**Aplicación Empresarial:**

* Componentes:
  1. Empresa.
  2. Requerimientos Funcionales (procesos, características y features que debemos desarrollar).
  3. Aplicación (la suma de componentes articulados que desarrollan la solución).
* Las aplicaciones empresariales se componen de múltiples componentes modulares y distribuidos en capas, es decir, aunque para los usuarios se les presenta la aplicación como un único servicio, puede que diferentes partes (ejemplo una parte está en cliente-servidor, una parte web, cliente móvil, etc) de la aplicación estén construidos sobre lenguajes de programación y base de datos distintos.
* Tipos de aplicaciones:
  1. A la medida (orientadas a solucionar un problema en específico).
  2. Multi Target (desarrolladas para mostrar múltiples servicios a distintos usuarios con diferentes necesidades, como por ejemplo las redes sociales).
* Se debe tener en cuenta el número de usuarios, los usuarios concurrentes (cantidad de usuarios que se conectan al mismo tiempo), respaldos de información, seguridad, soporte 24/7 todo el año, entre otras cosas.
* Las aplicaciones híbridas, donde existen diferentes Fronts que se precisan al usuario según sus necesidades.
* Existen conjuntos de herramientas y tecnologías que mejoran la calidad de la construcción de la aplicación como patrones de diseño, librerías de optimización, formas de empaquetar y configurar las apps, procesos de automatización y metodología de desarrollo.

**Aplicaciones web**

Normalmente, las aplicaciones web se componen de los siguientes elementos: el cliente (navegadores como Chrome, Firefox y Safari), el servidor web (donde alojamos la aplicación y llegan las peticiones) y la base de datos (donde persistimos toda la información de los usuarios y la aplicación).

Las aplicaciones web se acceden a través de una URL o dirección web, compuesta por los siguientes elementos: Protocolo de comunicación (http:// y https://), el dominio o IP del servidor (en producción encontramos dominios como www.platzi.com y desarrollo vemos 4 números separados por puntos + el puerto 192.168.0.11:8080) y el contexto o sección de la aplicación (por ejemplo, /appventas o /cursos/marca).

Ventajas de las aplicaciones web:

* Facilidad para su instalación y actualización
* Ahorro de recursos en equipos y dispositivos
* Compatibilidad Multiplataforma (independencia del Sistema Operativo)
* Soporte para múltiples usuarios concurrentes
* Acceso multidispositivo (computadoras, tablets, TV, teléfono móvil, etc)
* Soporte para peticiones síncronas y asíncronas.

**Servidor Web:**

* Flujo: Los usuarios acceden a las aplicaciones web por medio de un cliente o navegador, envían todos los requerimientos (peticiones/solicitudes o request) al servidor web que analiza la información, hace las consultas a base de datos o procesan la petición de manera adecuada para luego devolver al navegador la respuesta (o response) que se generará para el usuario.
* Contiene las características para soportar los usuarios concurrentes, manejar múltiples hilos, recursos, conectarse a diferentes bases de datos, etc.
* Servidor Web vs Servidor de Aplicaciones:
  + En el servidor web tiene un contenedor web que permite procesar o manejar todas las peticiones que son manejadas con tecnologías Java ([servlets](https://es.wikipedia.org/wiki/Java_Servlet), [JSP](https://es.wikipedia.org/wiki/JavaServer_Pages), [JSF](https://es.wikipedia.org/wiki/JavaServer_Faces)).
  + En el servidor de aplicación tiene un contenedor de negocio de [EJB](https://es.wikipedia.org/wiki/Enterprise_JavaBeans) y también un contenedor web.
  + En la mayoría de lenguajes de programación sólo disponen de los servidores web. Son contenedores web encargados de manejar todas las peticiones (como [Tomcat](https://es.wikipedia.org/wiki/Tomcat), [Jetty](https://es.wikipedia.org/wiki/Jetty) Server, GlassFish Web Profile, Apache Server (en PHP), IIE ‘Internet Information Server’ (en .NET)).
  + Desde la versión 1.4 de Java EE, también se utilizará los servidores de aplicación, donde se utiliza un componente de [EJB](https://es.wikipedia.org/wiki/Enterprise_JavaBeans)(Enterprise JavaBeans) que nos permite manejar la capa de negocios sin descuidar temas como la transaccionalidad, persistencia y el pull de conexiones.
  + Los servidores de aplicación actuales para Java existen el [JBoss](https://www.redhat.com/es/technologies/jboss-middleware/web-server), [WildFly](https://es.wikipedia.org/wiki/WildFly), [Payara](https://en.wikipedia.org/wiki/Payara_Server) Server, [GlassFish](https://es.wikipedia.org/wiki/GlassFish) Server para manejar temas de EJB.

**Servlet**

Los Servlets son componentes o clases de Java del lado del servidor web que permiten procesar peticiones del cliente y responderlas a través de la generación de contenido dinámico o redireccionarlas a otros recursos. Los Servlets de tipo HttpServletson los más utilizados ya que funcionan con el protocolo HTTP (lo encontramos en el paquete javax.servlet.http) pero, también podemos utilizar los GenericServletssi necesitamos cualquier otro protocolo (el paquete es javax.servlet).

**JSP**

Las JSP (JavaServer Page) son componentes del lado del servidor que nos permiten desarrollar páginas web con soporte para contenido dinámico gracias a la inclusión de código Java en código html utilizando los Tags (una sintaxis especial que podemos utilizar en cualquier momento para añadir contenido dinámico: <% for(1=0; i<10: i++) %>).

