
- Identity Field (persistencia)
- Sigle Table Inherence (persistencia)

DataAccessObject

+insertar()
+actualizar()
+eliminar()
+consultar()

DataAccessObjetImplementation

-conexion
-resultSet

+insertar()
+actualizar()
+eliminar()
+consultar()

Base de datos

MySQL

Microsoft® SQL Server®

Arquitectura de SW (estilo basado en servicios)

Presentación

Front-End

Microsoft .NET

Servicio Web

Lógica de Negocio

Dominio

Persistencia

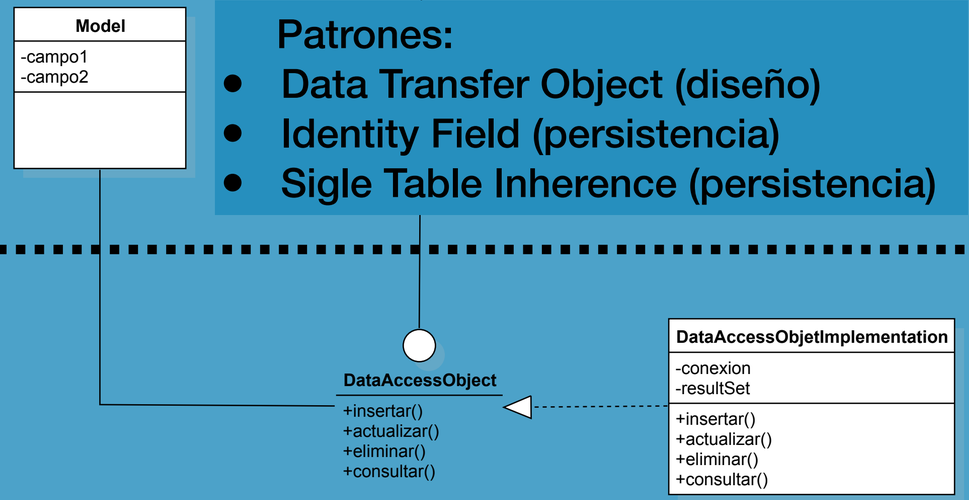
Back-End

Base de datos

Java

Patrón DAO

(patrón de interacción con BD Data Mapper)

