

TALLER WEB 1 -  
Hablando sobre MAVEN  
MAVEN maneja las dependencias del proyecto pero ademas

### QUE SON LAS DEPENDENCIAS?

Las dependencias son como las librerias que necesita el proyecto y cada una de esas librerias ya tiene problemas resueltos.

Por ejemplo la conexion a la base de datos, o calcular cuanto tiene que aportar una persona segun su sueldo, etc con solo enviar los parametros ya hay metodos, algoritmo que lo resuelve.

```
<!--plugin-->  
<groupId>org.eclipse.jetty</groupId>  
<artifactId>jetty-maven-plugin</artifactId>  
<version>9.4.45.v20220203</version>  
<configuration>  
  <scanIntervalSeconds>2</scanIntervalSeconds>  
  <httpConnector>  
    <port>8080</port>  
  </httpConnector>  
  <webApp>  
    <contextPath>/spring</contextPath>  
  </webApp>  
</configuration>  
</plugin>
```

Este plugin lo que hace es esta descargando internamente, todo esto lo hace maven. Por medio del Maven podemos configurar como levantar el proyecto.

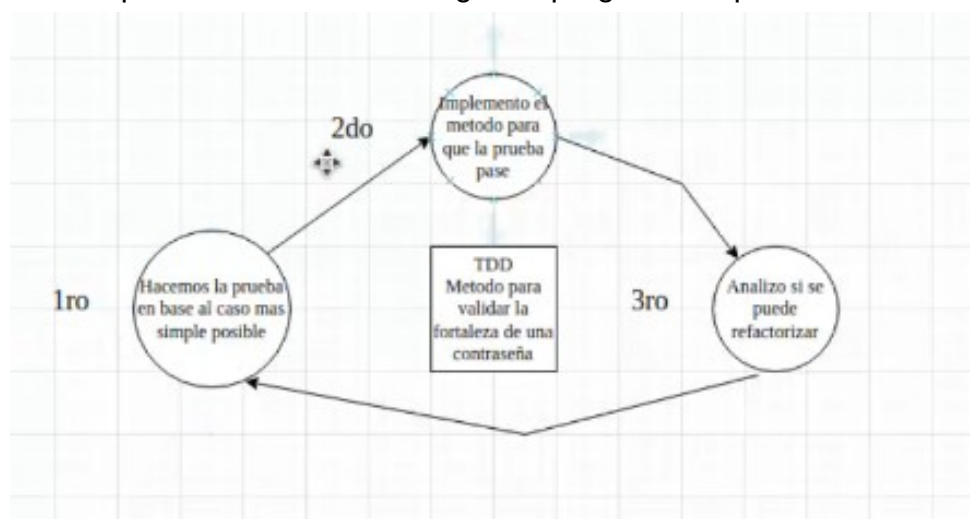
Este plugin me facilita descargarme un servidor llamado JETTY (como el TOMCAT)  
El JETTY me permite levantar la aplicacion y verla disponible en mi localhost 8080

MAVEN es parecido al NODE

Node es el motor de JS pero el modo de manejarse es parecido maven usa mvn en cambio node npm parecidos el manejo

Que es TDD?

Test Driven Development es una metodologia de programar a partir de casos de pruebas



Debe validar si la fortaleza de la contraseña es DEBIL, MEDIANA, FUERTE

1. Debo construir un metodo que valide la contraseña en base a ciertos arreglos

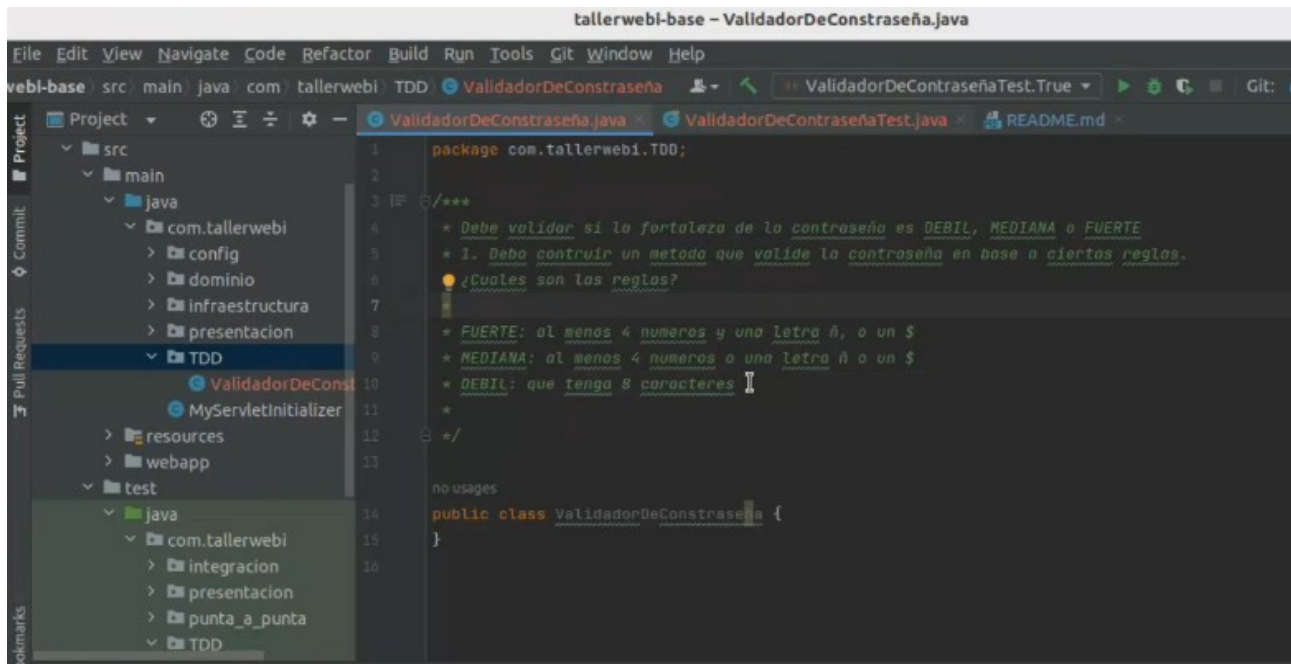
Cuales son las reglas?

FUERTE: Al menos 4 numeros y una letra enie o \$

MEDIANA: Al menos 4 numeros o una letra enie o \$

DEBIL: que tenga al menos 8 caracteres

INVALIDA: Menos de 8 caracteres



Crear en la carpeta test package TDD la clase ValidadorDeContraseñaTest con el sufijo test para hacer test junit

