



Página Principal



Mis cursos



Claustros Docente



Salir

/ Área personal / 55/1/1-4-4/9/5744-1 / Arquitectura N-Capas / Evaluación

**Comenzado el** miércoles, 16 de octubre de 2019, 18:16

**Estado** Finalizado

**Finalizado en** miércoles, 16 de octubre de 2019, 19:02

**Tiempo empleado** 46 minutos 25 segundos

**Puntos** 11,67/14,00

**Calificación** 8,33 de 10,00 (83%)

Pregunta **1**

Correcta

Puntúa 1,00  
sobre 1,00

Las principales funcionalidades de una aplicación distribuida están ubicadas en las 3 conocidas capas lógicas, pero existen otras 2 funcionalidades que no son exclusivas de un grupo lógico ya que deben aparecer en todos los elementos de la aplicación. Mencione una de ellas:  ✓

Seguridad y registro de sucesos, considerados servicios transversales

La respuesta correcta es: seguridad

Pregunta **2**

Parcialmente  
correcta

Puntúa 0,67  
sobre 1,00

Independientemente de la arquitectura de las aplicaciones distribuidas, estas comparten características comunes, dentro de ellas podemos mencionar la topología de red. Con respecto a esta característica cuál o cuáles de los siguientes puntos deben ser tomados en cuenta al momento del diseño de la arquitectura de la aplicación?

Seleccione una o más de una:

- ☒ a. Si la aplicación va a ser accedida por Internet ✓
- ☐ b. La distribución o ubicación de los componentes lógicos en las distintas unidades o equipos físicos
- ☐ c. El tráfico de red en relación al ancho de banda disponible y la cantidad de usuarios de la aplicación.
- ☒ d. El tipo de red a manejarse (WAN, LAN) mismo que afectaría al rendimiento de la red ✓

La respuesta correcta es: El tráfico de red en relación al ancho de banda disponible y la cantidad de usuarios de la aplicación., El tipo de red a manejarse (WAN, LAN) mismo que afectaría al rendimiento de la red, Si la aplicación va a ser accedida por Internet

Pregunta **3**

Correcta

Puntúa 1,00  
sobre 1,00

Seleccione la o las funciones que realiza la capa de servidor en una arquitectura Cliente-Servidor

Seleccione una o más de una:

- ☒ a. Procesar los datos recibidos y responder con los resultados obtenidos al usuario en caso de ser necesario ✓
- ☐ b. Procesar los datos que se le envían y mostrarlos al usuario
- ☐ c. Solicitar datos y presentarlo al usuario para que este realice cambios sobre ellos
- ☒ d. Proporcionar los datos que se le solicitan ✓

La respuesta correcta es: Proporcionar los datos que se le solicitan, Procesar los datos recibidos y responder con los resultados obtenidos al usuario en caso de ser necesario

Pregunta **4**

Correcta

Puntúa 1,00  
sobre 1,00

El advenimiento de las computadoras personales, constituyó el inicio de las aplicaciones distribuidas, puesto que ahora los usuarios disponían de un procesador y de un sistema de almacenamiento autónomo.

Seleccione una:

- ☐ Verdadero
- ☒ Falso ✓

La respuesta correcta es 'Falso'

Pregunta **5**

Correcta

Puntúa 1,00  
sobre 1,00

Las principales funcionalidades de una aplicación distribuida están ubicadas en las 3 conocidas capas lógicas, pero existen otras 2 funcionalidades que no son exclusivas de un grupo lógico ya que deben aparecer en todos los elementos de la aplicación. Mencione una de ellas:  ✓

La respuesta correcta es: seguridad

Pregunta **6**

Finalizado

Puntúa 1,00  
sobre 2,00

Diseñar el diagrama de clases para una aplicación que permita llevar brindar el servicio de parqueadero. Considere el patrón de arquitectura distribuida. A nivel de funcionalidades, considere únicamente registrar el ingreso de un vehículo, cobrar el servicio a lo que se vaya a retirar el vehículo y presentar los tickets pagados del día

Se tiene que distinguir cada uno de los componentes lógicos que sean necesarios.

*\*\* Diseña el diagrama en word , visio, paint o cualquier herramienta y adjuntar como una documento PDF.*



[JoseAtariguano.pdf](#)

Comentario:

En ON los métodos no dan respuesta a las funcionalidades solicitadas

Pregunta **7**

Correcta

Puntúa 1,00  
sobre 1,00

¿Los objetos de acceso a datos utilizan los objetos de negocio y las entidades de negocio, siendo una especie de cliente de la lógica de acceso a datos?

Seleccione una:

- ☐ Verdadero
- ☒ Falso ✓

La respuesta correcta es 'Falso'

Pregunta **8**

Correcta

Puntúa 1,00  
sobre 1,00

Los objetos de negocio así como las fachadas de negocio son una abstracción de un conjunto de entidades de datos (objetos de acceso a datos) relacionadas entre si.

Seleccione una:

- ☐ Verdadero
- ☒ Falso ✓

La respuesta correcta es 'Falso'



Pregunta **9**

Incorrecta

Puntúa 0,00  
sobre 1,00

Ejemplos de objetos de acceso a servicios podrían ser:

Seleccione una o más de una:

- ☐ a. Establecimiento de la conexión con el servidor
- ☐ b. Consulta de datos del SGBD
- ☒ c. Servidor de Red 
- ☐ d. Envío de un correo electrónico
- ☒ e. Consulta de la hora actual del sistema 
- ☐ f. Consulta de los recursos del equipo servidor

La respuesta correcta es: Envío de un correo electrónico, Consulta de la hora actual del sistema, Consulta de los recursos del equipo servidor

Pregunta **10**


Correcta

Puntúa 1,00  
sobre 1,00

Una aplicación distribuida en n-capas, distingue los distintos elementos que la conforman en tres grupos lógicos:

- La capa del servidor: que incluye los diferentes componentes que se encargarán de recibir las peticiones de datos, procesarlas y suministrar la información solicitada, así también las peticiones de acceso a servicios básicos del sistema.
- La capa de negocios: capa que encapsula las reglas de acceso a datos a través de los objetos de negocio, objetos que adicionalmente posibilitan la gestión de procesos internos de la aplicación, así como la lógica necesaria para interactuar con el usuario.
- La capa de interfaz de usuario, constituida por el software con la que el usuario interactúa para operar la aplicación.

Seleccione una:

- ☐ Verdadero
- ☒ Falso 

La respuesta correcta es 'Falso'

Pregunta **11**

Correcta

Puntúa 1,00  
sobre 1,00

Complete:

El siguiente concepto corresponde a:



"Componente/capa de la arquitectura de las aplicaciones distribuidas que Incluye aquellos elementos que se encargan de recibir las peticiones de datos o de acceso a servicios básicos del sistema y de suministrar a otros elementos la información solicitada."

La respuesta correcta es: capa de servidor

Pregunta **12**

Correcta

Puntúa 1,00  
sobre 1,00

En una aplicación distribuida en n-capas los procesos están distribuidos en diferentes capas no solo lógicas, sino también físicas. Esta distribución física posibilita:

Seleccione una:

- ☒ a. Que los procesos se ejecuten en diferentes equipos, que puedan incluso residir en plataformas o sistemas operativos distintos. ✓
- ☐ b. Que los procesos se ejecuten en equipos diferentes, con una correspondencia única entre una capa lógica y un equipo físico.
- ☐ c. Ninguna de las opciones son correctas
- ☐ d. Que los procesos se ejecuten en diferentes equipos, siempre que sean en plataformas homogéneas o sistemas operativos similares, condición que posibilita que tanto los recursos como la eficiencia global del sistema se optimicen.

La respuesta correcta es: Que los procesos se ejecuten en diferentes equipos, que puedan incluso residir en plataformas o sistemas operativos distintos.

Pregunta **13**

Correcta

Puntúa 1,00  
sobre 1,00

Independientemente de la arquitectura de las aplicaciones distribuidas, estas comparten características comunes, dentro de ellas podemos mencionar la Ubicación de la Lógica, misma que se refiere a cuál de los procesos físicos se sitúa en cada componente lógico. Dicha ubicación impacta sobre:

- 1. Rendimiento de la aplicación
- 2. Facilidad de programación
- 3. Distribución física de las capas (n capas lógicas, n capas físicas)
- 4. Reutilización de código
- 5. Homogenidad de la aplicación

Seleccione la combinación correcta.

Seleccione una:

- ☐ a. 1, 2, 3, 4
- ☐ b. 1, 2, 3, 4, 5
- ☐ c. 1, 3, 4
- ☒ d. 1, 2, 4 ✓
- ☐ e. 3, 5

La respuesta correcta es: 1, 2, 4

© 2016 Universidad Politécnica Salesiana

Development by UNADEDVI

Universidad Politécnica Salesiana  
Unidad Académica de Educación a Distancia y Virtual - UNADEDVI  
Dirección: Calle Vieja y Elia Liut  
Teléfono: (+593) 72862213 ext.: 1700  
Correo electrónico: unadedvi@ups.edu.ec



# APLICACIONES DISTRIBUIDAS

Ing. Cristian Timbi

# Introducción al Diseño de Aplicaciones Distribuidas

Introducción

Características de las Apps Distribuidas

Tipos de aplicaciones Distribuidas

Ejemplos

Arquitectura de las Aplicaciones Distribuidas

Capa de Servidor

Capa de negocios

Capa de presentación

Distribución física de las capas lógicas



## Generaciones de Computadores

- 1º Generación (Tubos al vacío)
- 2º Generación (Transistores)
- 3º Generación (Circ. Integrados)
- 4º Generación (Microprocesadores)
- 
- 5º Generación (Intelegencia Artificial)
- 6º Generación (Compu. Quántica)



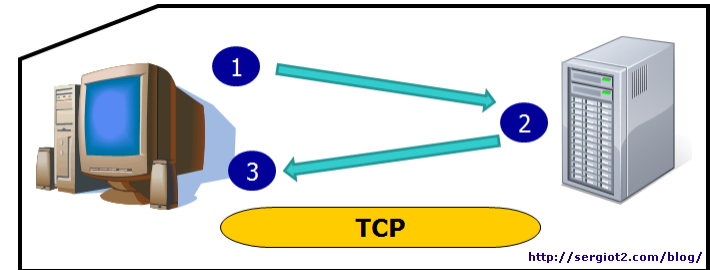
EVOLUCIÓN HISTÓRICA



Sis. Centrales



Apps Escritorio



Apps Distribuidas

EVOLUCIÓN

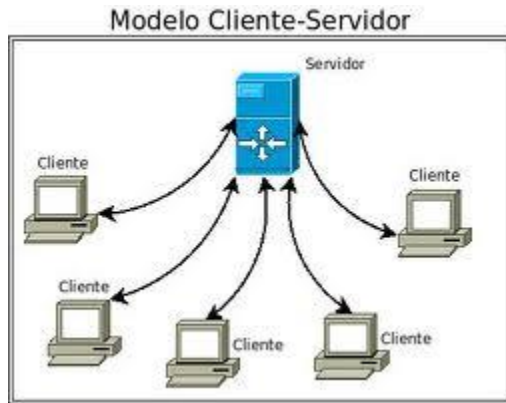
**"App. distribuida es aquella cuyo objetivo final se alcanza mediante la ejecución de diversos procesos independientes, que por lo general se ejecutan en equipos diferentes y que de una u otra forma se pasan datos entre ellos mediante protocolos de comunicación bien establecidos"**

## DEFINICIÓN

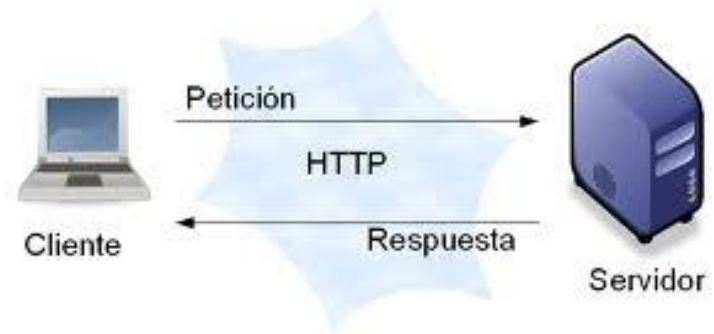
- Concurrencia
- Topología de Red
- Ubicación lógica
- Homogenidad de la plataforma
- Seguridad