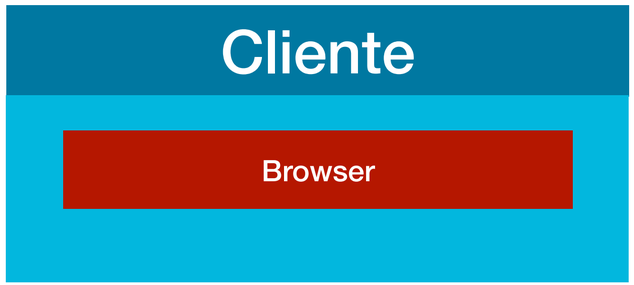


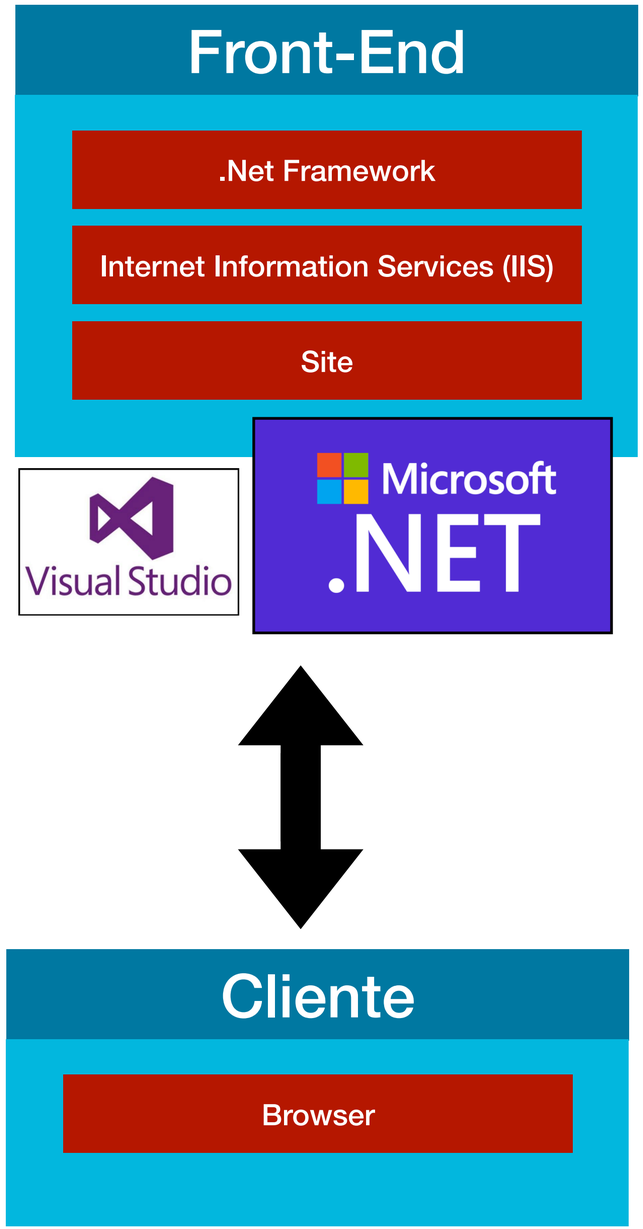
Patrones:

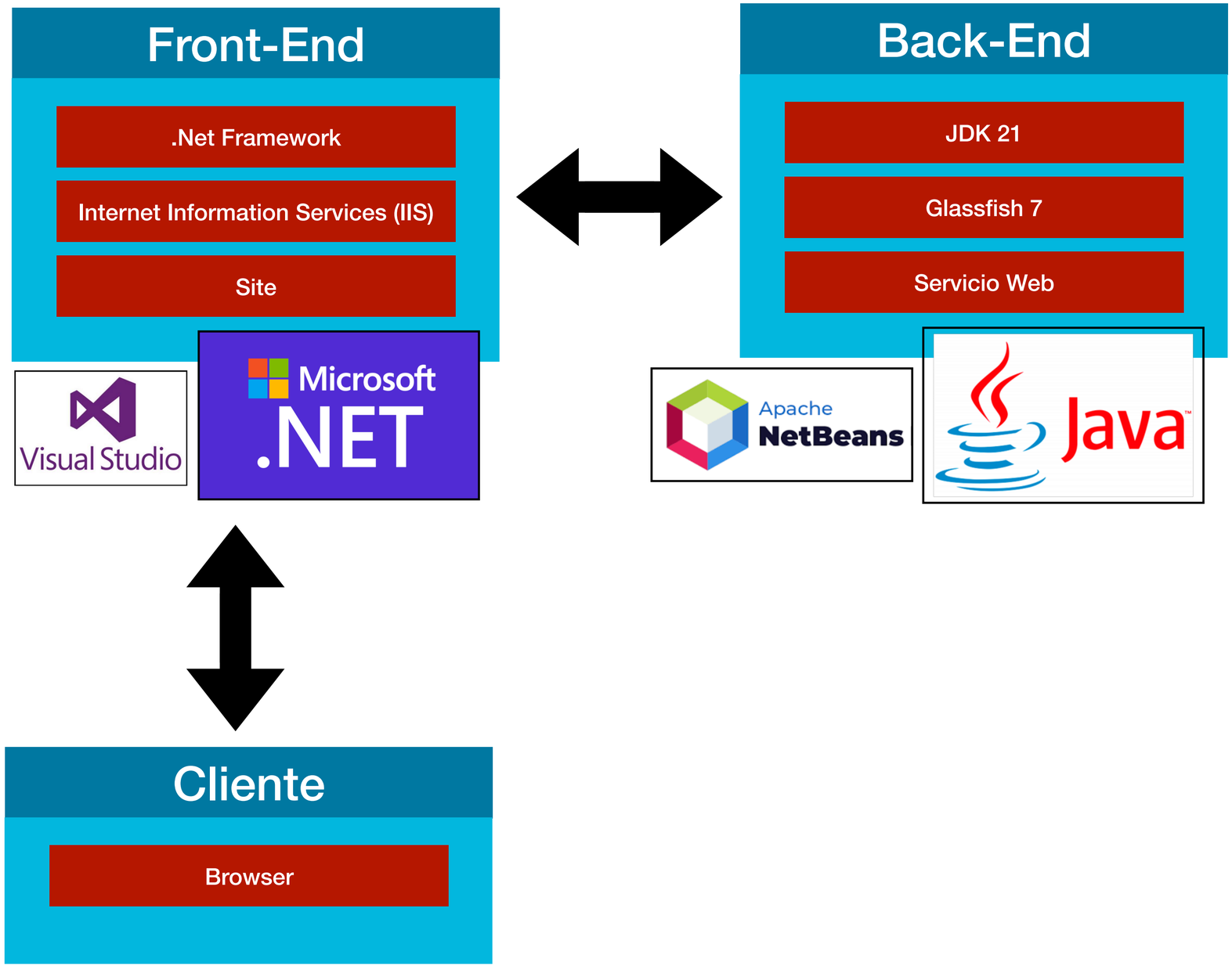
- Data Transfer Object (diseño)
- Identity Field (persistencia)
- Sigle Table Inherence (persistencia)