## Como correr un test en junit?

En intellij con el botoncito verde en el @Test

o usando maven en la terminal

**mvn clean test -Dtest=ValidadorDeContraseñaTest**

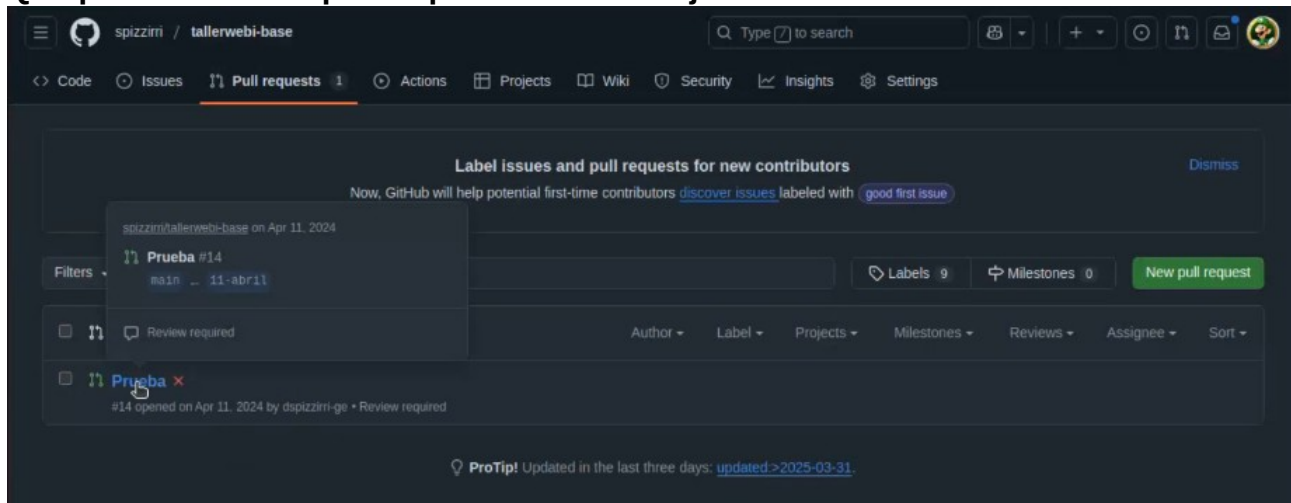
**si quiero correr todas las pruebas del proyecto uso**

**mvn clean test**

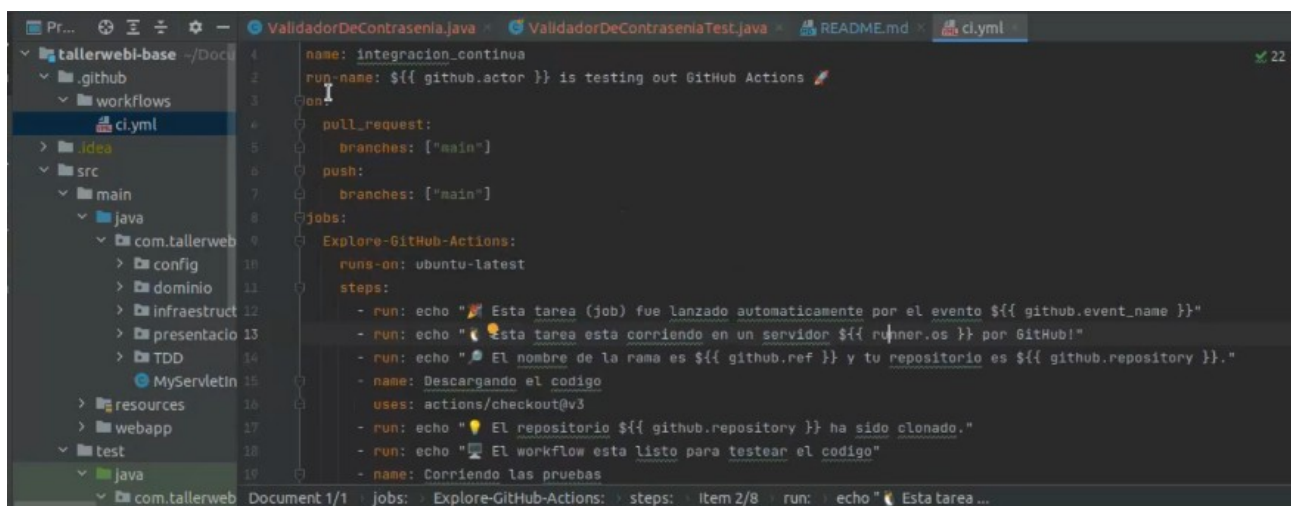
**Para ver si funciona el proyecto lo levantaremos en un web server llamado Jetty el cual lo va a levantar en una url especifica localhost:8080**

## GITHUB

### Que pasa si nuestro pull request sale cruz roja

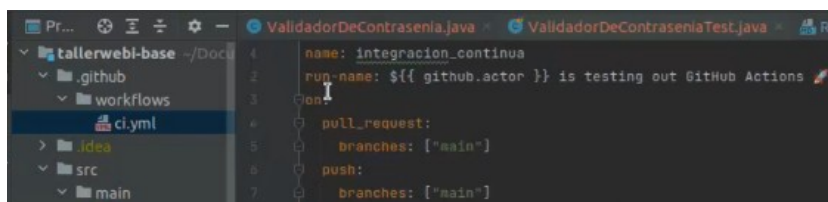


La cruz significa que los checks que ya tiene configurado nuestro proyecto es que las pruebas unitarias tienen que dar verde. Esa configuracion se hace en un archivo llamado **ci.yml** (esta en el proyecto base, en la carpeta **.github** → **workflows** → **ci.yml**)



0

Este archivo de configuracion lo que hace es que cuando hacemos un pull request en github, ese archivo corre y se fija en que momento tiene que correr Github lo analiza y se fija aqui tengo un archivo ci.yml y ve que hay eventos y ese evento le dice este archivo se tiene que ejecutar cuando hay un pull request o cuando se hace un push en la rama main.



Y que es lo que hace ese archivo o sea que hace cuando se hace un pull request o push se ejecuta?

Esto lo que va hacer en el servidor de Github va levantar una maquina virtual que va tener Linux y esa maquina virtual va descargar el codigo que estamos subiendo en el repositorio.

```
branches: { main }
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🚀 Esta tarea (job) fue lanzado automaticamente por el evento ${ github.event_name }"
      - run: echo "🌟 Esta tarea esta corriendo en un servidor ${ runner.os } por GitHub!"
      - run: echo "📁 El nombre de la rama es ${ github.ref } y tu repositorio es ${ github.repository }."
      - name: Descargando el codigo
        uses: actions/checkout@v3
      - run: echo "💡 El repositorio ${ github.repository } ha sido clonado."
      - run: echo "🔧 El workflow esta listo para testear el codigo"
      - name: Corriendo las pruebas
```

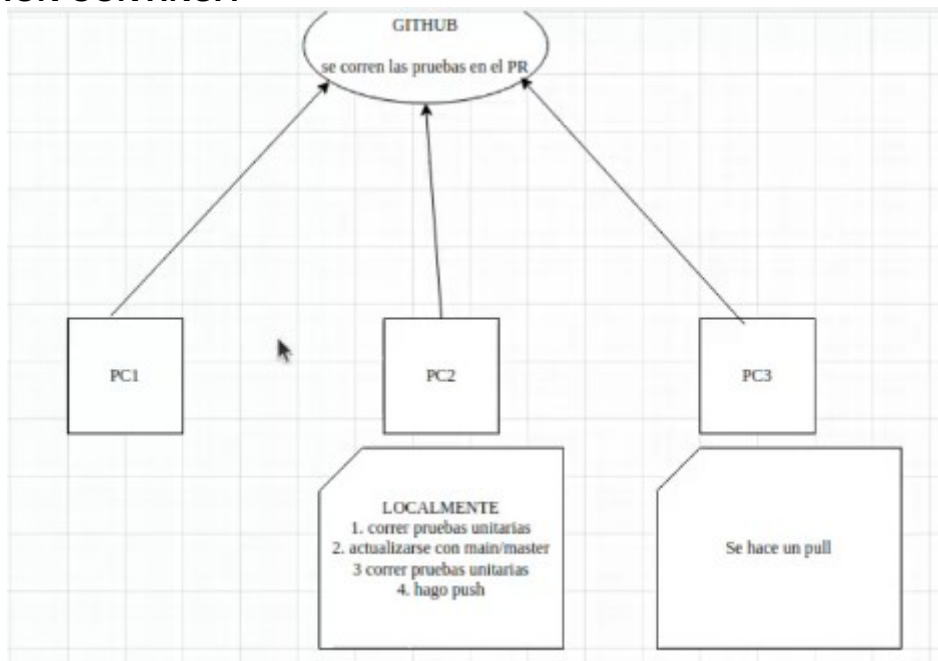
hent 1/1 | jobs: Explore-GitHub-Actions: | steps: | Item 2/8 | run: | echo "🌟 Esta tarea ...