## CARACTERISTICAS

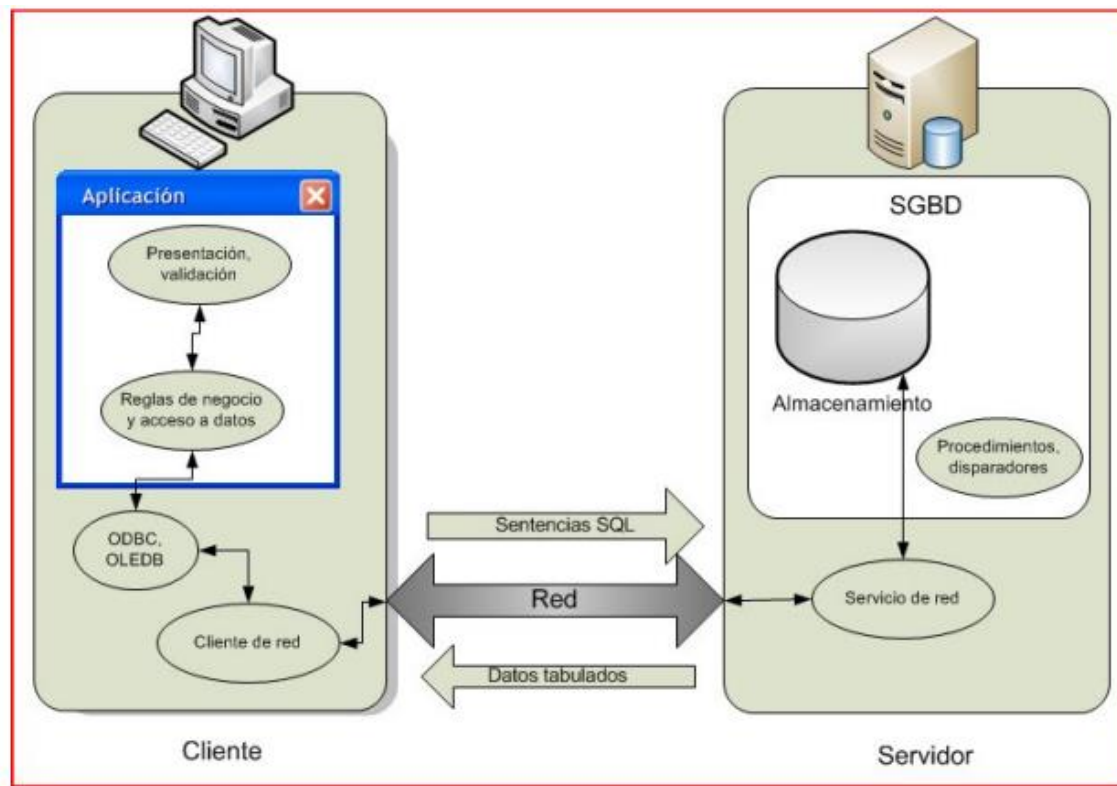
## Cliente Servidor



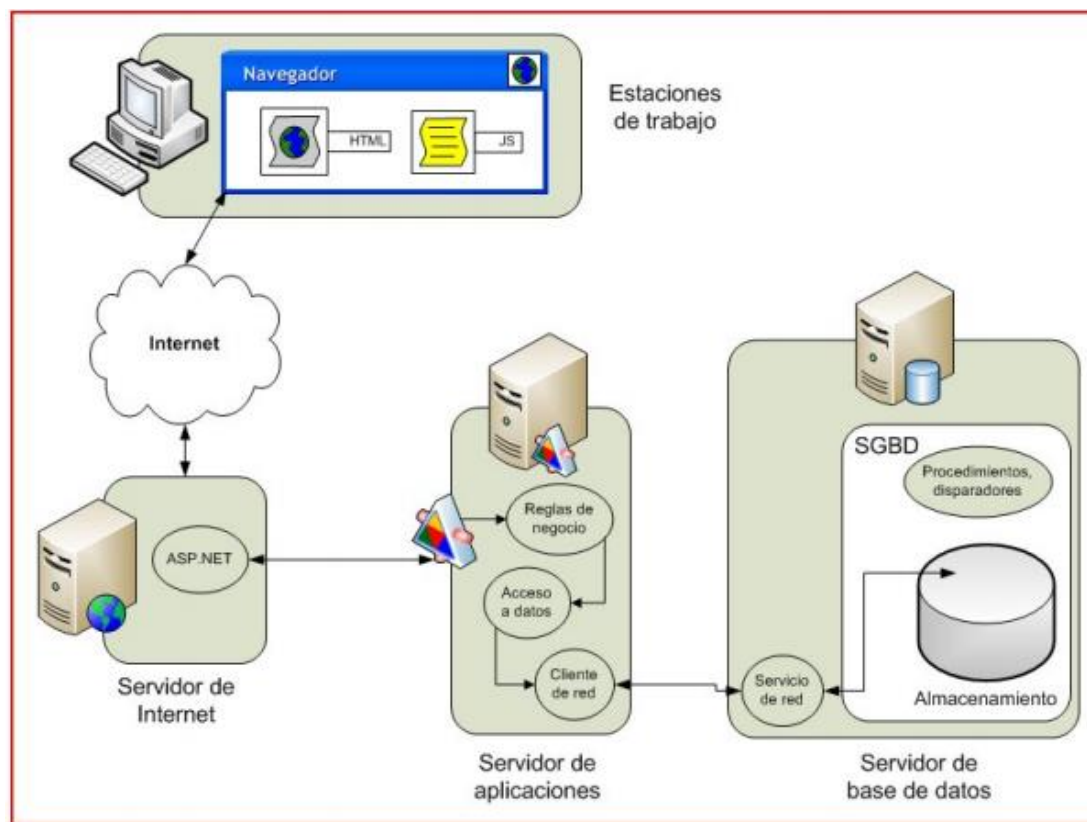
## N - Capas



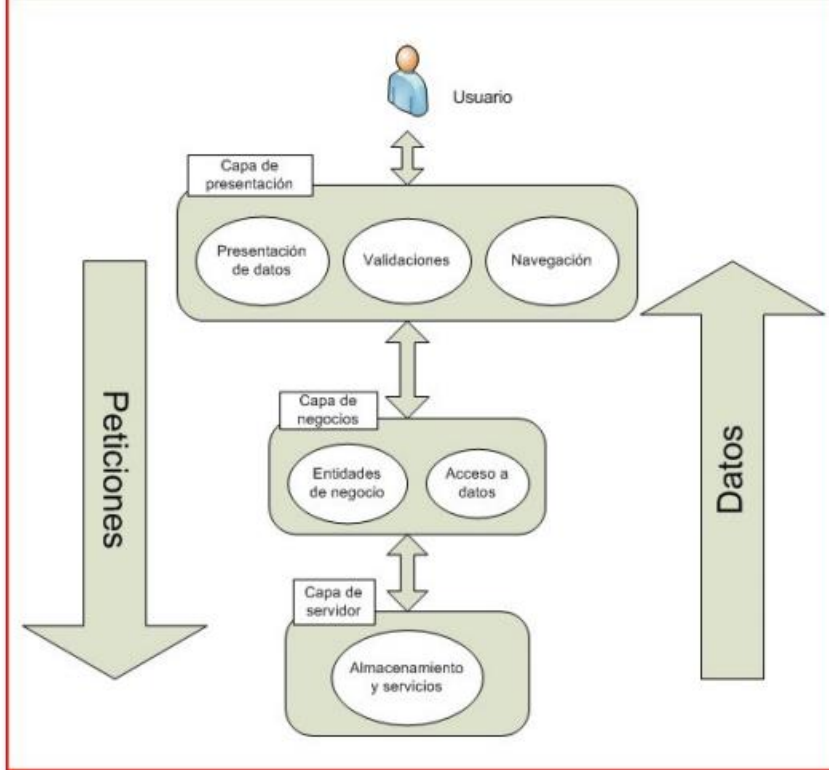
## TIPOS DE APPS DISTRIBUIDAS



## CLIENTE - SERVIDOR



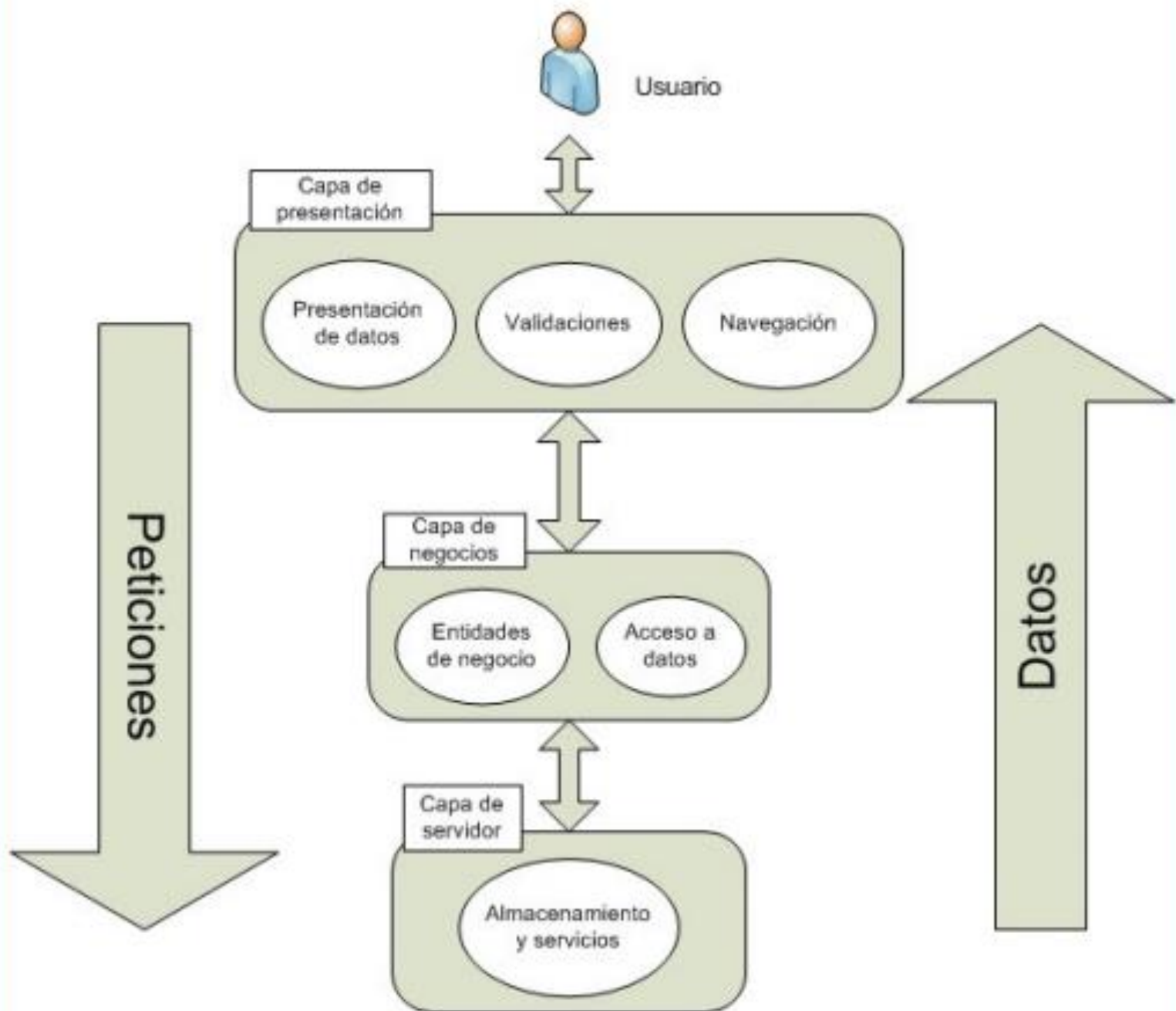
N-CAPAS



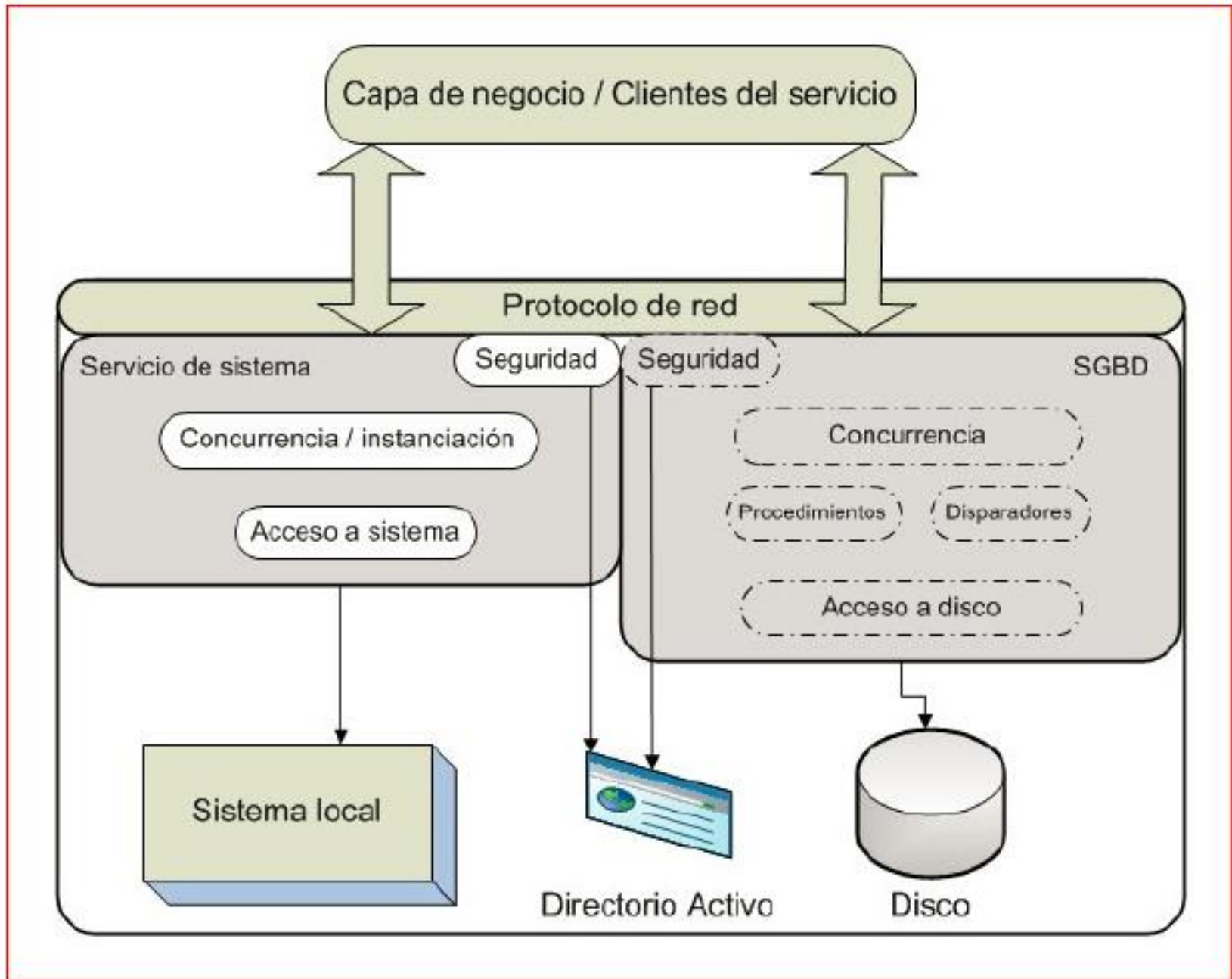
- Capa de Servidor
- Capa de Negocios
- Capa de Presentación

ARQUITECTURA DE LAS APPS DIS.