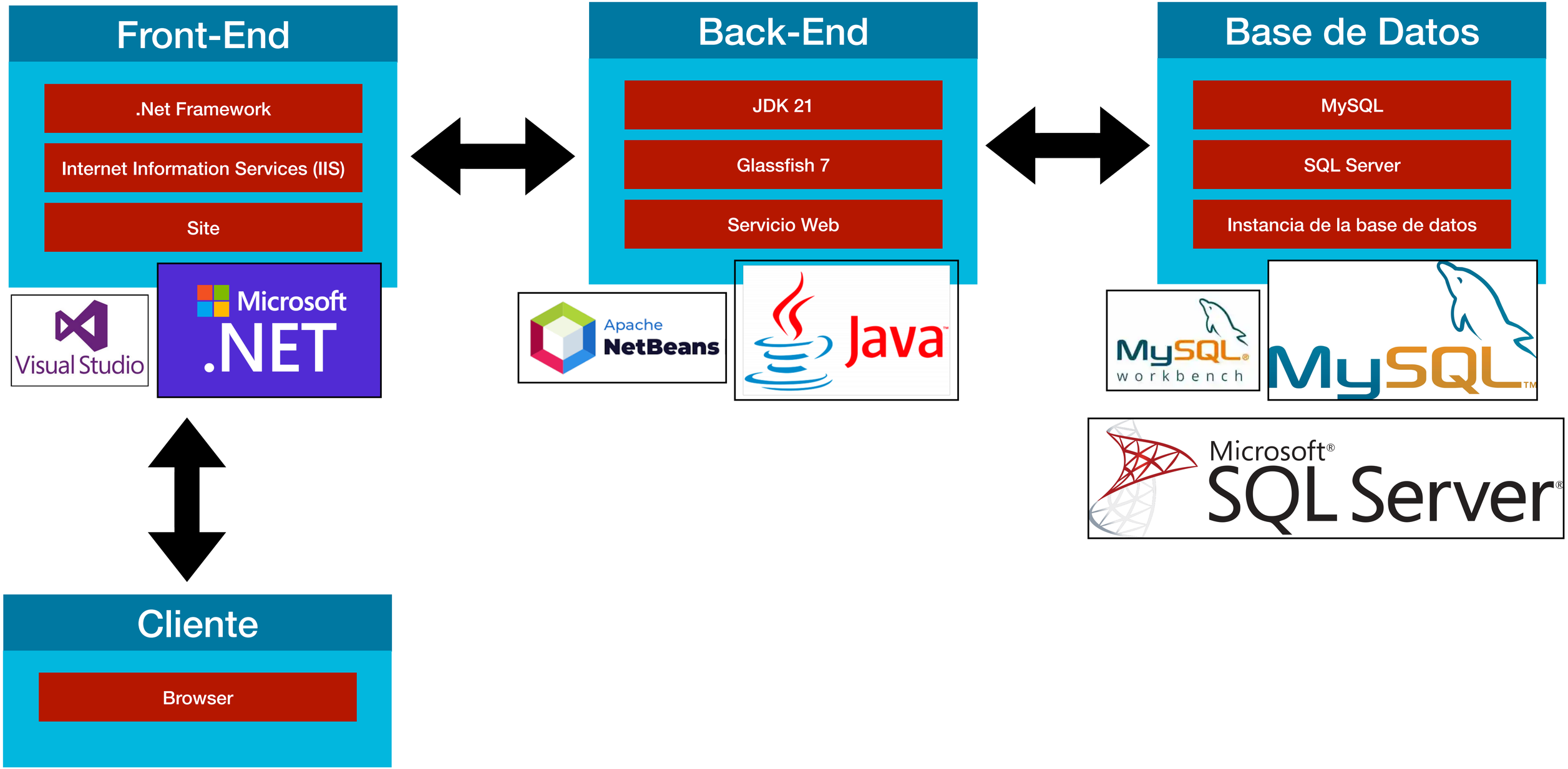
MySQL

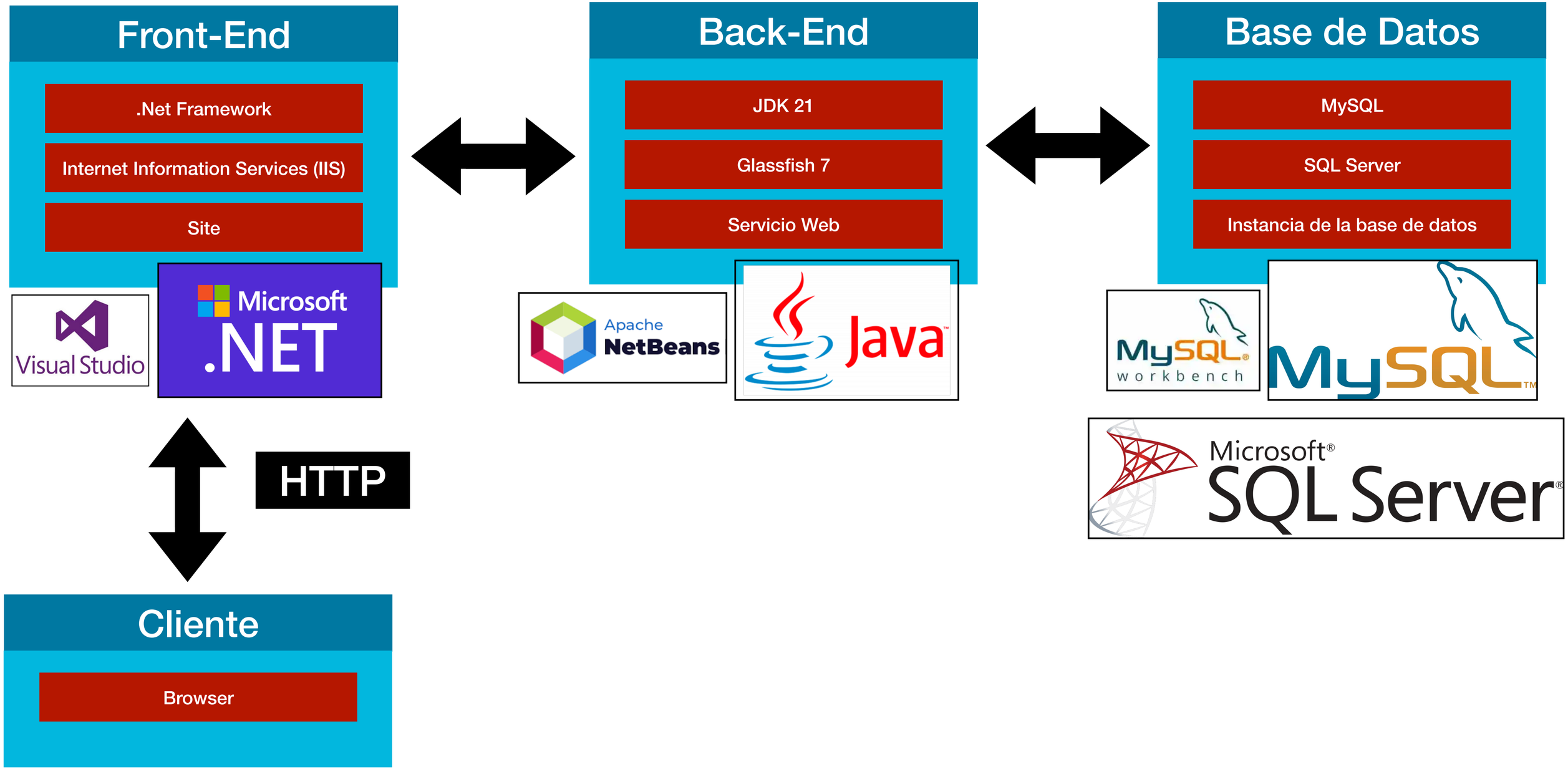