Y lo va a descargar para ejecutar las instrucciones que le vamos a decir, en este caso la unica instruccion que le decimos es mvn clean test, es decir va crear una maquina virtual que va vivir unos min en el servidor de github para que el codigo de nuestra rama se descargue ahi y corran las pruebas y despues esa maquina muere.

Y por que necesitamos que haga esto?

Para que la persona que haga ese pull request o push sepa si las pruebas unitarias estan pasando o no, entonces en caso que no pase las pruebas unitarias o sea en que alguna falle, uno va a ver un mensaje de este estilo

## INTEGRACION CONTINUA





spring-webmvc: es la dependencias que usamos web

spring-orm: es la dependencias que usamos para la conexion de la base de datos

spring-jdbc:

hsqldb: es una base de datos en memoria para testear se lo podria usar

thymeleaf: es la dependencias que usaremos en la parte visual

bootstrap

jakarta-servlet: Los servlet para poder construir endpoint

org.springframework

org.junit.jupiter

org.mockito

org.hamcrest

org.hibernate

playwrite

Cada una de estas dependencias estan incluidas en el archivo pom.xml para cuando usemos Maven, maven lo escanea y se encarga de buscar los repositorios de cada dependencias y ponerlo disponible para el proyecto.

En la seccion <build> tambien tenemos configuraciones pero generalmente no se toca, la unica parte que se podria tocar es la parte de pruebas para indicarle que se fije todo lo que termine en Test.java que lo ejecute como test

```
138         <configuration>
139             <failOnMissingWebXml>false</failOnMissingWebXml>
140         </configuration>
141     </plugin>
142
143     <!-- Plugin para ejecutar las pruebas -->
144     <plugin>
145         <groupId>org.apache.maven.plugins</groupId>
146         <artifactId>maven-surefire-plugin</artifactId>
147         <version>3.0.0-M5</version> <!-- Ajusta la versión según tus necesidades -->
148
149         <configuration>
150             <!-- Configuración específica para las pruebas -->
151             <includes>
152                 <include>*/**Test*.java</include>
153             </includes>
154         </configuration>
155     </plugin>
156
157     <plugin>
158         <groupId>org.eclipse.jetty</groupId>
159         <artifactId>jetty-maven-plugin</artifactId>
160         <version>9.4.45.v20220823</version>
```



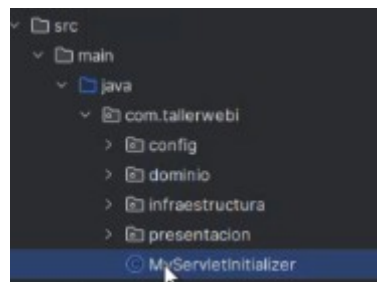
En la parte de build donde se configura el server Jetty podemos cambiar el puerto 8080 y en el contextPath que es el path de la url de nuestro sistema localhost:8080/spring/...

```
<plugin>
  <groupId>org.eclipse.jetty</groupId>
  <artifactId>jetty-maven-plugin</artifactId>
  <version>9.4.45.v20220203</version>
  <configuration>
    <scanIntervalSeconds>2</scanIntervalSeconds>
    <httpConnector>
      <port>8080</port>
    </httpConnector>
    <webApp>
      <contextPath>/spring</contextPath>
    </webApp>
  </configuration>
</plugin>
```

## Estructurar el proyecto

Tenemos las carpeta SRC (que contiene el main y test) y Target.

El main estara conformado en 3 grandes grupos de archivos o paquetes.



- Presentacion: Este paquete contendra clases que tenga que ver con la interaccion directa con una VISTA.

Una VISTA es un elemento html que usa el usuario para interactuar con el software.

Vamos a tener clases que tendran diferentes responsabilidades pero siempre tendra algo que ver con esa interaccion que tiene con el usuario, con los elementos del html.

Dentro de la presentacion estaran los Controladores.

Los CONTROLADORES tendran los ENDPOINTS, cada URL en donde se va hacer una peticion.

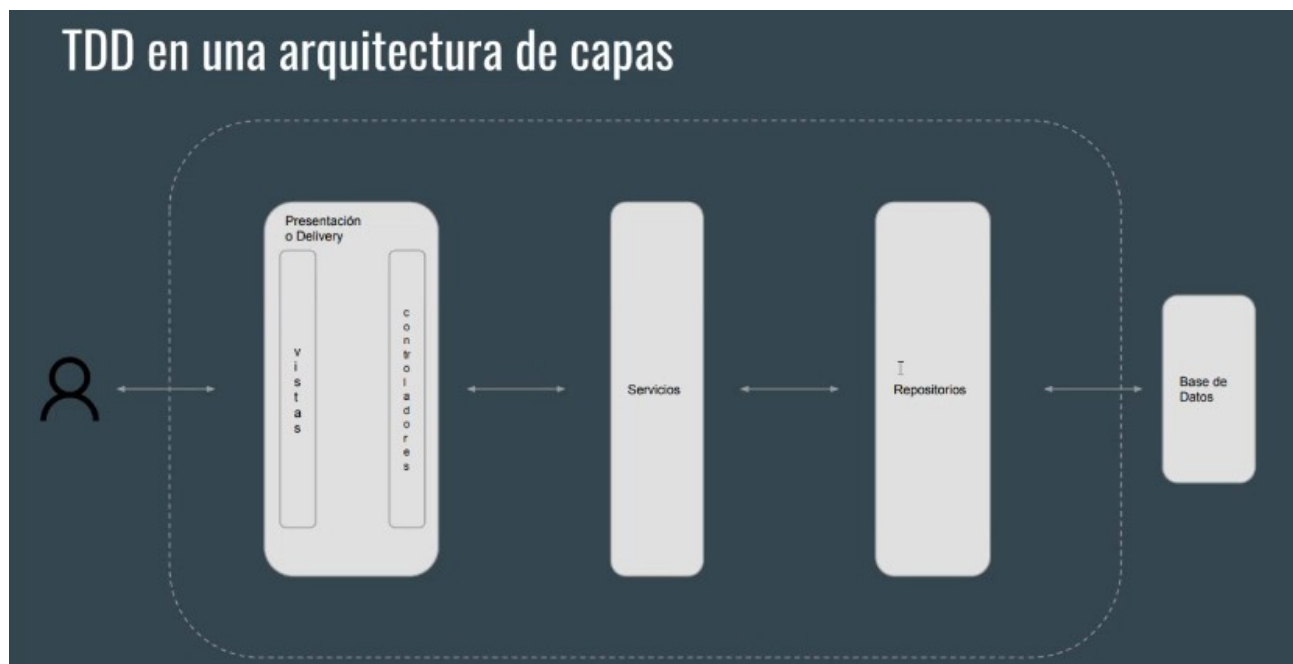
Hay servicios web en la cual podemos pedir alguna informacion y hacer diferentes operaciones en nuestro proyecto de esquema MVC

Cada archivos html que va a mostrar o pedir informacion al usuario va interacuar con ciertas clases a las cuales les llamaremos CONTROLADOR.

Cada controlador debería estar asociado con una entidad por ejemplo su nombre sería ControladorAuntenticacion o ControladorLogin si tiene que ver en registrarse, enviar datos y validar el usuario.

**No es correcto hacer un ControladorDarDeAlta donde estaran los metodos que dan de alta de todas las entidades o de Guardar y guarda animales, persona eso no sería lo correcto.**

## TDD en una Arquitectura de capas



En el paquete presentacion estan las Vistas y controladores. Estos 2 siempre se van a comunicar.