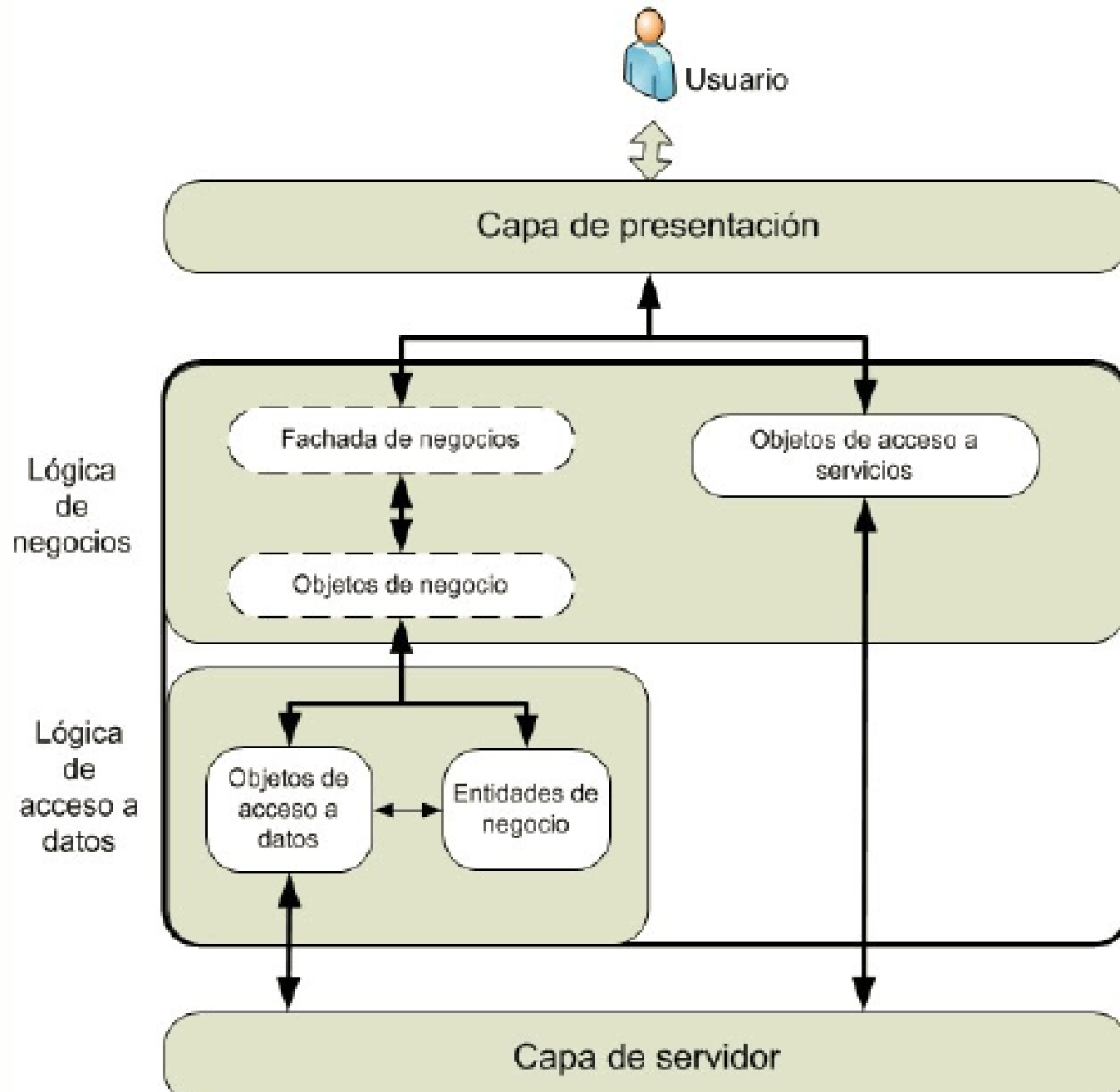




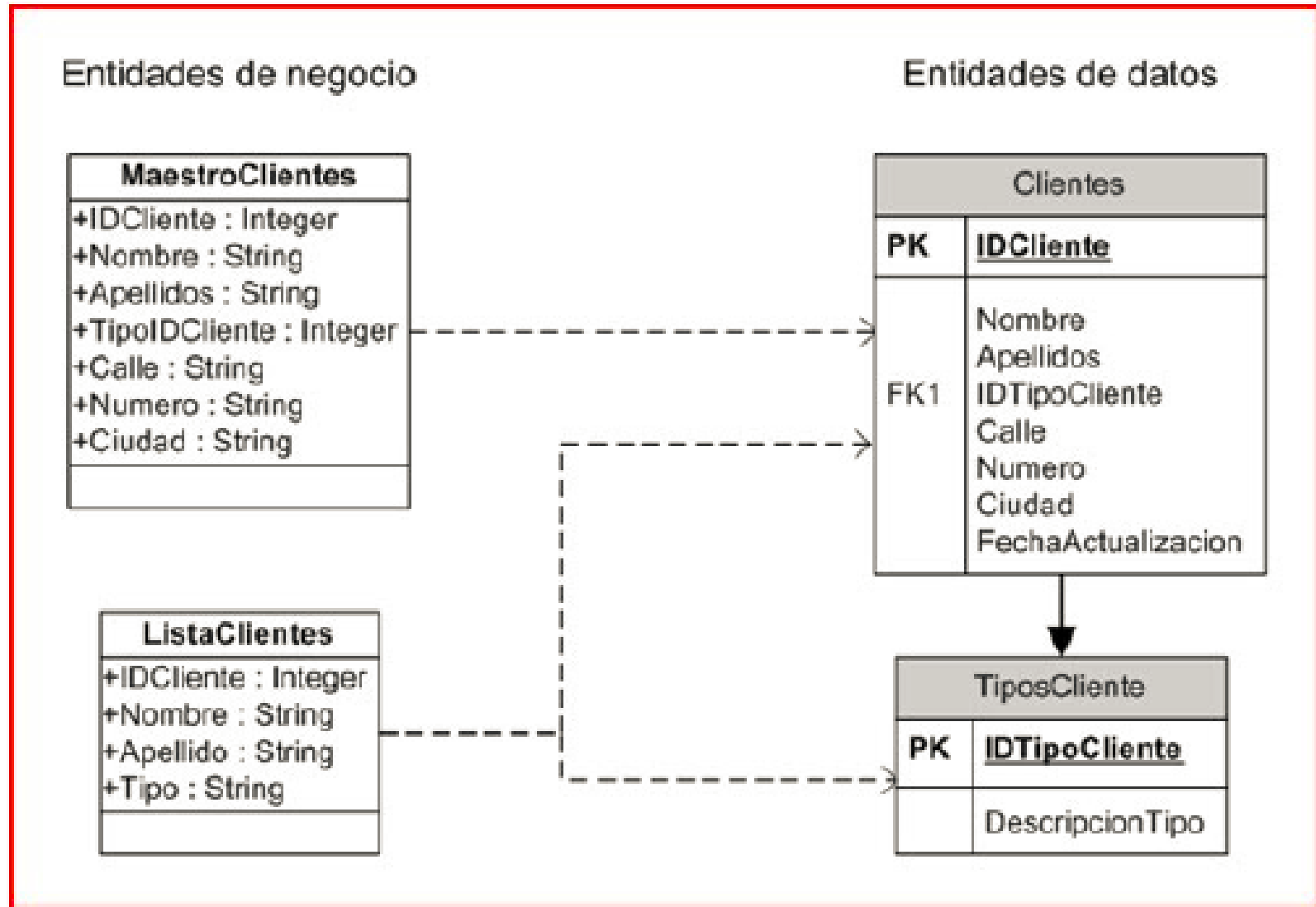
## CAPA DE SERVIDOR



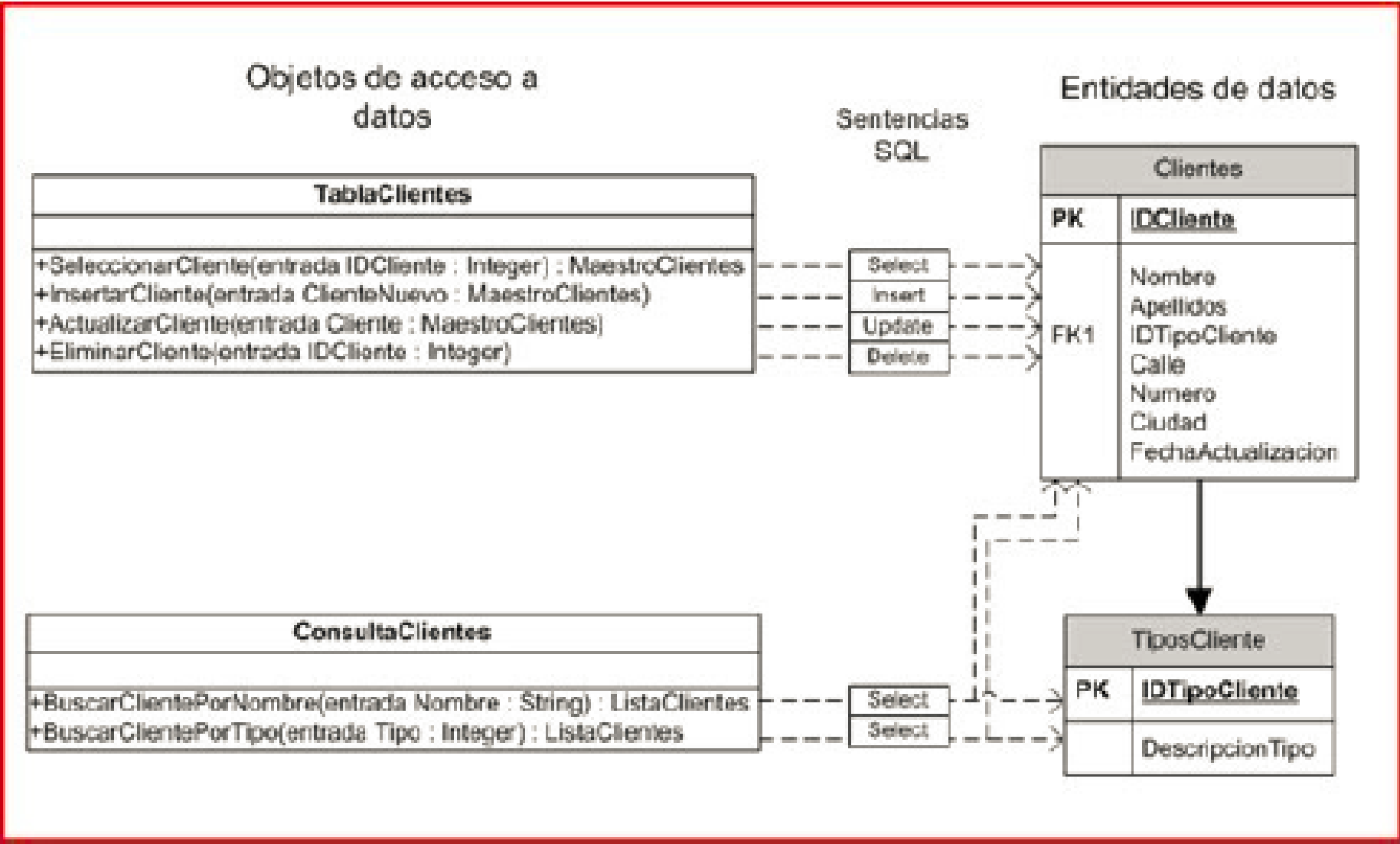
# CAPA DE NEGOCIO



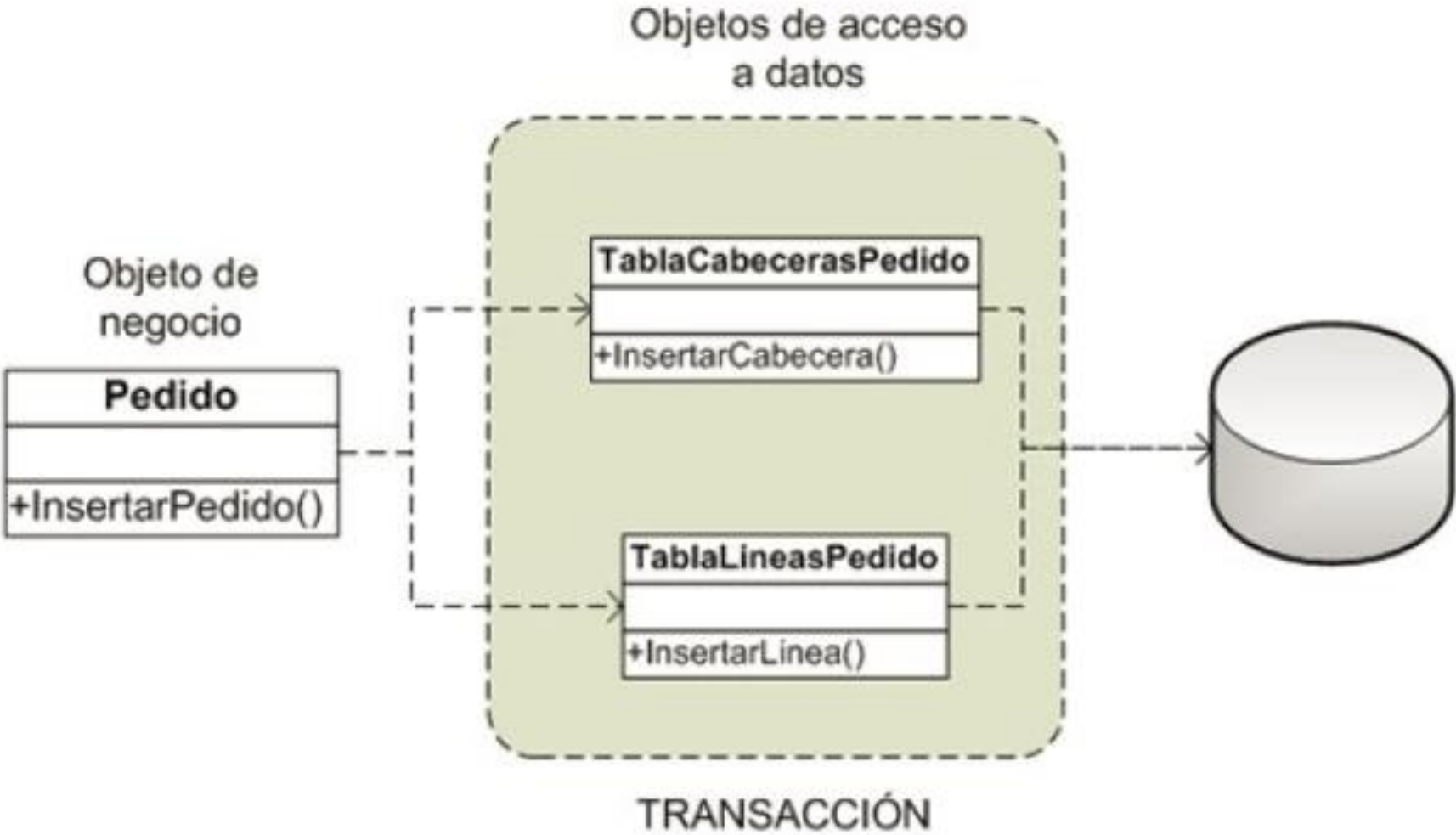
# ENTIDADES DE NEGOCIO



# OBJETOS DE ACCESO A DATOS - ENTIDADES



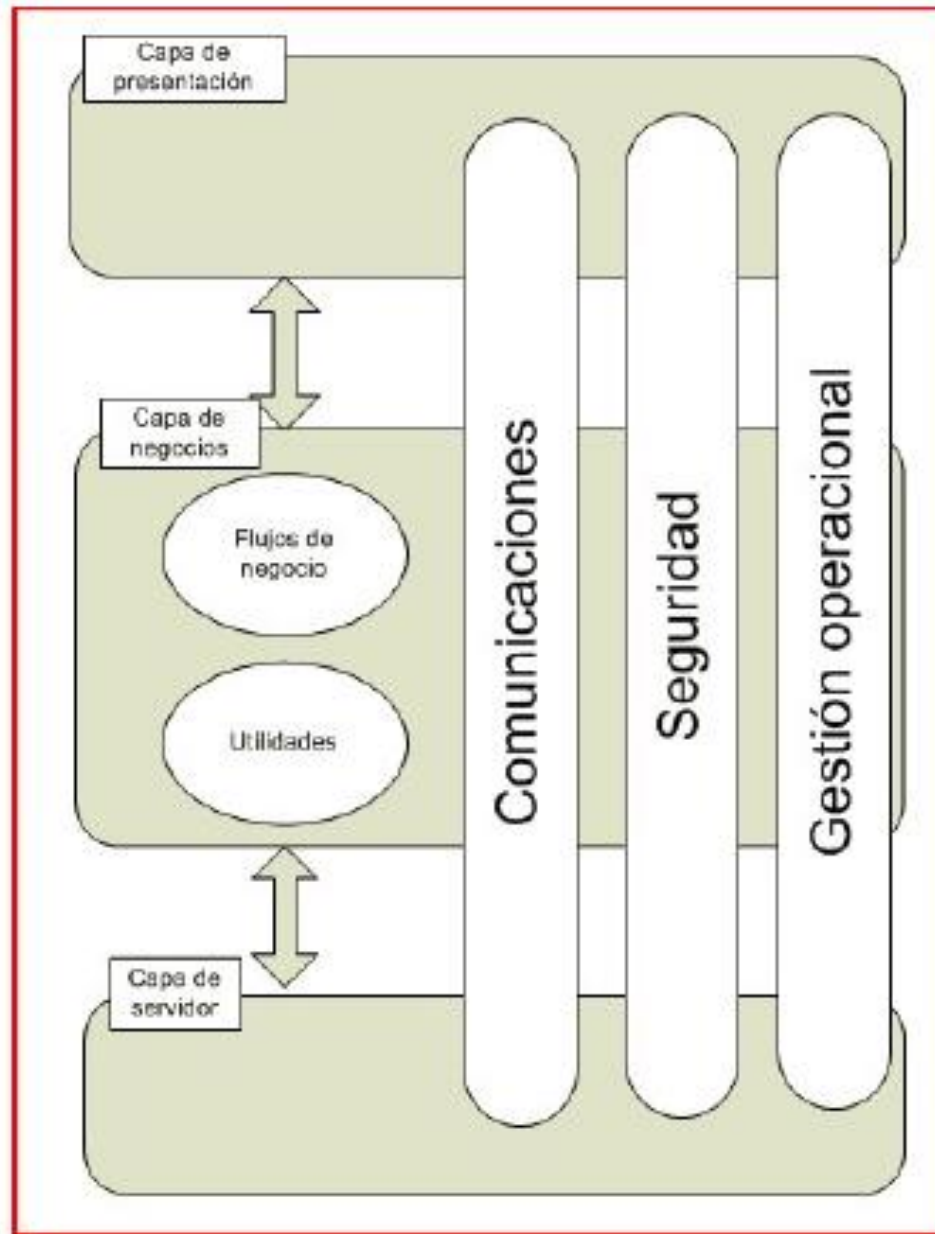
OBJETO DE NEGOCIO - OBJETOS DE ACCESO A DATOS