Microsoft® SQL Server®

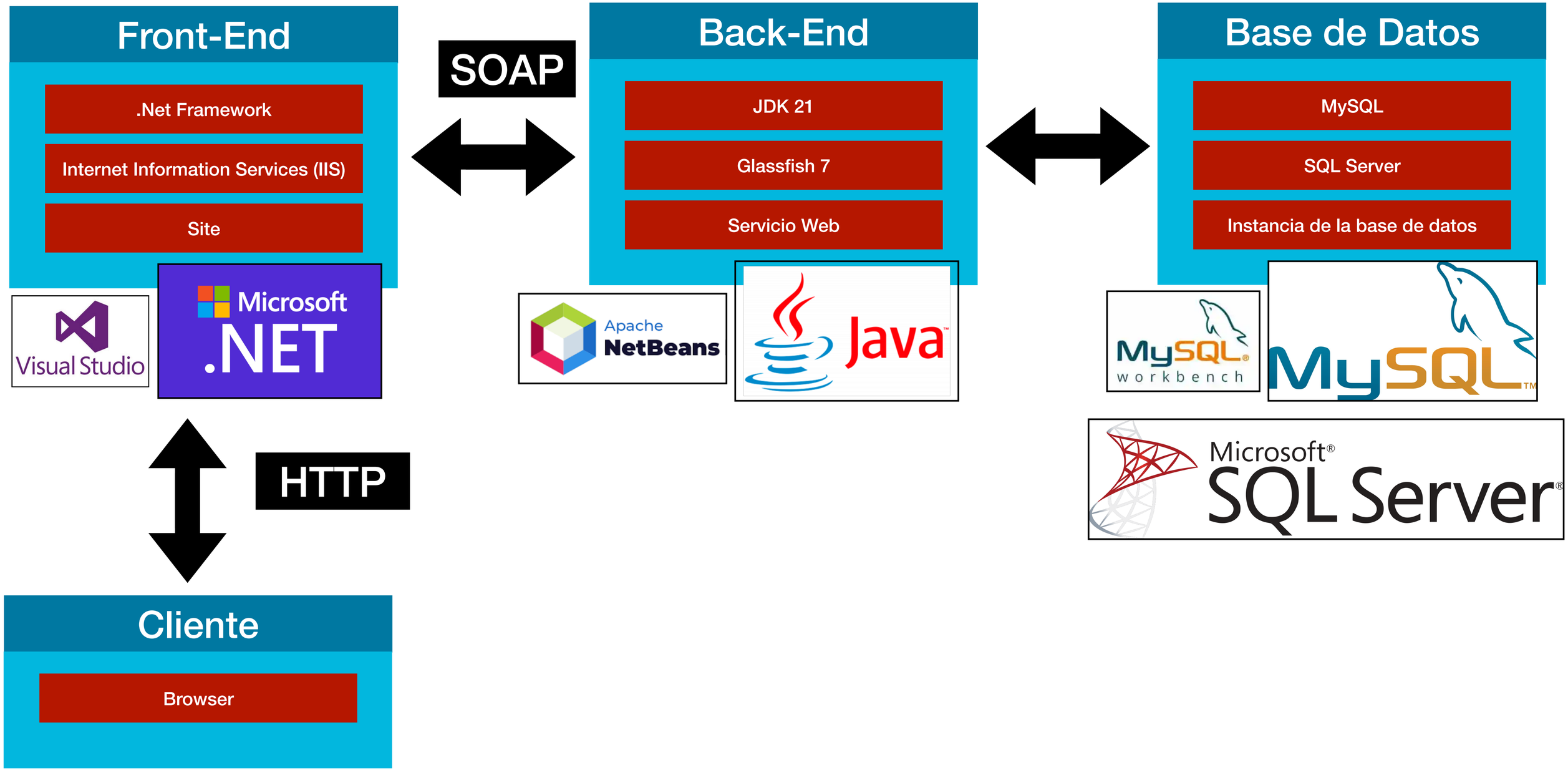

Cliente