Con TDD vamos hacer muchas pruebas y lo haremos por capas

Vamos a tener pruebas en:

- la capa de presentacion,
- en la capa servicios /dominio,
- y en la capa de repositorio/ infraestructura

La forma de comunicarse entre capas es como indica las flechas:

El usuario interactua con las vistas y las vistas estan conjuncion con los controladores y cada uno de ellos puede interactuar con uno o varios servicios.

Esos servicios pueden hablar con uno o varios repositorios. Por ejemplo tenemos servicioUsuario que interactua con el RepositorioUsuario y tenemos el servicioAutenticacion y este puede tambien hablar con el repoUsuario no es necesario estrictamente crear un RepoAutenticacion.



CAPA DE PRESENTACION: sera la cara para el usuario en donde Spring con Thymeleaf tomara un archivo de plantilla html y junto con el codigo de java cumplir con la pantalla que se va a mostrar al usuario para que el usuario use

CAPA DE SERVICIOS: En la capa de servicios o de dominio. Dominio es todo lo que se puede controlar, todo lo referido del uso de ese software que estamos desarrollando. Es como decir yo tengo dominio de esto, lo conozco y se como hacerlo...

Por lo tanto en dominio o servicio **tenemos la forma de como se hace las cosas, las validaciones. O sea si existe reglas de negocio va en la capa Servicio.**

CAPA DE REPOSITARIOS: Aqui vamos a poner clases que tiene relacion a la comunicacion con la base de datos. Va haber una clase Repo para usuario donde exista codigo donde ejecuta una sentencia hacia la BBDD.

Va haber clases que se encarguen de comunicarse con la bbdd. Y esa clases va tener metodos.

EJEMPLO:

1. El usuario accede a nuestro sitio.
2. El sitio le muestra al usuario la VISTA, (un html)
3. El usuario interactua con esa vista y le dice me quiero DAR DE ALTA y le pide la vista de Registrarse
4. El sitio le muestra esa vista por medio de un metodo que tiene un endpoint y que sabe devolver eso el cual esta en el controlador
5. El usuario introduce sus datos en el formulario.
6. Hay otro metodo que esta en el controlador que sabe guardar el registro.

Esta interaccion no tuvimos que ir hacia la bbdd

7. El controlador pregunta al servicio si es valido los datos, o sea ahora el servicio se encarga de validar los datos ingresados por el usuario y si habia un campo calculado extra ese lo va hacer

8. Una vez que este todo listo los datos validados por el servicios, este se va comunicar con el repositorio para que este lo guarde en la bbdd.

9.El repositorio se hace cargo y lo guardda en la bbdd. Esto se llama las diferencias de responsabilidades.

PREGUNTA: Si el servicio se encarga de todo porque no hace directamente el insertar a la bbdd???

Es asi por la arquitectura que se plantea pero supongamos que hicimos una consulta compleja en SQL con 4 joins y la use en varios metodos de diferentes clases de Servicios y tuve un error al hacer las consultas entonces tendre que ir una por una a poder corregir en cambio si lo tengo en el repositorio solo me toca modificarla en el repo y no clase por clase del servicio que la esta usando.

Por ejemplo viendo el código de Taller Web base

El elemento login.html va interactuar con el controlador que es una clase .java y lo que va a pasar es que la vista trabaja con thymeleaf.

Thymeleaf se trabaja con th:estructura

aquí está indicando que este input está trabajando como un campo

```
<div id="loginbox" style="..." class="mainbox col-md-6 col-md-offset-3 col-sm-8 col-sm-offset-2">
  <form action="#" th:action="@{/validar-login}" method="POST" th:object="${datosLogin}">
    <h3 class="form-signin-heading">Taller Web I</h3>
    <hr class="colorgraph"><br>

    <input th:field="*{email}" id="email" type="email" class="form-control" required/>
    <input th:field="*{password}" type="password" id="password" class="form-control" required/>

    <button id="btn-login" class="btn btn-lg btn-primary btn-block" type="Submit">Login</button>

    <p th:if="${error != null}" class="alert alert-danger" th:text="'Error ' + ${error}">
    </p>
  </form>
  <a id="ir-a-registrarme" href="nuevo-usuario">Registrarme</a>
</div>
```

Thymeleaf va ayudar con la asociación de los elementos HTML con el contenido de Java, la vinculación entre el dato duro que está guardado en la BBDD y como mostrarlo y todo eso se muestra por pantalla.

## QUE ES THYMELEAF?

Thymeleaf nos va ayudar en la presentación, la presentación es lo que se muestra en el usuario.

Gracias a Thymeleaf tenemos en el FORMULARIO un objeto Java que está vinculado con el HTML y los inputs serán campos propios de ese objeto.

Cuando hacemos `th:field="*{email}"` estaríamos accediendo a la información es como hacer un `setEmail()` y un `setPassword()` automático.

Además de ayudarnos de vincular la información que traemos desde Java sino también nos provee estructura de control para hacer uso en el HTML como **if** y **for**.

No confundirse con el framework **SPRING MVC** vs **SPRINGBOOT**

## QUE HACE EL COMANDO mvn clean test?

El `mvn clean` lo que hace es hacer una limpieza de los archivos que se descargó anteriormente y los descarga de nuevo y corre los tests.

En la carpeta webapp → resources estaran los archivos css y .js que seran estaticos.

En la carpeta resources tendremos un archivo .sql que lo que hace es insertar un usuario en la tabla de Usuario.

Que pasa si queremos filtrar una busqueda?

En ese caso pasa por cada capas y en el repositorio hace la consulta sql WHERE pero en nuestro proyecto usaremos HIBERNATE.

El controlador sabe que el usuario quiere pelotas

El servicios sabe a quien preguntarle por las pelotas

El repositorio es el que filtra

Hibernate es un ORM, una herramienta que define un lenguaje para consultar a la bbdd propio neutral y tiene un conversor a la sentencia sql que nosotros hems especificado en la configuracion ya que se traduce segun el lenguaje sql que use sql server o mysql, por ejemplo sql server usa TOP en cambio mysql usa LIMIT.

Hibernate es como la clase MATH nos da metodos ya fabricados para consultar la bbdd

**La CLASE MyServletInitializer**

La comunicacion usando el framework Spring se da mediante SERVLET.

Uno levanta un servlet y disponibiliza ciertos archivos que tiene ciertos endpoints y se encarga de levantar archivos de config necesarios.

El modelo mas antiguo de Spring habia que crear un servlet por endpoint o sea un archivo por endpoint

```

1 package com.tallerweb1;
2
3 import com.tallerweb1.config.DatabaseInitializationConfig;
4 import com.tallerweb1.config.HibernateConfig;
5 import com.tallerweb1.config.SpringWebConfig;
6 import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;
7
8 public class MyServletInitializer {
9     extends AbstractAnnotationConfigDispatcherServletInitializer {
10
11         // services and data sources
12         @Override
13         protected Class<?>[] getRootConfigClasses() {
14             return new Class[]{};
15         }
16
17         // controller, view resolver, handler mapping
18         @Override
19         protected Class<?>[] getServletConfigClasses() {
20             return new Class[]{SpringWebConfig.class, HibernateConfig.class, DatabaseInitializationConfig.class};
21         }
22
23         @Override
24         protected String[] getServletMappings() {
25             return new String[]{"/"};
26         }
27     }
28 }

```

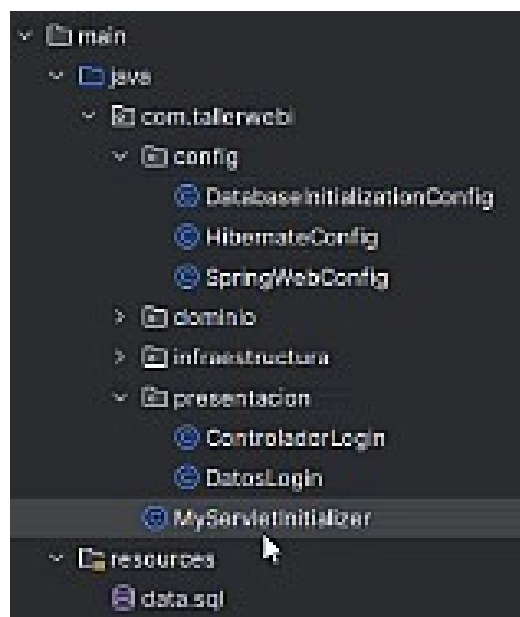
El pom es el que orquesta las dependencias de maven.