# FACHADA DE NEGOCIO

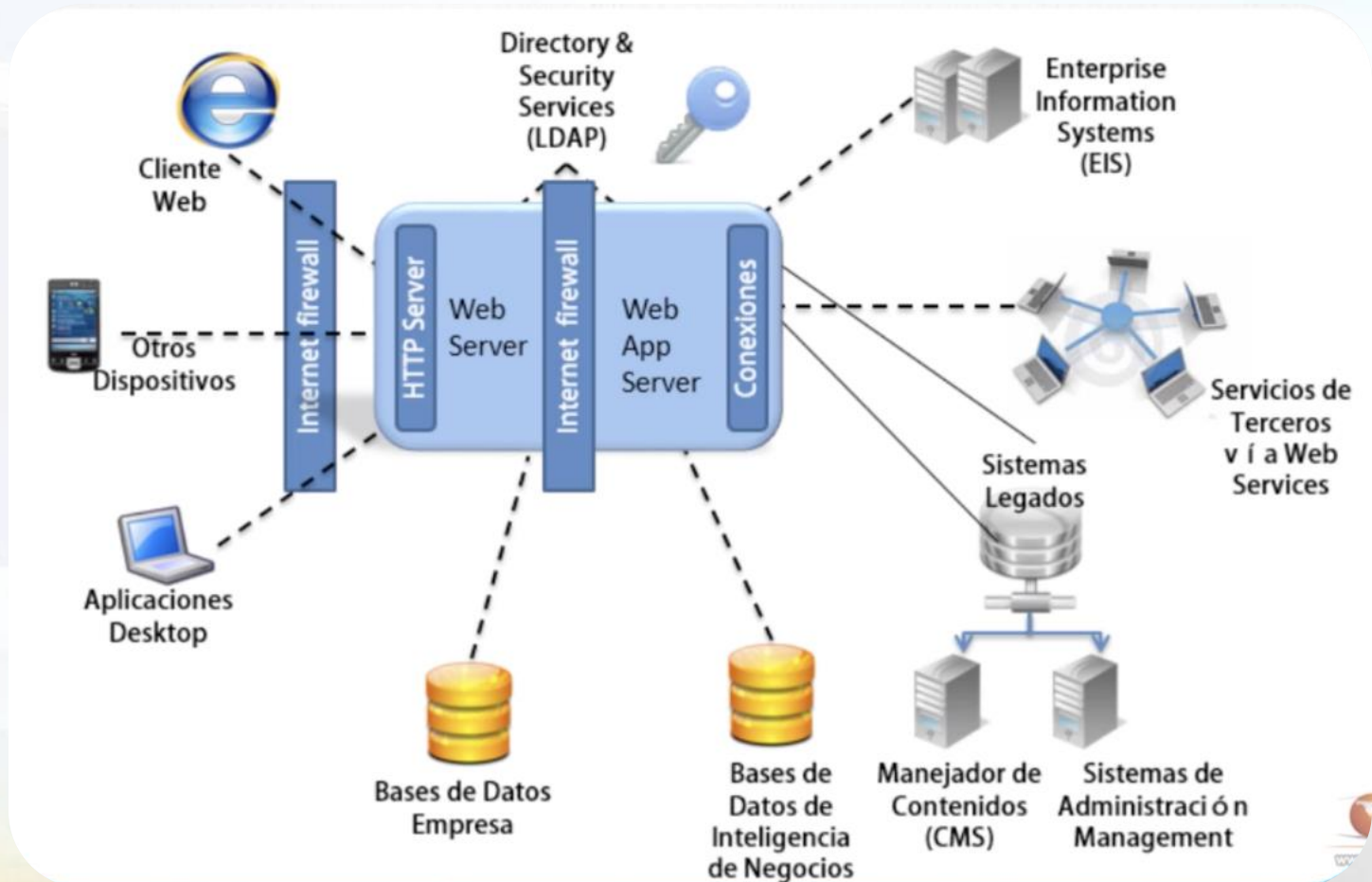


## OTROS OBJETOS





# JEE Introducción



# Las aplicaciones empresariales

- **Proporcionan la lógica del negocio para una empresa.**
- **Se gestionan de forma centralizada**
- **Posibilitan interactuar con otras aplicaciones de la propia empresa o de terceros.**

