**¿Qué es una aplicación empresarial?**

Las Aplicaciones están conformadas por 3 componentes principales: La Empresa, sus Requerimientos Funcionales (los procesos, características y features que debemos desarrollar) y la Aplicación Empresarial (la suma de componentes articulados que desarrollan la solución).

También debemos tener en cuenta que las aplicaciones se componen de múltiples componentes modulares y distribuido en capas, es decir, aunque a los usuarios se les presenta la aplicación como un único servicio, puede que diferentes partes de la aplicación estén construidos sobre lenguajes de programación y bases de datos distintos.

Existen dos tipos de aplicaciones: Aplicaciones a la medida (orientadas a solucionar un problema en específico) y Aplicaciones Multi Target (desarrolladas para mostrar múltiples servicios a distintos usuarios con diferentes necesidades, así como las redes sociales).

También encontramos características especiales y sensible en este tipo de aplicaciones: el número de usuarios, usuarios concurrentes (una gran cantidad de usuarios conectados al mismo tiempo), respaldos de la información, soporte 24/7 y 365 días al año, entre otras.

**¿Cómo funcionan las 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](http://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

**Servidores Web vs Servidores de Aplicación**

Normalmente, los usuarios acceden a las aplicaciones web por medio de un cliente o navegador que envía todos los requerimientos al servidor web que, después de analizar la información, se encarga de realizar algunas consultas a la base datos o procesar la petición de manera adecuada para devolver al navegador la respuesta que generáramos para el usuario.

La mayoría de lenguajes de programación solo disponen de los servidores web, contenedores web encargados de manejar todas las peticiones (como Tomcat, Jetty Server, GlassFish Web Profile, Apache Server en aplicaciones de PHP o Internet Information Server para aplicaciones .NET). Pero, desde la versión 1.4 de Java EE, también utilizamos los servidores de aplicación, servidores donde utilizamos un componente de EJB (Enterprise JavaBeans) que nos permiten manejar la capa de negocios sin descuidar temas como la transaccionalidad y el pull de conexiones.

**Servidor Web:** A nivel de componentes un servidor web está formado por un contenedor web, el cual permite poder procesar o manejar todas las peticiones realizadas con tecnologías Java como los servlets.

**Servidor Aplicación:** Nació como un componente adicional, es un contenedor de EJB (Enterprise JavaBeans) el cual es el componente principal para el maneja del negocio.

**Patrón de diseño MVC**

Los Patrones de Diseño funcionan a partir de una serie de capas o secciones con sus respectivas responsabilidades, que se comunican entre sí, pero deben estar muy bien separadas para agilizar el desarrollo. Gracias a los patrones de diseño podemos reducir la duplicación de código y facilitar mantenimiento de la aplicación.

El Patrón de Diseño MVC tiene los siguientes elementos:

* **Capa de Modelo**: Organización y estructura de todas las clases o componentes relacionados con la base de datos.
* **Capa de Negocio**: Reglas, análisis y requerimientos funcionales principales o secundarios de la aplicación.
* **Capa de Vista**: Formularios y componentes visuales con los que los usuarios deben interactuar.

**¿Qué son los 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 HttpServlet son los más utilizados ya que funcionan con el protocolo HTTP (lo encontramos en el paquete javax.servlet.http) pero, también podemos utilizar los GenericServlets si necesitamos cualquier otro protocolo (el paquete es javax.servlet).

**Características y práctica de 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++) %>).

**Arquitectura de las 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, prácticas 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 último, el desarrollo de nuestra aplicación en base a las decisiones de las etapas anteriores.

**Características de 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 está limitado al lado del servidor
* Desarrollo basado en POJOS (Plan Old Java Object, Objeto Java Plano Antiguo).
* Bajo acoplamiento
* Programación declarativa
* Reducción de código boilerplate
* Arquitectura en capas

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 módulos 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 tecnología. 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 al 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 características estandarizadas en cuanto al tema transaccional, consultas y pool de conexión independientemente de la capa de persistencia.

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

**Optimizar Pojos con Lombok**

Los POJOs (Plain Old Java Objects) son clases simples de Java que no dependen de un framework en especial y Lombok es una librería que nos permite eliminar código repetitivo (Getters y Setters) que todavía estamos obligados a escribir cuando trabajamos con Java.