La clase MyServletInitializer es el que orquesta todo al resto

Tiene 2 metodos que son fundamentales para que funcione el proyecto

el getServletMapping() lo que dice es mapea todo lo que venga con venga luego de la / o sea la url que usamos en el controlador.

El metodo getServletConfigClasses() lo que hace es ejecutar las tres clases que retorna, las cuales son SpringWebConfig, HibernateConfig, y DatabaseInitializationConfig



El script SQL que esta en el data.sql se ejecuta gracias a la clase DataBaseInitializationConfig que es invocado desde **MyServletInitializer**  
Y es justo quien dice por medio de la clase populador que tendra que agregar el script que esta en data.sql

Si necesitamos escribir otro archivo sql tendremos que ir a tocar la clase DataBaseInitializationConfig

## CLASE HIBERNATECONFIG

Tiene el origen, la conexion a la base de datos

```
@Bean
public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName("org.hsqldb.jdbcDriver");
    dataSource.setUrl("jdbc:hsqldb:mem:db_");
    dataSource.setUsername("sa");
    dataSource.setPassword("");
    return dataSource;
}
```

Este metodo se encarga de escanear las clases que estan en el package dominio para levantar todas las entidades de las bases de datos

```
@Bean
public LocalSessionFactoryBean sessionFactory(DataSource dataSource) {
    LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
    sessionFactory.setDataSource(dataSource);
    sessionFactory.setPackagesToScan("com.tallerwebi.dominio");
    sessionFactory.setHibernateProperties(hibernateProperties());
    return sessionFactory;
}
```

pppp

```
private Properties hibernateProperties() { 1 usage 1 damian spizzirri
    Properties properties = new Properties();
    properties.setProperty("hibernate.dialect", "org.hibernate.dialect.HSQLDialect");
    properties.setProperty("hibernate.show_sql", "true");
    properties.setProperty("hibernate.format_sql", "true");
    properties.setProperty("hibernate.hbm2ddl.auto", "create");
    return properties;
}
```

hibernate.dialect: Aquí podemos cambiar el dialecto de acuerdo al sistema de gestión de la base de datos que elijamos.

El "hibernate.show\_sql, true" para ver en la consola los scripts sql de la bbdd

"hibernate.hbm2ddl.auto, create" es una propiedad que nos indica como tiene que comportarse la bbdd, cuando se levante el proyecto si la tiene que volver a crearla (create) o actualizarla (update)

## CLASE SpringWebConfig

esta clase tiene las configuraciones mas directas del Spring para que el proyecto funcione

```
@EnableWebMvc 1 damian spizzirri
@Configuration
@ComponentScan({"com.tallerwebi.presentacion", "com.tallerwebi.dominio", "com.tallerwebi.infraestructura"})
public class SpringWebConfig implements WebMvcConfigurer {
```

El @ComponentScan lo que hace es escanear, revisar las clases que tiene en los 3 paquetes presentacion, dominio e infraestructura

```
@Override 2 usages 1 damian spizzirri
public void addResourceHandlers(final ResourceHandlerRegistry registry) {
    registry.addResourceHandler(...pathPatterns: "/css/**").addResourceLocations("/resources/core/css/");
    registry.addResourceHandler(...pathPatterns: "/js/**").addResourceLocations("/resources/core/js/");
    registry.addResourceHandler(...pathPatterns: "/webjars/**").addResourceLocations("/webjars/");
}
```

Este metodo lo que hace es agregar recursos extras que son los recursos estaticos, aqui podemos agregar la url de la carpeta img

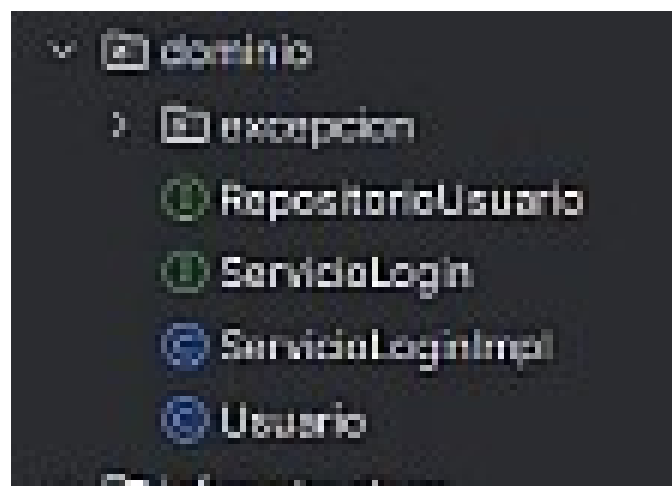
**no se que es la carpeta webjar**



```
// Spring + Thymeleaf
@Bean
public SpringResourceTemplateResolver templateResolver() {
    // SpringResourceTemplateResolver automatically integrates with Spring's own
    // resource resolution infrastructure, which is highly recommended.
    SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();
    templateResolver.setApplicationContext(this.applicationContext);
    templateResolver.setPrefix("/WEB-INF/views/thymeleaf/");
    templateResolver.setSuffix(".html");
    // HTML is the default value, added here for the sake of clarity.
    templateResolver.setTemplateMode(TemplateMode.HTML);
    // Template cache is true by default. Set to false if you want
    // templates to be automatically updated when modified.
    templateResolver.setCacheable(true);
    return templateResolver;
}
```

Este metodo templateResolver lo que hace es cuando yo tenga que mostrar por pantalla donde esta los html y este indicara al proyecto indicando el prefijo y sufijo de la url

## PACKAGE DOMINIO – CAPA DE SERVICIO



En este paquete esta Todo lo referido al conocimiento del negocio.

## CUALES SON LOS ARCHIVOS QUE VAN A ESTAR EN EL PAQUETE?

En el dominio vamos a tener interfaces y clases.

Las interfaces van a responder por los metodos que nosotros les vamos a disponibilizar al controlador.

```

public interface RepositorioUsuario { 5 usages 1 implementation 1 damian spizziri

    Usuario buscarUsuario(String email, String password); 2 usages 1 implementation 1 damian spizziri
    void guardar(Usuario usuario); 1 usage 1 implementation 1 damian spizziri
    Usuario buscar(String email); no usages 1 implementation 1 damian spizziri
    void modificar(Usuario usuario); no usages 1 implementation 1 damian spizziri
}

```

**En la puerta de entrada, imaginemos que viene el controlador con los datos en la mano y toca la puerta y le va atender la INTERFAZ Repositorio y le va a preguntar que quieres?**

Y el controlador le va a decir quiero guardar(Usuario user) dame un usuario

Y la interfaz va a conocer como se implementa o sea como se hace registrar o guardar.