*“En el mundo de la tecnología de la información, las aplicaciones empresariales deben estar diseñados, contruidos y producidos por menos dinero, con mayor velocidad, y con menos recursos.”*

Alternativa → JEE

- Conjunto de APIs que permite:
  - Acortar los tiempos de desarrollo.
  - Reducción de la complejidad de las aplicaciones
  - Mejor del rendimiento y seguridad de las aplicaciones

# Modelo de aplicación JEE

El modelo de aplicación Java EE define una arquitectura para la implementación de servicios como aplicaciones de varios niveles que ofrecen escalabilidad, accesibilidad y capacidad de gestión, necesaria para aplicaciones de nivel empresarial.

Dividiendo el trabajo necesario en dos partes:

- La lógica de negocio y la presentación → a ser implementado por el desarrollador
- Los servicios estándares del sistema → implementados por la plataforma Java EE (servidores de aplicaciones)

# Java JEE

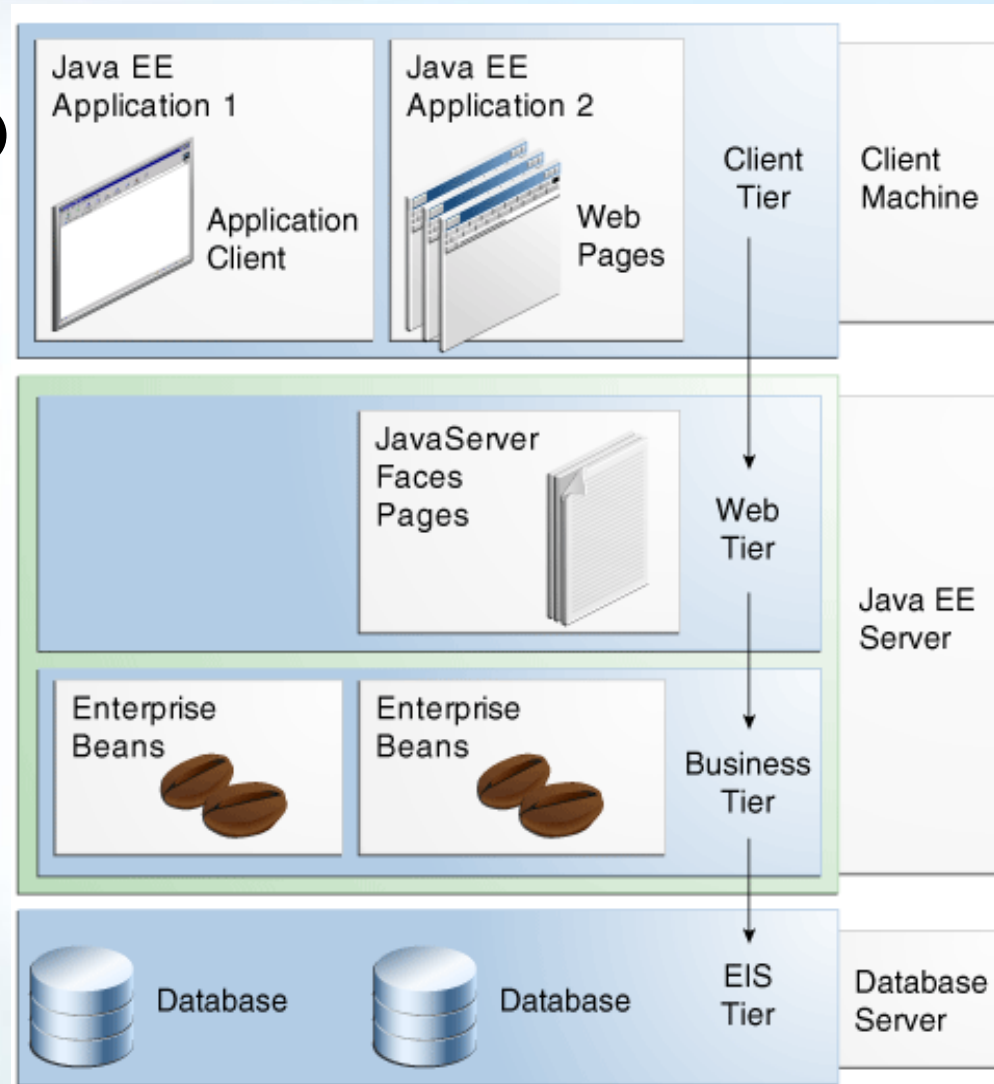
- Utiliza un modelo de programación simplificado
- En lugar de Descriptores de despliegue XML , simplemente se introducen **anotaciones** directamente en un archivo de código fuente de Java, y el servidor Java EE configurará el componente en la implementación y ejecución.
  - **Ejemplo:**
    - **Persistencia y Constraint de BD**
    - **EJB (Enterprise Java Beans)**
    - **WebService**

```
@Stateless
@Named
@Path("/status")
public class StatusBean {
    ...
    @GET
    @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    @Path("/{eventId}/")
    public Event getEvent(@PathParam("eventId") Long eventId) {
        ...
    }
}
```

# Arquitectura Distribuida (Multinivel)

Distribuidos en:

- Componentes de cliente de nivel se ejecutan en la máquina cliente.
- Componentes de nivel Web se ejecutan en el servidor Java EE.
- Componentes de la capa de negocio se ejecutan en el servidor Java EE.
- Sistema de información empresarial (EIS) -tier software se ejecuta en el servidor EIS.