**Añadir JPA Repository**

Los repositorios son la herramienta principal de Spring Data, trabajan sobre una capa de abstracción un poco más alta en las operaciones CRUD y nos evitan el trabajo de crear una clase abstracta con todas las operaciones CRUD relacionadas con la base de datos y, a partir de esta clase, crear una nueva clase para administrar las diferentes entidades. Existen varios repositorios de acuerdo a la tecnología a utilizar, como CrudRepository, JPARepository y MongoRepository (ambos basados en PageAndSortingRepository).

**Implementar las operaciones de Consulta con JPA Repository**

Spring Data cuenta con soporte para realizar consultas personalizadas basado en los atributos de la clase base que se generan de forma automática en el tiempo de ejecución (Query Generation Strategy). También soporta otro tipo de consultas basadas en JPQL (Java Persistence Query Language) como @Query y @NamedQuery.

**Implementar servicio de negocio para las operaciones CRUD**

Seguimos trabajando en la parte de negocio, ahora que definimos nuestro repositorio, el siguiente paso la parte de servicios (con el paquete com.platzi.ereservation.negocio.service), vamos a trabajar algunas clases para exponer estas operaciones en la capa de negocio. Recuerda que debemos indicar que las clases son un bean de la capa de negocios utilizando la anotación @Service.

**Manejar la transaccionalidad de un servicio de negocio**

Las transacciones son el conjunto de operaciones que afectan la base de datos (updates o deletes, no se recomienda realizar consultas de solo lectura) y se ejecutan en bloque, es decir, se ejecutan todas con éxito (commit) o no ejecutamos ninguna (rollback). Las transacciones empiezan y terminan a nivel de servicio y nunca a nivel de capa de datos, para indicar que una clase o método será transaccional utilizamos a anotación @Transactional.

**Instalar y configurar Swagger**

Swagger es una herramienta que nos permite documentar y ejecutar APIs Web de forma interactiva, provee una manera muy sencilla de implementar clientes un múltiples lenguajes. Para configurar esta herramienta junto con Spring, vamos a utilizar el proyecto Spring Fox configurando las dependencias en el archivo pom.xml.

**Características de los Servicios Web y Spring Rest**

Los servicios web son aplicaciones construidas con el fin de poder intercambiar información con otras aplicaciones utilizando protocolos estandarizados como SOAP y REST (utilizando los formatos XML y JSON, respectivamente) sin que los usuarios se enteren de que están navegando entre aplicaciones o servicios diferentes. Este concepto lo conocemos como interoperabilidad.

Por otra parte, Spring Rest es una manera de construir servicios web con Spring utilizando la arquitectura REST y aprovechando la experiencia de Spring MVC. Para configurar los servicios web utilizamos la anotación @RestCotroller en la clase base, la cual combina los comportamientos de las anotaciones @Controller y @ResponseBody.

**Manejar la seguridad en una aplicación web**

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

**Características, instalación e integración de Spring Security**

Spring Security es un framework que nos permite gestionar completamente la seguridad de nuestras aplicaciones Java. Entre sus características podemos encontrar las siguientes:

* Gestión de la seguridad en varios niveles
* Configuración de seguridad portable
* Soporte para múltiples modelos de autenticación (HTTP Basic, LDAP, OAuth, HTTP Digest, entre otros)

**Realizar la configuración de Docker Plugin**

Una vez terminada nuestra aplicación vamos a configurar el plugin de Docker en nuestro archivo pom.xml. Recuerda que debes tener docker instalado y puedes aprender mucho mejor cómo funcionan los despliegues de aplicaciones con contenedores en el Curso de Fundamentos de Docker de Platzi.

**Crear una imagen docker de la aplicación**

Después de configurar el plugin de Docker vamos a construir una imagen de nuestra aplicación que podremos revisar utilizando el comando docker images. Para construir la imagen debemos crear un archivo application-pro.yaml con la configuración de nuestra aplicación en producción, configurar el nombre del proyecto final en el archivo pom.xml y crear una tarea de ejecución en el IDE.