Browser

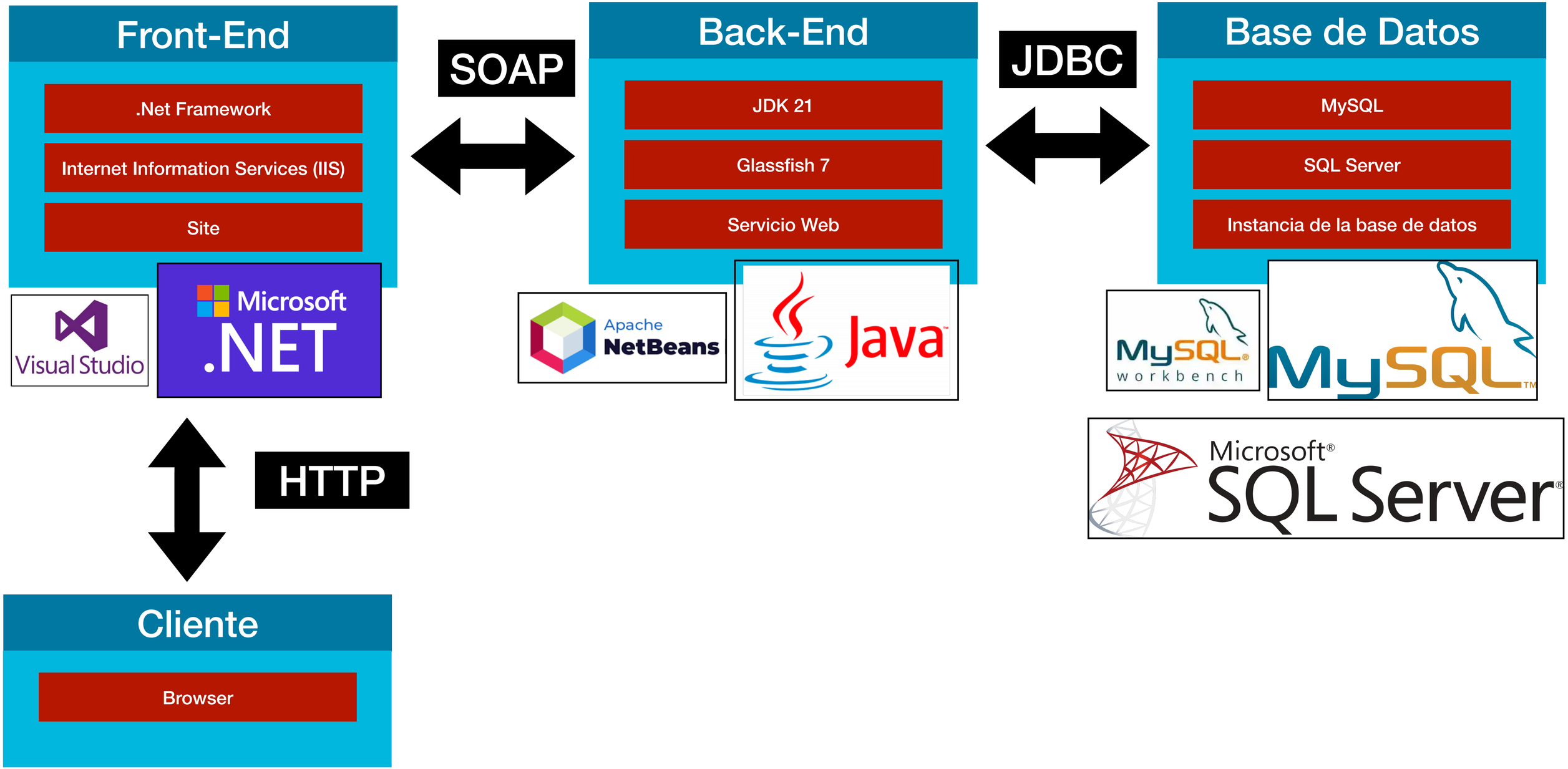

Front-End
.Net Framework
Internet Information Services (IIS)
Site
Visual Studio
Microsoft
.NET
Cliente
Browser