# Características

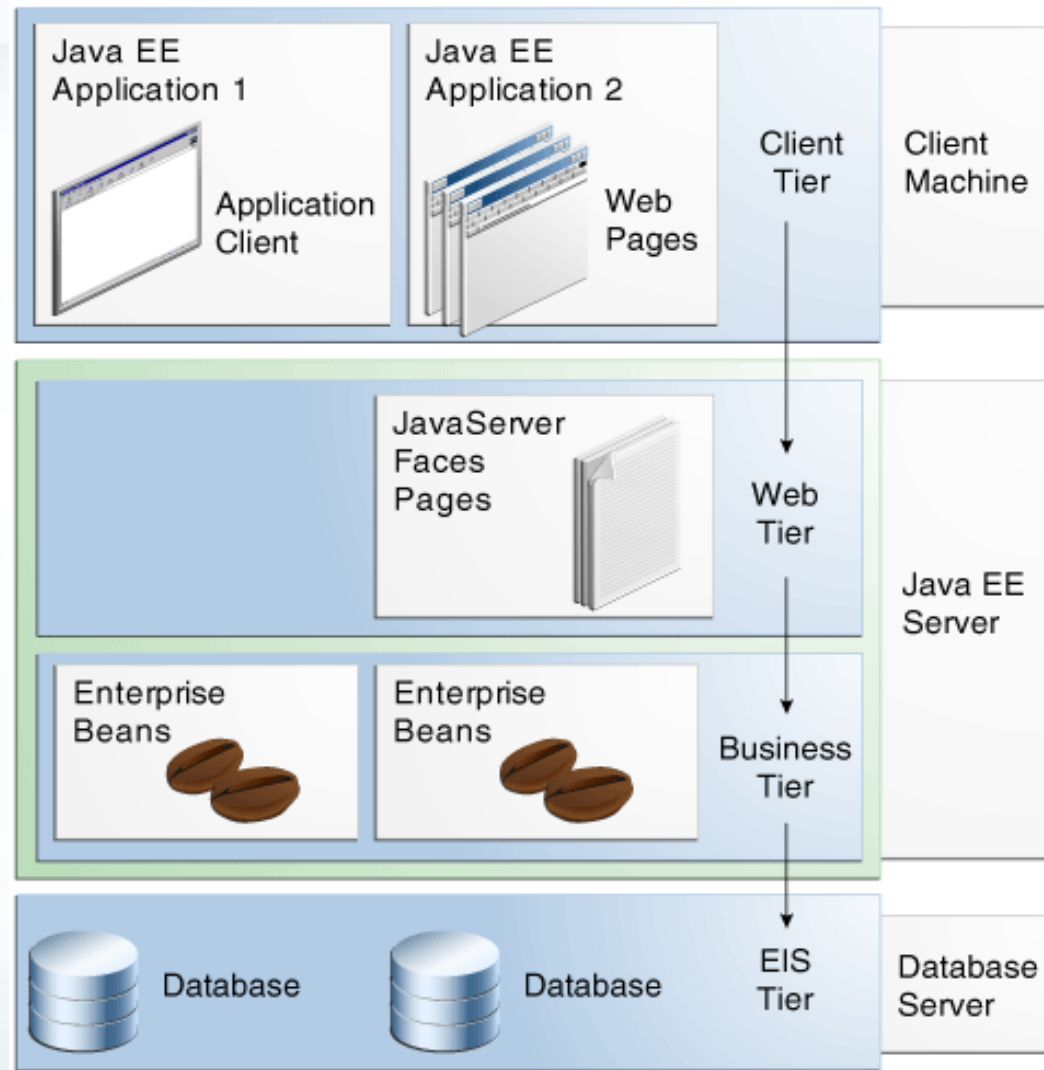
## Seguridad

### Componentes JEE

- Aplicaciones cliente y applets
- Servlets, JSF, JSP
- EJB

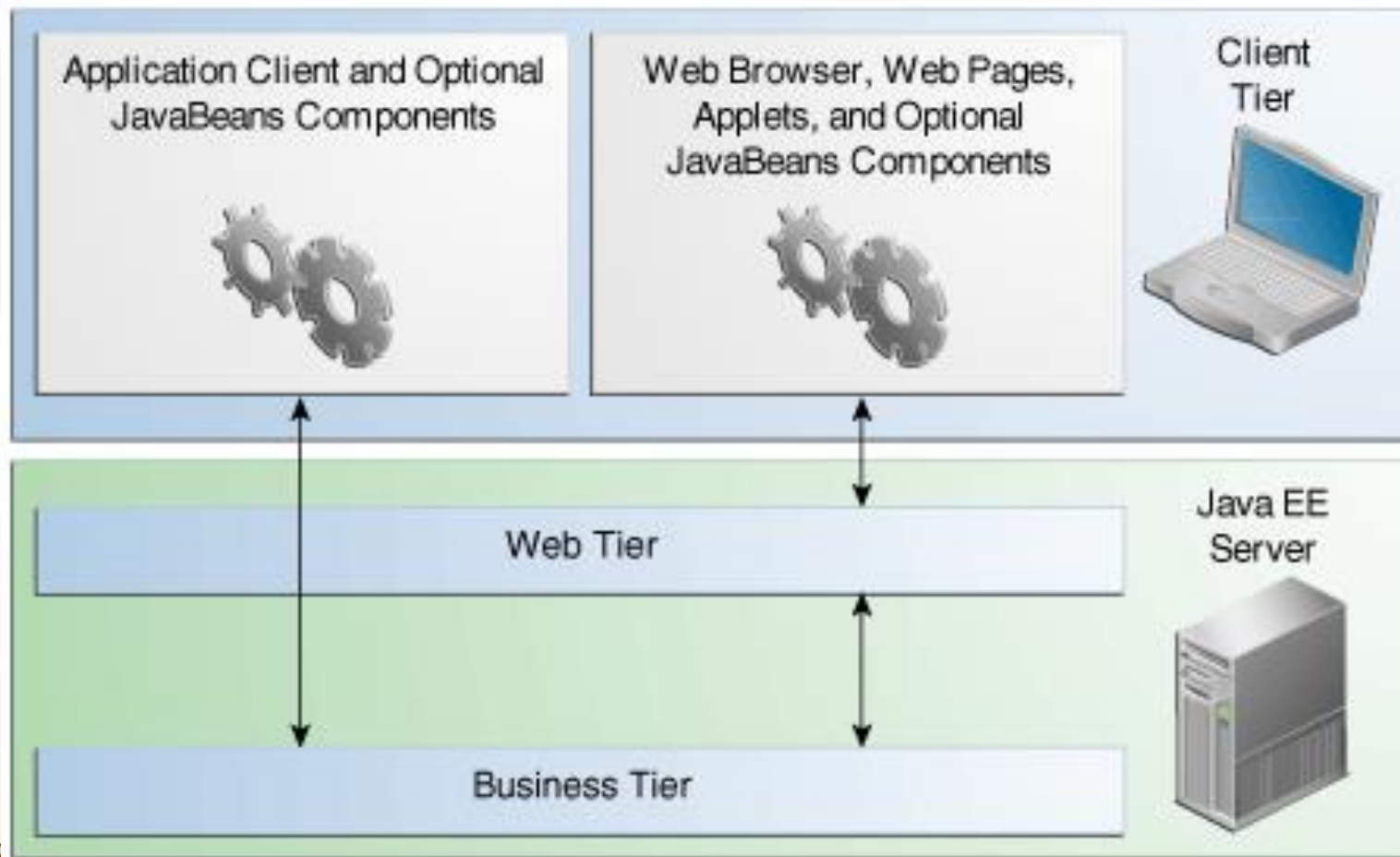
### Cientes JEE

- Clientes web
- Aplicaciones Cliente (standalone)
- Swing
- JavaBeans



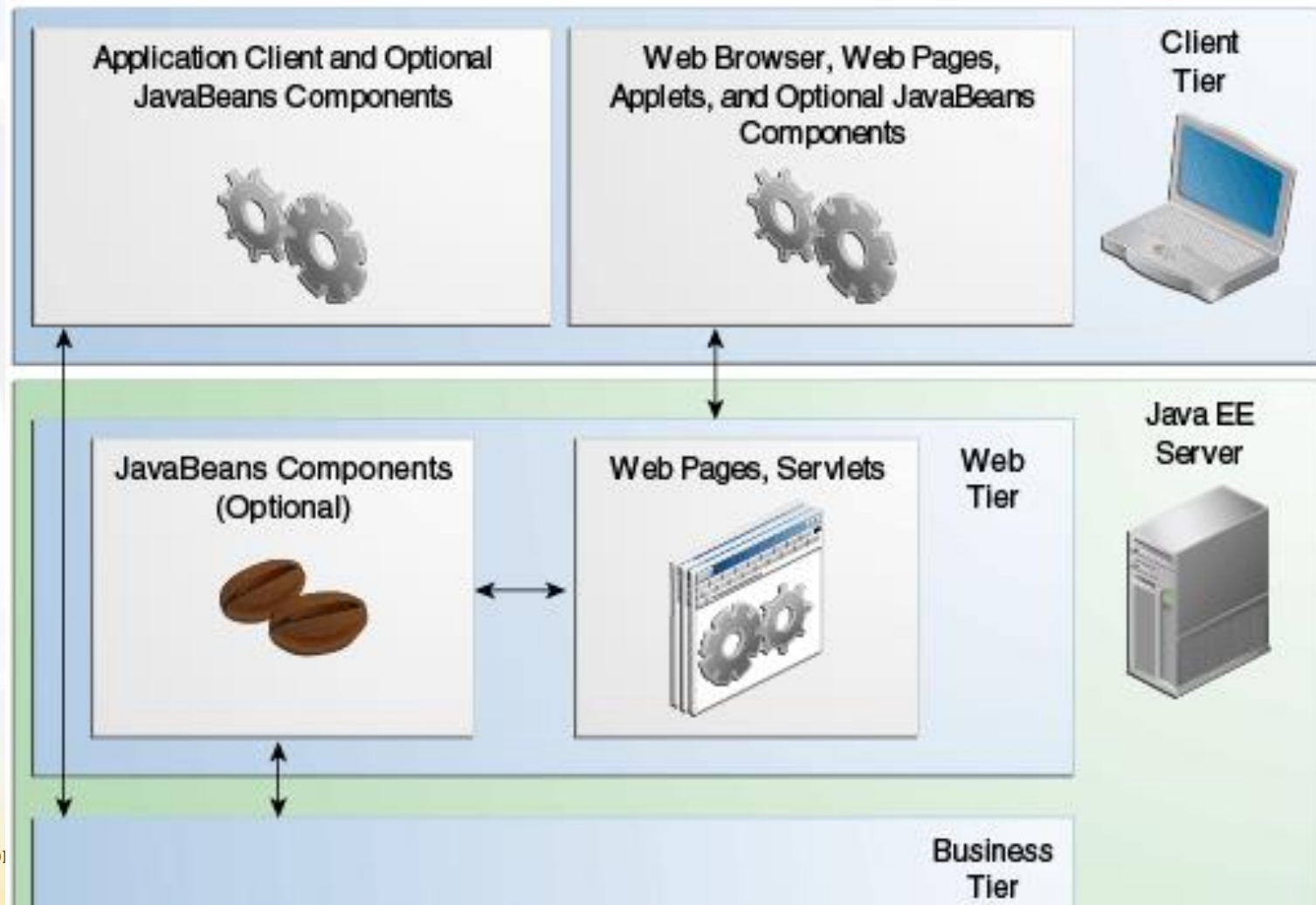
/2020

# Servidor de comunicaciones JEE



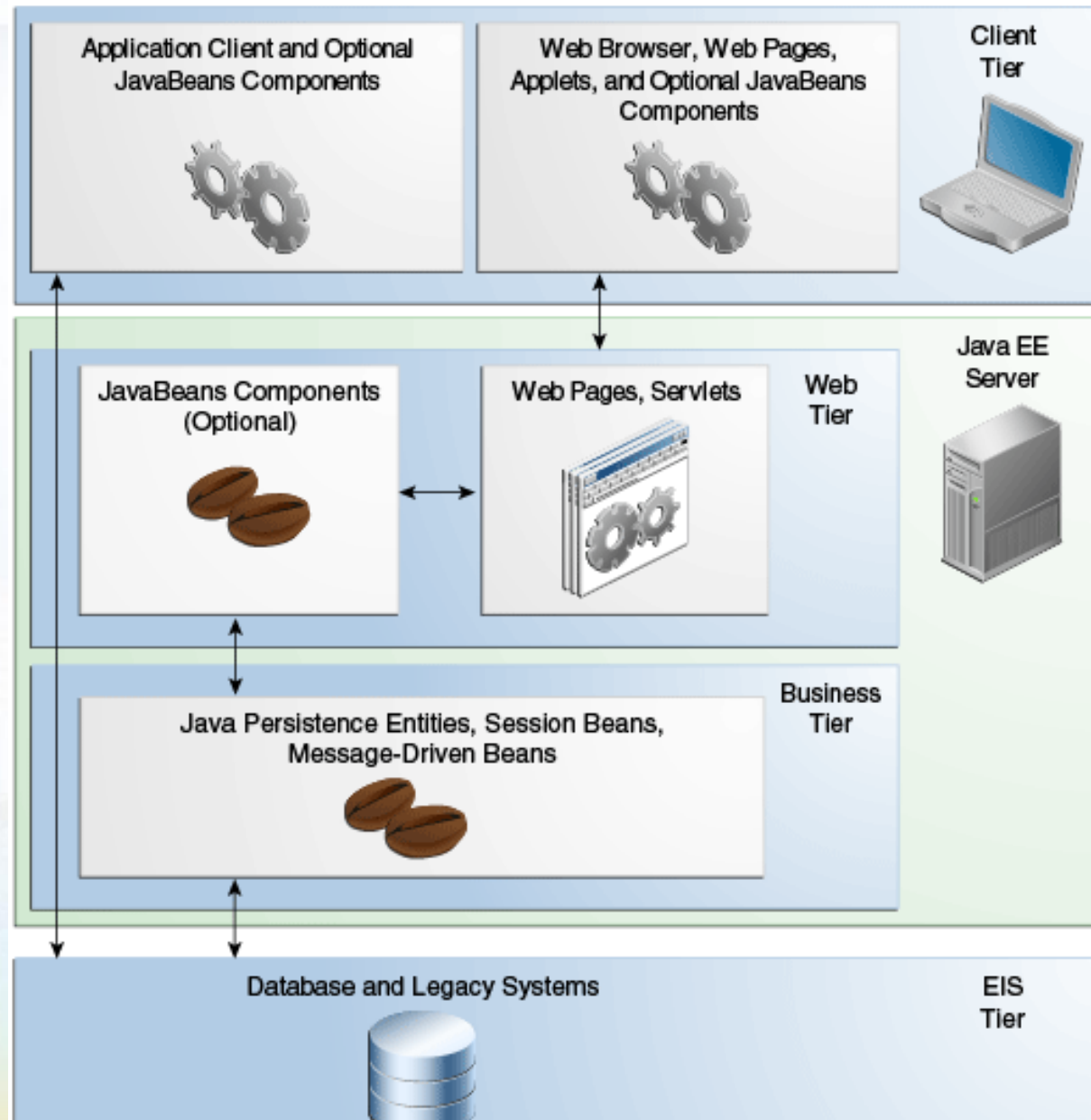


# Componentes Web





# Componentes de Negocio



# Contenedores JEE

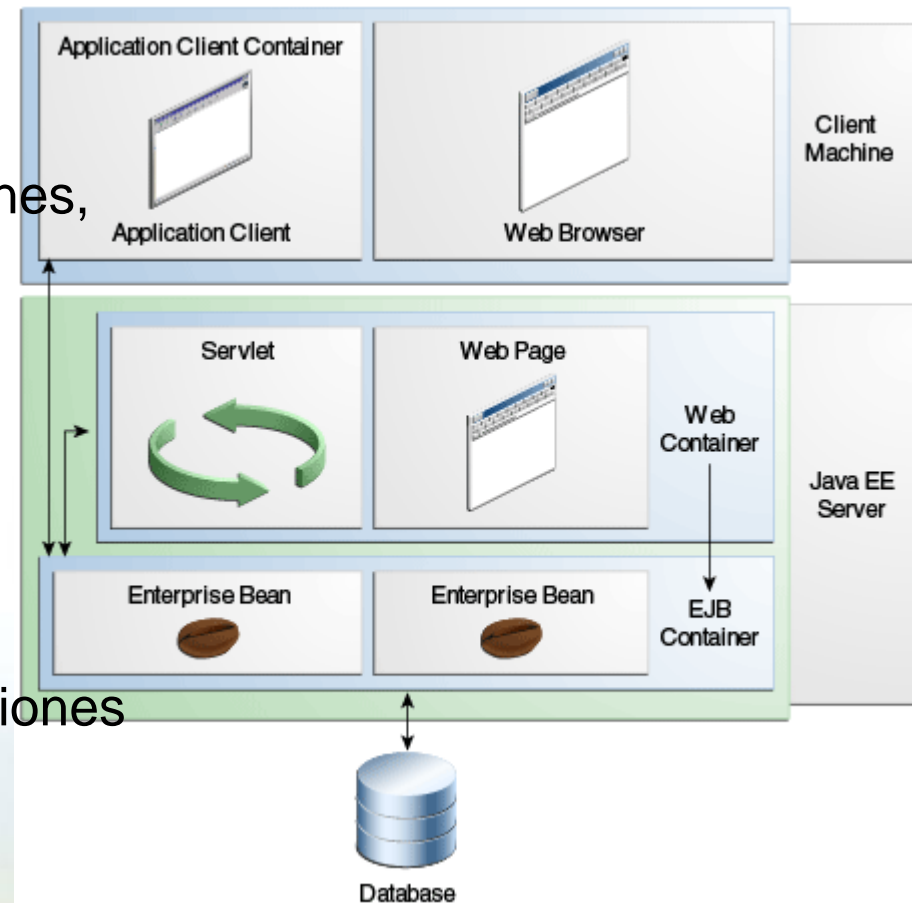
**Contenedores** son la interfaz entre un componente y el de bajo nivel.