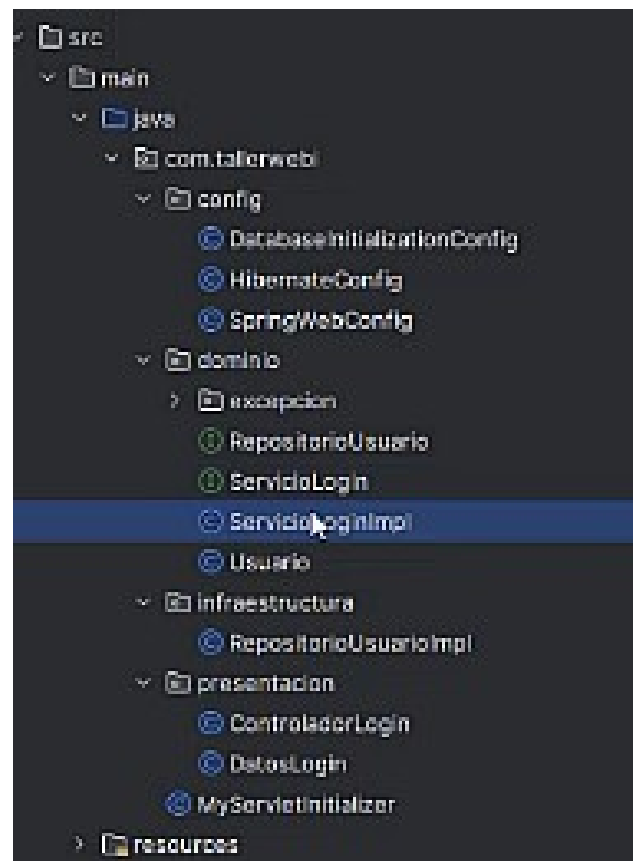
La interfaz es el listado de operaciones que nos va a dejar usar el servicio y vamos a contar con una clase que implemente esa interfaz.

El controlador trabaja con la interfaz ServicioLogin y no trabaja con la clase que lo implementa.

La interfaz me deja saber cuales son los metodos con las cuales otras clases pueda interactuar con ellas.

La clase x pregunta a la interfaz vos te encargas del login? Si entonces quiero consultarUsuario.

**Tengo una clase que implementa esa interfaz y escribo el metodo registrar de una manera especifica y esta bien pero puedo tener otra clase que implemente la misma interfaz y que registrar sea de otra forma.**



Por que la interfaz RepositorioUsuario esta en la carpeta dominio- servicio y no junto a la infraestructura?

El metodo registrar que esta en la clase ServicioLoginImpl va en la carpeta dominio porque se que hacer y como hacerlo

En cambio en la carpeta infraestructura va el RepositorioUsuarioImpl porque no me interesa tanto como lo guarda en la bbdd o interactua.

Si quiero saber si hay algun metodo especifico como buscar animales por mancha que tiene que tener mi sistema, el elemento que obliga que eso suceda es la interfaz RepositorioUsuario que tendra que estar en el dominio.

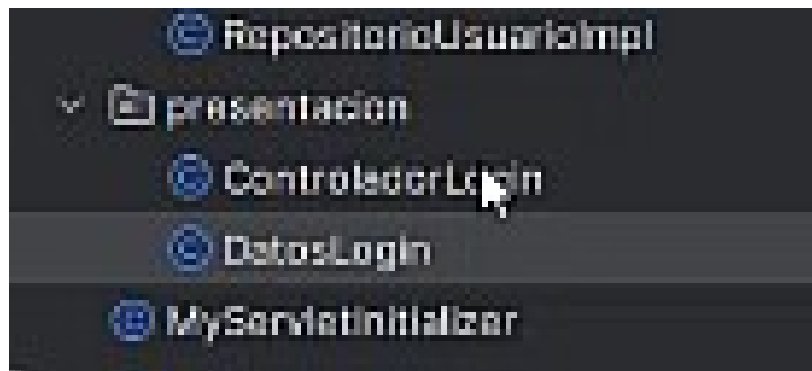
Si yo me llevo el paquete DOMINO solito a otro proyecto ese tiene que funcionar igualmente como una unidad

## DOMINIO

- Tiene lo que se necesita saber para ir a la bbdd. (RepositorioUsuario)
- Tiene lo que se necesita saber como regla de negocios (ServicioLogin)
- Tiene la implementacion de ese negocio (ServicioLoginImpl)
- Tiene una entidad de bbdd (Usuario)

El dominio es un paquete en si mismo cerrado de negocio

## PRESENTACION



En la carpeta PRESENTACION esta la **clase DatosLogin** que es similar a la entidad Usuario y que no vive en el dominio.

## Esta clase se la conoce como DTO (Data Transfer Object)

Seria como una clase de presentacion y su funcion principal es no exponer publicamente como hacer una entidad de dominio. Ya que en el dominio es como si estaria la receta como crear la coca cola. Nadie va querer compartir esto

**Por cada entidad que exista en nuestra BBDD y vaya viajar a un html deberia haber al menos 1 DTO.**

**Por cada vez que vaya manda informacion o esa informacion va interactuar con el html o con lo que sea que este procesando en otro lado**

### Que es ModelAndView?

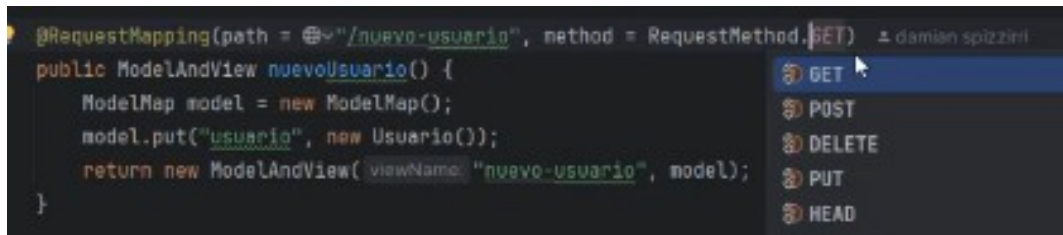
ModelAndView es un objeto que nos permite almacenar elementos, o sea podemos almacenar maps,

Los mapas estan compuesto por una clave y valor (valor puede ser un objeto, literal o string)

Entonces como objeto que se comunica con la vista donde le tenes que llevar informacion tiene la capacidad de hacerlo. A su vez conoce a la vista donde tiene que llevar esa info.

Es un objeto que se usa para vincularlos porq tiene todo el comportamiento y conocimiento de lo que tiene que hacer para interactuar con las vistas

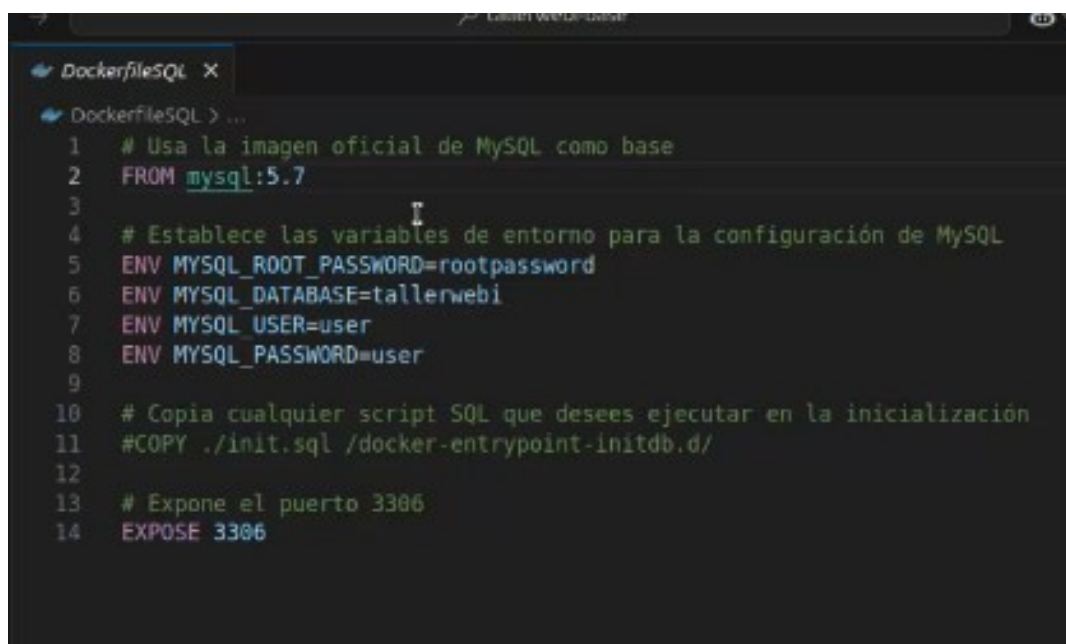
Que hace la anotacion @RequestMapping del metodo del controlador?  
Esa anotacion va a identificar a la url por nombre y por tipo de verbo http.