El desafío de esta clase es crear una página con JSP para visualizar la información de los usuarios que se registran.

**Aplicaciones web**

Así como la construcción de casas, el proceso de construcción de las aplicaciones consiste en 3 etapas: definición de los todos requerimientos y características (lenguaje de programación, tipo de base de datos, practicas de seguridad a implementar, etc), después, el diseño de los componentes y cómo se van a comunicar entre ellos (para darle una vista previa al usuario de cómo vamos a desarrollar la aplicación) y, por ultimo, el desarrollo de nuestra aplicación en base a las decisiones de las etapas anteriores.

**Spring Framework**

Spring Framework es un framework Open Source creado por Rod Jhonson en 2003, fue creado debido a la necesidad de optimizar nuestro código Java y como una alternativa para solucionar la complejidad de otras tecnologías más pesadas en ese momento (especialmente EJB).

Características de Spring:

* NO esta limitado al lado del servidor
* Desarrollo basado en POJOS
* Bajo acoplamiento
* Programación declarativa
* Reducción de código boilerplate
* Arquitectura en capas

Características de Spring Boot

Spring Boot es una tecnología que permite optimizar los tiempos de desarrollo en la creación y despliegue de proyectos permitiéndonos a los desarrolladores enfocarnos en el desarrollo de la aplicación. Entre las características que tiene Spring Boot se puede mencionar las siguientes:

Aplicaciones Standalone

* Una aplicación en Spring Boot es empaquetada en un JAR con todas las dependencias para poder ejecutarse sin necesidad de instalar un servidor web. Permitiendo funcionar como una aplicación standalone.

Servidores Embebidos

* Spring Boot soporta a Tomcat y Jetty como servidores embebidos.

Configuración Simple

* Spring Boot soporta cada una de las características de los mòdulos de Spring como son Spring MVC, Spring Data, Spring Rest, Spring Security y lo hace de una manera simple, a través de dependencias simples una sola por cada tecnologìa. Adicionalmente la forma de construirla y configurarla es simple y óptima de manera online a través de la herramienta Spring Initializr.

Características de Producción Listas

* Spring Boot viene con características de configuración predefinidas para ambientes de producción. Adicionalmente se puede configurar características de acuerdo a la ambiente a través de Spring Profiles.

**Persistencia de Objetos**

En el mundo del desarrollo de software de cualquier aplicación independientemente la tecnología, plataforma en que esté desarrollado o se ponga en producción. Toda aplicación necesita almacenar su información, tomando en cuenta que lo más preciado e importante es dicha información en base a la cual gira el desarrollo del negocio y se insume la aplicación para la automatización de los procesos.

Esta información debe poder ser recuperada en cualquier momento dado tal como fue ingresada, a este proceso de almacenar información y poder recuperarla en cualquier momento dado de forma íntegra independientemente el medio de almacenamiento, se la conoce con el nombre de persistencia.

Existen varios mecanismos de persistencia entre estos se tiene archivos planos, archivos binarios, base de datos relacional, base de datos documentales, etc.

A nivel de especificaciones o tecnologías utilizadas en Java para poder persistir información, se tiene las siguientes:

JDBC

* Conjunto de clases para poder gestionar la información de la base de datos de forma natural utilizando sentencias DML y DDL.  
  Capas de Persistencia.- También conocida como ORM, ejemplos de estás se tiene a Hibernate, EclipseLink, ToplLink, Datanucleos, entre otras. Las cuales permiten persistir información y gestionar la data pero como objetos. gracias a que una de las principales características de este tipo de capas de persistencia es mapear un modelo relacional a un modelo objetual.

JPA

* Es un administrador de persistencia que utiliza a una capa de persistencia como proveedor de persistencia. Otorgando caracterìsticas estandarizadas en cuanto al tema transaccional, consultas y pool de conexión independientemente de la capa de persistencia.

Cabe mencionar finalmente que JPA es la especificaciòn jerárquica que utiliza a las capas de persistencia, y éstas utilizan JDBC para persistir la información.

**Docker**  
#version  
docker -v

#bajarse postgres  
docker pull postgres:9.6.6-alpine

#listar imagenes  
docker images

#borrar imagenes  
docker rm 4f1b17bd2638

\*\*#crea contenedor \*\*  
docker run -d --name postgres -p 5432:5432 -e POSTGRES\_PASSWORD=platzi postgres:9.6.6-alpine

#listar contenedores  
docker ps -a

#borrar contenedores  
docker rmi 587aa1d0e586

#detener contenedor  
docker stop postgres

#iniciar/reiniciar contenedor  
docker restart postgres

**Seguridad**

OWASP es un proyecto open source dedicado a determinar y combatir las causas que hacen que el software sea inseguro (enfocado sobre todo a aplicaciones Java y .Net). Al construir aplicaciones web debemos considerar los siguientes factores de seguridad:

Filtros de Autenticación y Manejo de la Autorización (URLs y contenido al que los usuarios pueden acceder o no dependiendo de sus permisos)

Control de Páginas de Error (configuración de la respuesta que entregamos ante errores tipo 500, 404, entre otros)

Envío de peticiones seguras

**Front-End**

No utilizar JSP por limitaciones con SprnigBoot. Usar Thymeleaf.