## Servicios

- Modelo de seguridad
- Modelo de gestión de transacciones,
- Modelo de búsquedas JNDI
- Modelo de conectividad remota

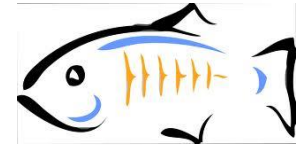
## Tipos

- Servidor Java EE
- Contenedor EJB
- Contenedor Web
- Contenedor de cliente de aplicaciones
- Contenedor Applet



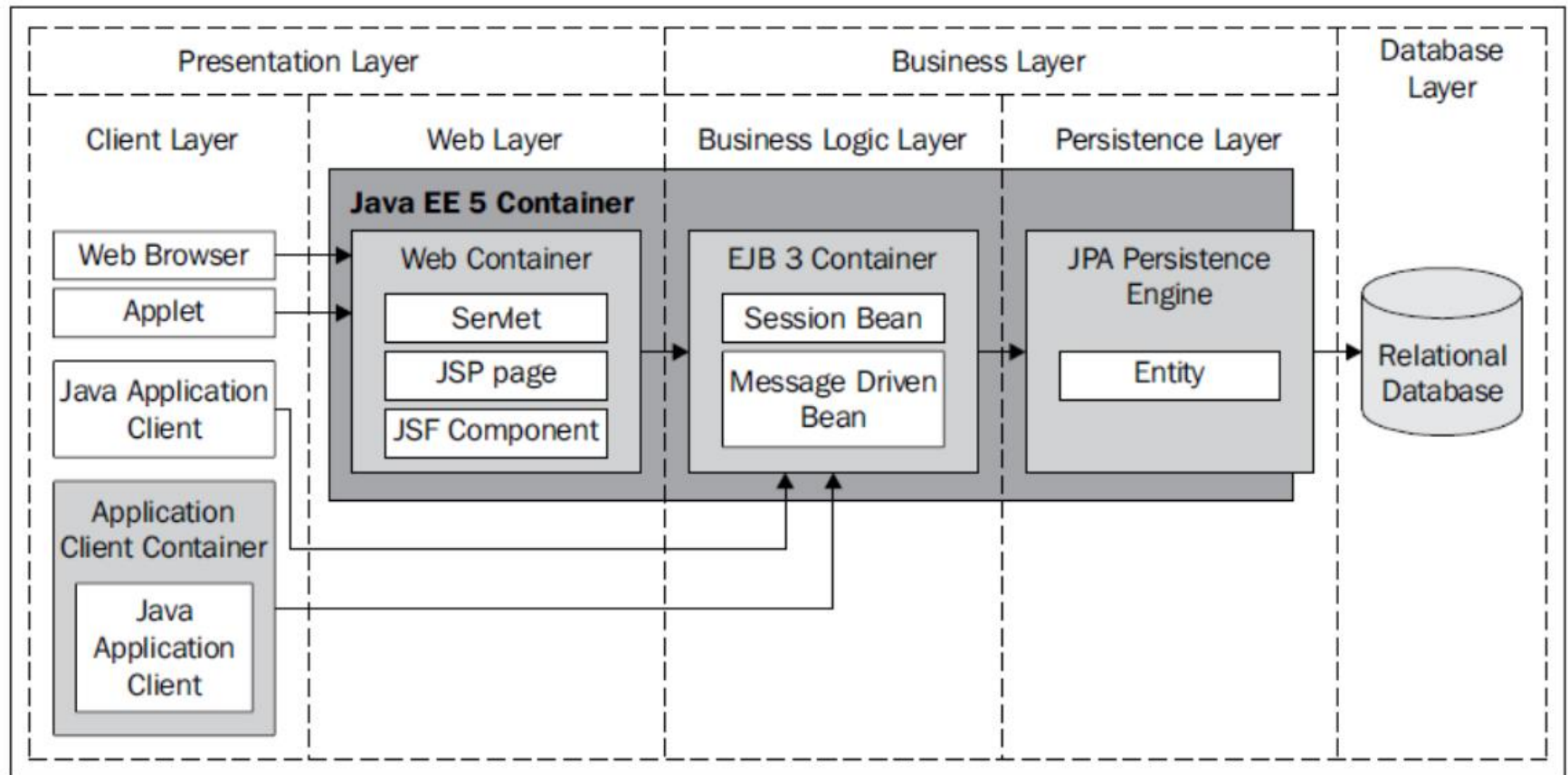
# Java EE APIs

- Enterprise JavaBeans (EJB).
- Java Servlet
- JavaServer Pages (JSP)
- JavaServer Pages Standard Tag Library
- JavaServer Faces
- Java Persistence API
- Java Message Service (JMS).
- Java Transaction API (JTA).
- Java API for RESTful Web Services
- Managed Beans
- Contexts and Dependency Injection for Java EE
- Dependency Injection for Java
- Bean Validation
- Java Message Service API
- JavaMail API

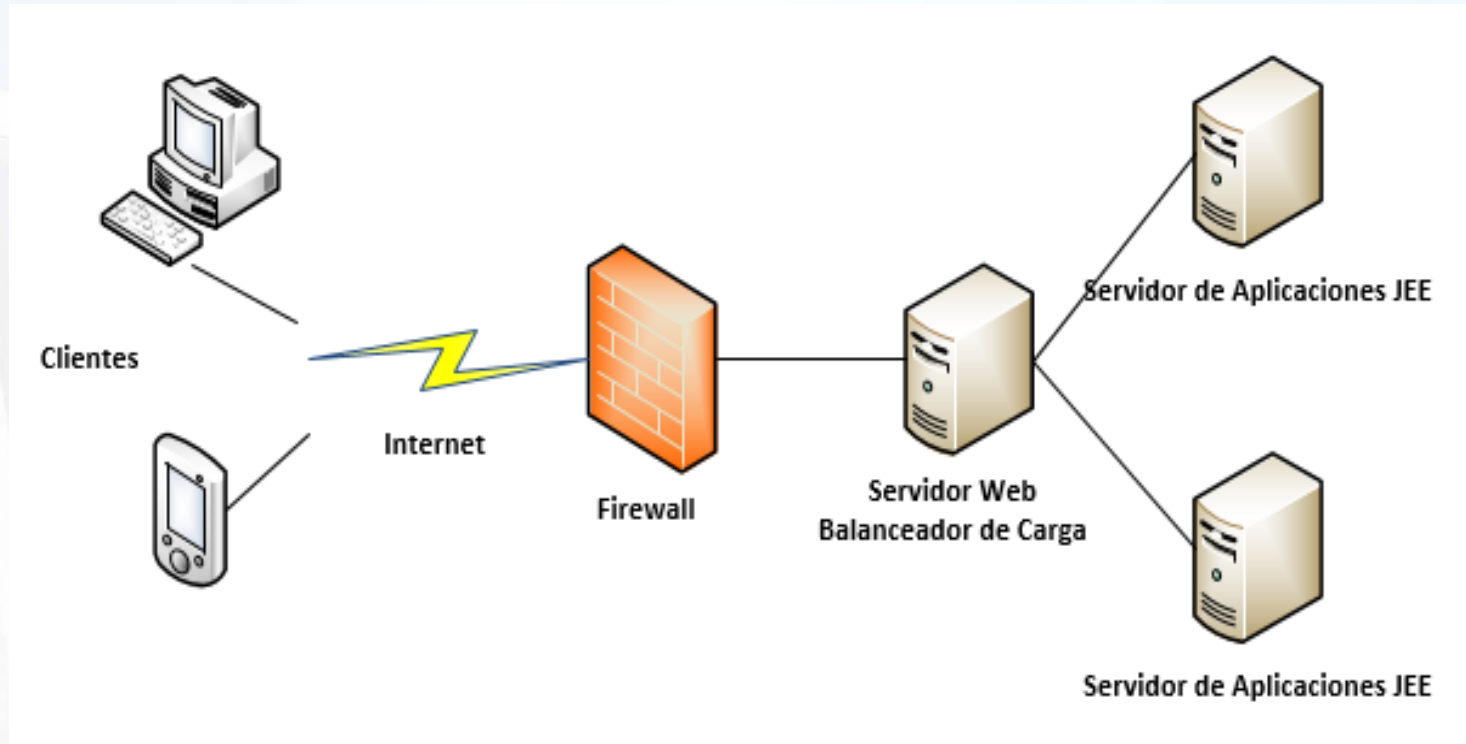


# Antecedentes

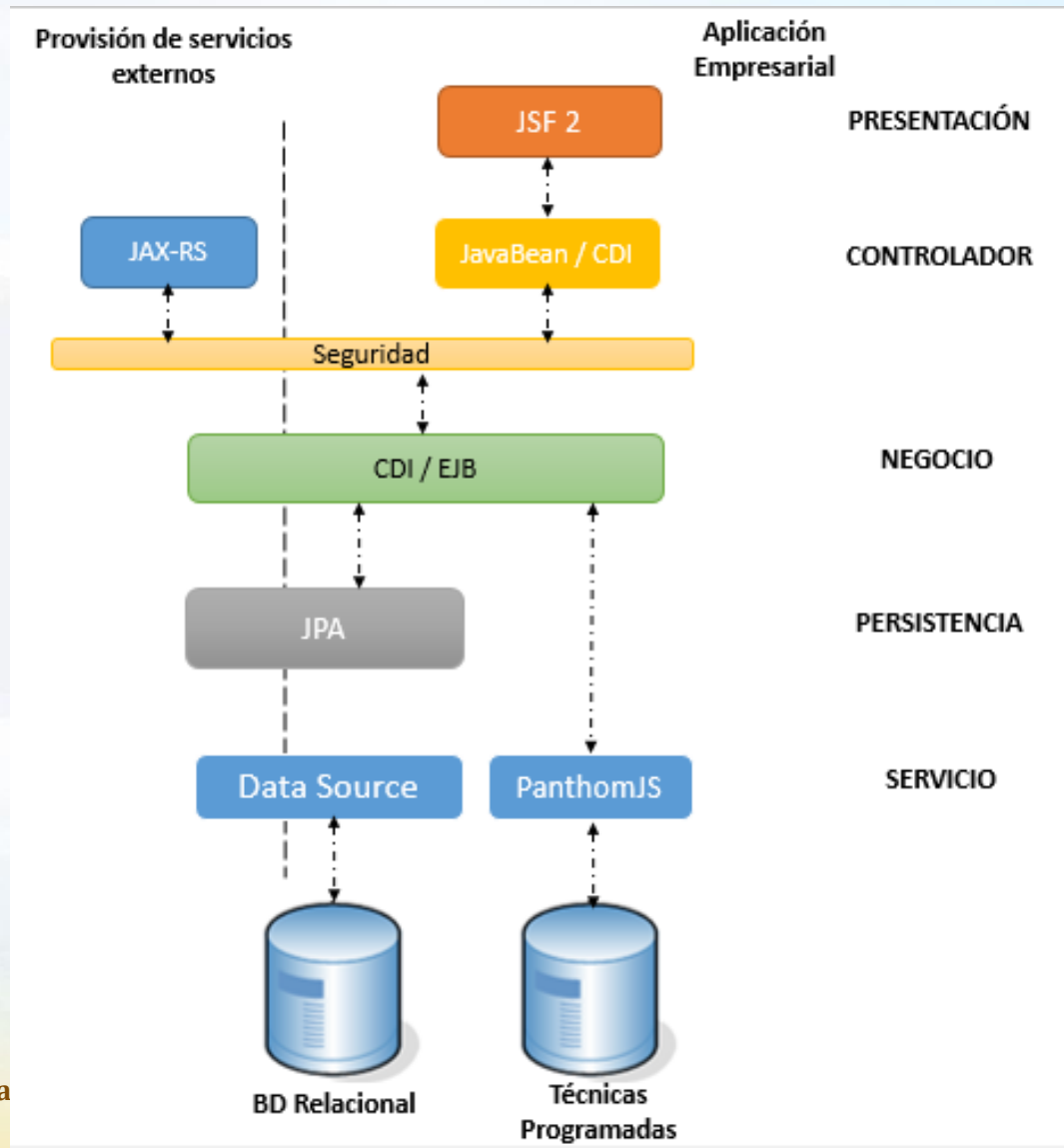
## Arquitectura JEE (3 capas lógicas)



# Arquitectura (ejemplo)



# Arquitectura (ejemplo)



# Instalación de Plataforma de desarrollo

- Instalar JDK 1.8
- Descargar e instalar Eclipse IDE for Enterprise Java Developers de la página: <https://www.eclipse.org/downloads/packages/>

## Eclipse IDE for Enterprise Java Developers (includes Incubating components)

400 MB 173,230 DOWNLOADS



Tools for developers creating Java Enterprise and Web applications, including a Java IDE, tools for Web Services, JPA and Data Tools, JSF, Mylyn, Maven and Gradle, Git, and more.