Todos los controladores se van a comunicar con uno o varios servicios.  
Por eso en la construccion del controlador tiene que conocer esos servicios.  
Asi que el funcionamiento de esta clase controlador depende de un servicio

---

## CLASE 29-05 REPASO DE DOCKER



### QUE ES DOCKER HUB?

Es un centralizador de repositorios de imagenes.

Hay persona que construye imagenes de base que nosotros lo podemos extender con mas herramientas o configuracion.

Lo que dice el archivito DockerfileSQL en **FROM mysql:5.7**  
de que ya existe esa imagen y q la va a usar.

Este archivo lo que dice toma esta imagen la que tiene configurado MYSQL version 5.7



Y la siguiente seccion es la parte de configuracion

```
# Establece las variables de entorno para la configuración de MySQL
ENV MYSQL_ROOT_PASSWORD=rootpassword
ENV MYSQL_DATABASE=tallerwebi
ENV MYSQL_USER=user
ENV MYSQL_PASSWORD=user|
```

A eso mysql 5.7 necesito que le configures ciertas variables de entornos

Luego expone el puerto 3306 para el mysql, tambien lo podemos cambiar por otro puerto. Por ej: el profe tiene el servicio mysql andando en el puerto 3306,y lo tendra que frenar porque tengo el mismo puerto.

-----  
Que es lo que se trae al hacer FROM mysql 5.7?



Viendo el dibujito de la ballena podemos pensar que lo que contiene arriba es como si fuesen herramientas.

Como si la ballena llevase un conjunto de herramientas, es como si en la pc te instalas ciertas herramientas como visual studio code, docker desktop. Como de ese estilos pero otra, muchos son servicios esas herramientas en lo general. Por ej: mysql es un servicio, jetty corre como un servicio no hay estrictamente una interfaz de usuario, muchas veces estan corriendo en SO de estilo SERVER. No tienen interfaz grafica.

Entonces practicamente lo que decimos en el dockerfile que traeme esto de herramienta y lo otro que lo voy a necesitar para trabajar.

Existe un dockerfile mas general que ya configuro una parte que uno ya la esta usando para crear otra imagen.



Por ejemplo para open JDK alguien hizo tal dockerfile, o sea ya existe algo y uno trabaja en base de esto. Respecto al lenguaje puede ser debian o ubuntu

```
dockerfile:jetty > ...  
# Usa la imagen base de OpenJDK 11  
FROM openjdk:11 I  
  
# Define una variable para la versión de Jetty  
ENV JETTY_VERSION=9.4.56.v20240826  
ENV JETTY_HOME=/opt/jetty  
  
# Descarga y extrae Jetty  
RUN apt-get update && apt-get install -y wget && \  
    mkdir -p $JETTY_HOME && \  
    wget https://repol.maven.org/maven2/org/eclipse/jetty/jetty-distribution/${JETTY_VERSION}/jetty-distribution-  
    tar -xzf jetty-distribution-${JETTY_VERSION}.tar.gz -C $JETTY_HOME --strip-components=1 && \  
    rm jetty-distribution-${JETTY_VERSION}.tar.gz  
  
# Copia el archivo WAR al directorio webapps de Jetty  
COPY target/tallerwebi-base-1.0-SNAPSHOT.war $JETTY_HOME/webapps/  
  
# Expone el puerto en el que Jetty corre (por defecto 8080)  
EXPOSE 8080  
  
# Me paro en el directorio de Jetty  
WORKDIR $JETTY_HOME
```

Entonces el dockerfile dice voy a usar tal herramienta OPENJDK 11 que viene con java 11.

Luego le agrega la variable de entorno.

Lo ejecuta e incluso hace un install inclusive

Ese dockerfile al final del día genera una imagen la cual nosotros construimos, cuando hacemos un BUILD.

A partir de esa imagen, que es como una fabriquita podremos generar varios contenedores.

QUE ES UN CONTENEDOR?

La imagen generada es como si fuese una clase y los contenedores son como si fuesen las instancias.

Podemos crear un contenedor o muchos incluso de distintos nombres.

Podemos inclusive no borrar las imagenes y darles nombres de versiones. Por ej: "La imagen de la entrega primer sprint nivel 1, La imagen de la entrega del 2sprint.

El contenedor es como un paquetito cerrado ya armado y va ser así siempre.

Salvo que cambie algo en el dockerfile, hacemos un UWARE que tiene datos actualizados porque tuvieron un incremento de producto y tenemos que eliminar esa imagen y usar otra y ahí crear un nuevo contenedor.

Nosotros podemos mantener una imagen con su contenedor y no estaría mal quizás sea en vano. Pero ya lo tenes

El contenedor termina siendo similar a una maquina virtual es lo que realmente se levanta y te da los servicios que uno estuvo configurando,

-----

UNA PREGUNTA.

Cuando uno tiene mucha concurrencia o sea nuestro sitio web es muy visitado por ahi se tendra que balancear carga.

QUE ES BALANCEADO DE CARGA?

Balanceo de carga es como tener imaginemos que la imagen docker generada sea, un servidor que escucha peticiones y tiene que tirar a uno o a otro. Entonces uno puede generar contenedores lo cuanto que necesites y despues si no lo queremos mas, lo podes boletear y de ese modo escalar rapido.

Tendria que agregar los posibles destinos al balanceador de carga.

En arquitectura mediana se usaba uno chiquito o mediano.  
Ya en arquitectura mas grande se usa 2 o mas contenedores.

Y tener muchos de esos contenedores atras muchas apis o una API que uno tenga que contestar.

De este estilo podemos levantar tanto contenedores que necesitemos y tu balanceador tenga que soportarlo y saber que existe esa api y asi mandarle las peticiones y vos me devolves la informacion.

Entonces en ese sentido podriamos tener mas de uno contenedor,

Mi API sera la misma solo que necesitare mas virtualizaciones con mas recursos para poder contestarles a todos.

Es como decir la fila del supermercado se hizo muy grande y necesito abrir mas cajas. .

Y cuando la fila se hizo chiquita y las cajas no tienen sentido, las cierran.

Tener mas contenedor implica tener mas recursos?

En si tenemos que ver que nuestra computadora que ejecuta eso tenga lo suficiente recursos para levantar la cantidad de contenedores que necesitamos...seria lo mas prudente. Esa parte de arquitectura, la tenemos que manejar, gestionarla acorde lo que estamos levantando.

Si uno usa recursos al tope, se podra trabajar? La gran mayoria trata de darle el 90% de uso con el espacio necesario para que todo funcione bien. Pero los explota bastante.