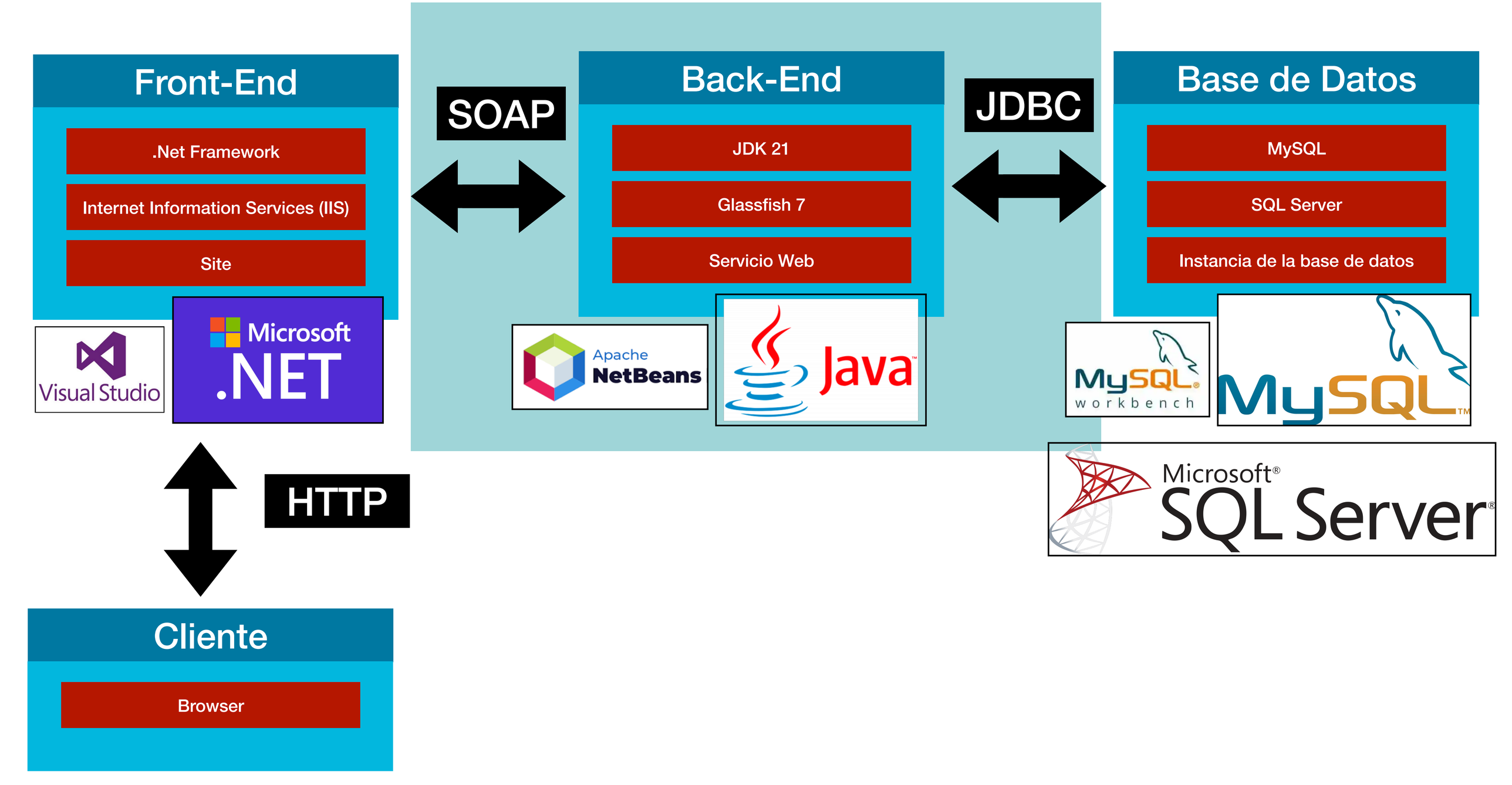

Front-End
.Net Framework
Internet Information Services (IIS)
Site
Visual Studio
Microsoft
.NET
Back-End
JDK 21
Glassfish 7
Servicio Web
Apache
NetBeans
Java
Cliente
Browser


Front-End
.Net Framework
Internet Information Services (IIS)
Site
Visual Studio
Microsoft .NET
Back-End
JDK 21
Glassfish 7
Servicio Web
Apache NetBeans
Java
Base de Datos
MySQL
SQL Server
Instancia de la base de datos
MySQL workbench
MySQL
Microsoft SQL Server
Cliente
Browser


Front-End
.Net Framework
Internet Information Services (IIS)
Site
Visual Studio
Microsoft .NET
Back-End
JDK 21
Glassfish 7
Servicio Web
Apache NetBeans
Java
Base de Datos
MySQL
SQL Server
Instancia de la base de datos
MySQL workbench
MySQL
Microsoft SQL Server
HTTP
Cliente
Browser


Front-End
.Net Framework
Internet Information Services (IIS)
Site
Visual Studio
Microsoft .NET
SOAP
Back-End
JDK 21
Glassfish 7
Servicio Web
Apache NetBeans
Java
Base de Datos
MySQL
SQL Server
Instancia de la base de datos
MySQL workbench
MySQL
Microsoft SQL Server
HTTP
Cliente
Browser


Front-End
.Net Framework
Internet Information Services (IIS)
Site
Visual Studio
Microsoft .NET
SOAP
Back-End
JDK 21
Glassfish 7
Servicio Web
Apache NetBeans
Java
JDBC
Base de Datos
MySQL
SQL Server
Instancia de la base de datos
MySQL workbench
MySQL
Microsoft SQL Server
HTTP
Cliente
Browser


Front-End
.Net Framework
Internet Information Services (IIS)
Site
Visual Studio
Microsoft .NET
SOAP
Back-End
JDK 21
Glassfish 7
Servicio Web
Apache NetBeans
Java
JDBC
Base de Datos
MySQL
SQL Server
Instancia de la base de datos
MySQL workbench
MySQL
Microsoft SQL Server
HTTP
Cliente
Browser