Click [here](#) to file a bug against Eclipse Web Tools Platform.

Click [here](#) to file a bug against Eclipse Platform.

Click [here](#) to file a bug against Maven integration for web projects.

Click [here](#) to report an issue against Eclipse Wild Web Developer (incubating).



Windows 64-bit  
Mac Cocoa 64-bit  
Linux 64-bit

- Descargar servidor de aplicaciones a usar Wildfly 18.0.1.Final de la página: <https://wildfly.org/downloads/>

18.0.1.Final	2019-11-14	Java EE Full & Web Distribution	LGPL	178 MB	<a href="#">ZIP</a>	SHA-1
				177 MB	<a href="#">TGZ</a>	SHA-1
		Servlet Only Distribution	LGPL	41 MB	<a href="#">ZIP</a>	SHA-1

# JEE (Jakarta) 8 – EJB Enterprise Java Beans



# EJB (Enterprise Java Beans)

## ¿Qué es?

Un Enterprise Bean es un componente del lado del servidor que **encapsula la lógica de negocios** de una aplicación. (proposito de la aplicación)

## Beneficios:

- Simplifican el desarrollo de grandes aplicaciones distribuidas.
  - Desarrollador (dedicado a construir la lógica)
  - Contenedor EJB (servicios a nivel de sistema: transaccionalidad, seguridad, etc.)
- Clientes ligeros (posibilita la separación de la presentación)
- Se puede construir nuevas aplicaciones a partir de beans existentes

## Cuando usar?

- Aplicación escalable (cabida a más usuarios y funcionalidades)
- Se requiera garantizar la Integridad de los datos (transaccionalidad)
- Se requiera variedad de clientes (clientes remotos)

# EJB (Enterprise Java Beans)

## Tipos de EJB

Tipo		
Session Beans	Sesión	Realiza una tarea para un cliente (funcionalidad); Posibilitan guardar datos  @Stateless (Sin estado) @Stefull. (Con estado por sesión) @Singleton (Con estado por aplicación)
Message-driven Beans	Impulsados por mensajes	Actúa como escucha para un tipo de mensajería particular, como la API de Java Message Service

# EJB (Enterprise Java Beans)

## Session Beans

Encapsula la lógica empresarial que un cliente puede invocar mediante programación a través de:

- Vistas de cliente de servicio web,
- Invocación remoto o
- invocación local

Con estado @Stateful	Son estado @Stateless	De aplicación @Singleton
En un bean de sesión con estado, las variables de instancia representan el estado de una sesión única de cliente	Un bean de sesión sin estado no mantiene un estado de conversación con el cliente	Un bean de sesión singleton se instancia una vez por aplicación y existe para el ciclo de vida de la aplicación.  Se puede instanciar al inicio de la aplicación ( tareas de inicialización)

# Practica

Ejemplo de cliente en:

<https://gitlab.com/appdis/01-ejb-cliente>

Ejemplo de servidor en:

<https://gitlab.com/appdis/01-ejb-server>

# JEE (Jakarta) 8 – JPA Java Persistence API

# JPA (Java Persistence API)

## ¿Qué es?

JPA proporciona un recurso de mapeo de objeto/relacional para administrar datos relacionales y persistirlos en una BD, en aplicaciones Java

## Incluye:

- La API de persistencia de Java
- El lenguaje de consulta
- La API de criterios de persistencia de Java
- Mapeo relacional (Objeto/metadatos)



## JPA -> Características

- Persistencia utilizando POJOs
- No intrusivo  
(no se requiere extender funcionalidades)
- Consultas a la BD utilizando Objetos JAVA
- Configuración simplificada  
(tiene configuraciones por defecto)
- Integración simple con las demás capas

```
@Entity
public class Persona {

    @Id
    @GeneratedValue
    private Long personaId;

    @Column(nullable = false)
    private String nombre;

    private String apePaterno;

    private String apeMaterno;
    private String email;
    private Integer telefono;

    // Constructores, getters, setters
}
```

## JPA -> Entidades

- Típicamente, una entidad representa una tabla en una base de datos relacional
- Cada instancia de entidad corresponde a una fila en esa tabla
- Usa anotaciones de mapeo objeto/relacional para mapear las entidades y las relaciones de entidad con los datos relacionales en la BD

### Requisitos

- Debe ser anotada con `javax.persistence.Entity` → `@Entity`.
- Debe tener un constructor público o protegido sin argumentos. Si se puede tener otros constructores.
- No debe ser declarada **final** así también no deben declarar métodos ni variables como **final**.
- Si fuese consumida o devuelta de manera remota debe ser `Serializable`
- Las variables de instancia persistentes deben declararse privadas, protegidas o privadas de paquetes y solo se puede acceder directamente a ellas mediante los métodos de la clase de entidad (getter y setters).

```
@Entity
public class Persona {

    @Id
    @GeneratedValue
    private Long personaId;

    @Column(nullable = false)
    private String nombre;

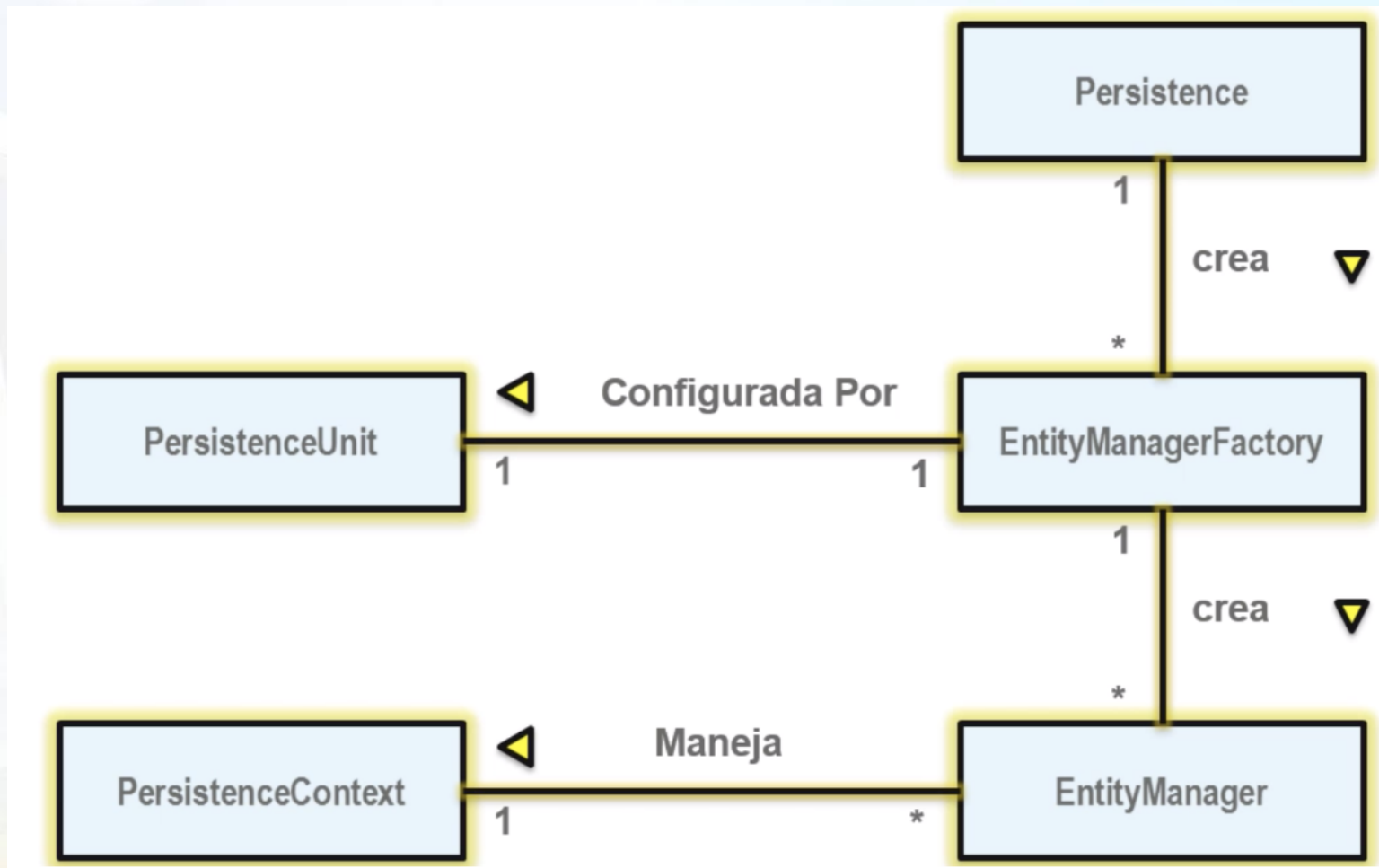
    private String apePaterno;

    private String apeMaterno;
    private String email;
    private Integer telefono;

    // Constructores, getters, setters
}
```



## JPA -> EntityManager



## JPA -> Configurando la unidad de persistencia

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<persistence version="2.2"
  xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">
  <persistence-unit name="PersonaService">
```

```
    <jta-data-source>java:jboss/datasources/my-app-webappDS</jta-data-source>
    <properties>
```

```
      <property name="javax.persistence.schema-generation.database.action" value="drop-and-create"/>
      <property name="javax.persistence.schema-generation.create-source" value="metadata"/>
      <property name="javax.persistence.schema-generation.drop-source" value="metadata"/>
```

```
      <!-- Properties for Hibernate -->
```

```
      <property name="hibernate.show_sql" value="true" />
```

```
    </properties>
```

```
  </persistence-unit>
```

```
</persistence>
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- This is an unmanaged datasource. It should be used for proofs of concept
or testing only. It uses H2, an in memory database that ships with JBoss
AS. -->
```

```
<datasources xmlns="http://www.jboss.org/ironjacamar/schema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.jboss.org/ironjacamar/schema http://docs.jboss.org/ironjacamar/schema" />
```

```
<!-- The datasource is bound into JNDI at this location. We reference
this in META-INF/persistence.xml -->
```

```
<datasource jndi-name="java:jboss/datasources/my-app-webappDS"
  pool-name="jboss-javaee-webapp" enabled="true"
  use-java-context="true">
```

```
  <connection-url>jdbc:h2:mem:my-app-webappDS;DB_CLOSE_ON_EXIT=FALSE;DB_CLOSE_DELAY=1000</connection-url>
```

```
  <driver>h2</driver>
```

```
<security>
```

## JPA -> Configurando la unidad de persistencia

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is an unmanaged datasource. It should be used for proofs of concept
or testing only. It uses H2, an in memory database that ships with JBoss
AS. -->
<datasources xmlns="http://www.jboss.org/ironjacamar/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.jboss.org/ironjacamar/schema http://docs.jboss.org/ironjacamar/schema/datasources_1_0.xsd">
<!-- The datasource is bound into JNDI at this location. We reference
this in META-INF/persistence.xml -->
<datasource jndi-name="java:jboss/datasources/my-app-webappDS"
pool-name="jboss-javaee-webapp" enabled="true"
use-java-context="true">
<connection-url>jdbc:h2:mem:my-app-webappDS;DB_CLOSE_ON_EXIT=FALSE;DB_CLOSE_DELAY=-1</connection-url>
<driver>h2</driver>
<security>
<user-name>sa</user-name>
<password>sa</password>
</security>
</datasource>
</datasources>
```

## JPA -> Uso de la unidad de persistencia. (DAO)

```
@Stateless
public class PersonaServiceBean implements PersonaService {

    @PersistenceContext(unitName="PersonaService")
    EntityManager em;

    public void agregarPersona(Persona persona) {
        em.persist(persona);
    }

    public Persona encontrarPersona(int idPersona) {
        return em.find(Persona.class, idPersona);
    }

    public Persona modificarNombrePersona(int idPersona, String nuevoNombre) {
        Persona persona = em.find(Persona.class, idPersona);
        if (persona != null) {
            persona.setNombre(nuevoNombre);
        }
        return persona;
    }

    public void eliminarPersona(int idPersona) {
        Employee emp = em.find(Employee.class, idPersona);
        em.remove(emp);
    }
}
```