Lo que son productivo lo tienen a nivel alto de uso y acorde lo que tiene que resolver. Hay muchos trabajos con eso, hay varios test que te pueden ayudar de esa parte.

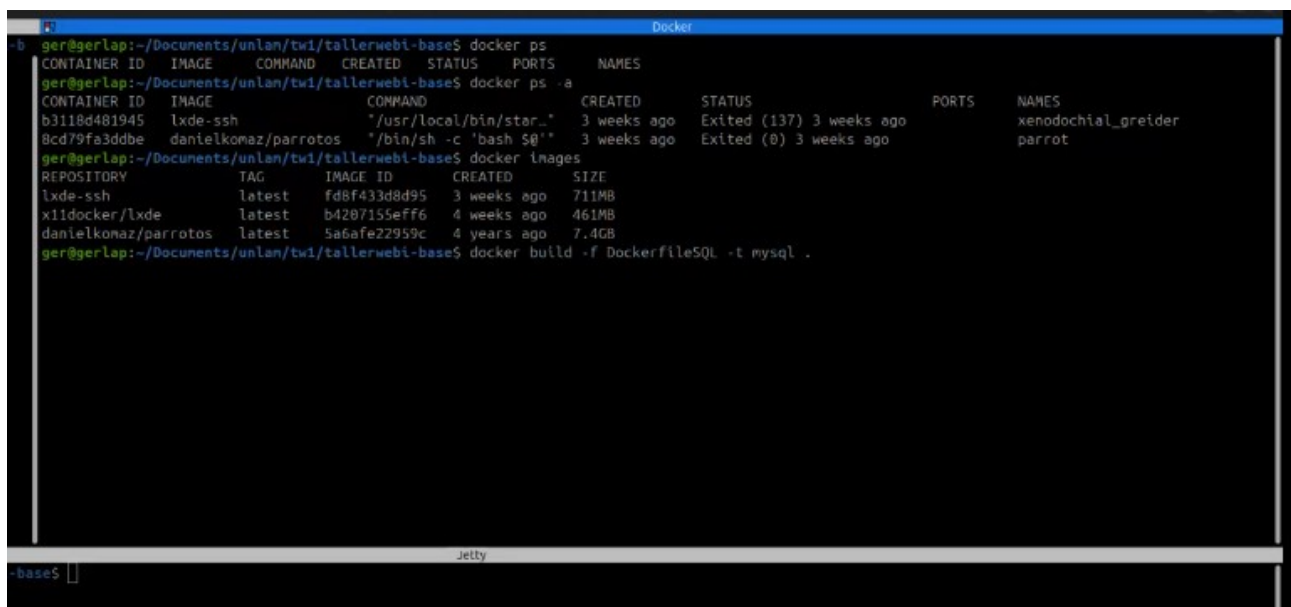
La gente de seguridad informatica hacen los test DE DENEGACION DE SERVICIOS (DOS) que te llenan de request o lo que sea hasta que se caiga. Entonces ahi encuentras el punto y desde ahi lo puedes medir cuanto hosting vas a necesitar para tanta ocurrencia.

Y despues estan las malas que se caigan.

En la peli de facebook habia una escena en donde se fijaban cuanto tardaba en caerse.

Hay herramientas que te permiten ensamblar dockerfile o sea podemos tener dockerfile repartidos en otros o uno muy grande.

### COMANDOS BASICOS DE DOCKER:



```
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
b3118d481945   lxde-ssh   "/usr/local/bin/star..." 3 weeks ago    Exited (137) 3 weeks ago          xenodochial_greider
8cd79fa3ddb6   danielkomaz/parrotos "/bin/sh -c 'bash $@"    3 weeks ago    Exited (0) 3 weeks ago          parrot
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
lxde-ssh       latest   fd8f433d8d95   3 weeks ago    711MB
x11docker/lxde latest   b4207155eff6   4 weeks ago    461MB
danielkomaz/parrotos latest   5a6afe22959c   4 years ago    7.4GB
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker build -f DockerfileSQL -t mysql .
```

**docker ps** : te muestra los contenedores que estan ejecutandose

**docker ps -a**: a (all) te muestra todo los contenedores existentes.

Usando este comando podemos saber cual existe para poder correrlo o frenarlo. Ademas es importante darle un nombre a los contenedores y no por defecto, un nombre distintivo y amigable para trabajar

**docker images**: vas a tener un listado de imagenes ya generadas

**docker rm [nombre\_contenedor]**: (remove) es para quitar el contenedor.

**docker rmi**: para quitar imagenes

Vamos a generar la IMAGEN DE BASE DE DATOS.

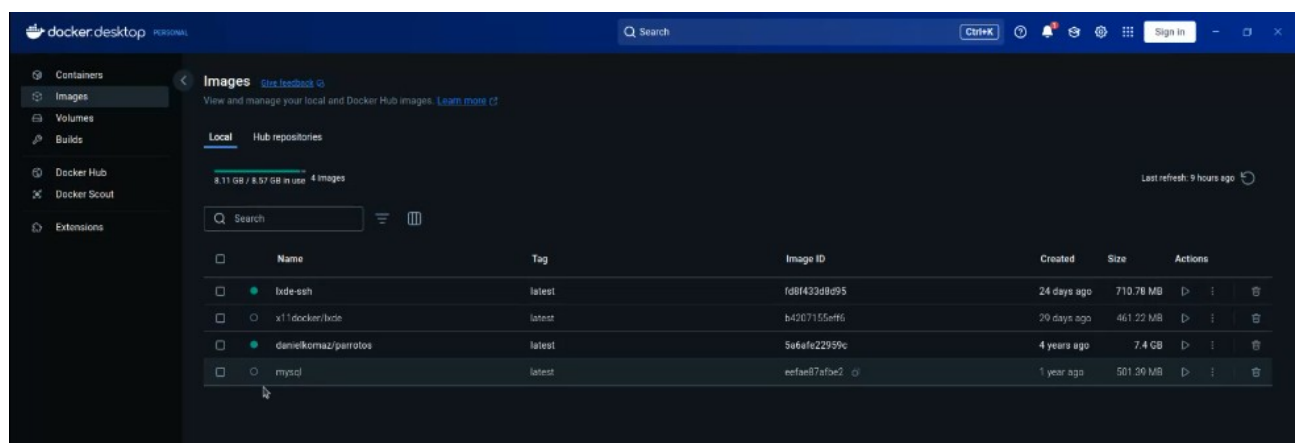
El README.md de nuestro proyecto podemos hacer:

***docker build -f DockerfileSQL -t mysql .***

***docker images:*** vere mi imagen mysql creada.

```
- SecretsUsedInArgOrEnv: Do not use ARG or ENV instructions for sensitive da
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
lxde-ssh             latest         fd8f433d8d95   3 weeks ago    711MB
xlidocker/lxde       latest         b4207155eff6   4 weeks ago    461MB
mysql                latest         eefae87afbe2   17 months ago  501MB
danielkonaz/parrotos latest         5a6afe22959c   4 years ago    7.4GB
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$
```

***Y ademas lo podemos ver la imagen creada en el dockerdesktop***



*Luego pediremos correr nuestra base de datos que es un contenedor haciendo lo siguiente*

```
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker run --name mysql-container -d -p 3306:3306 mysql
f4781080ce41664a766dc83e39271ffbdfdb2c98b0d45d191058d3876237dc34
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker ps
```

*Y este comando te da un hash y lo crear con el nombre mysql-container*

*Y lo podemos observar con docker ps*

```
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED      STATUS      PORTS                               NAMES
f4781080ce41   mysql    "docker-entrypoint.s..."  5 seconds ago Up 5 seconds 0.0.0.0:3306->3306/tcp, 33060/tcp  mysql-container
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ ^C
```

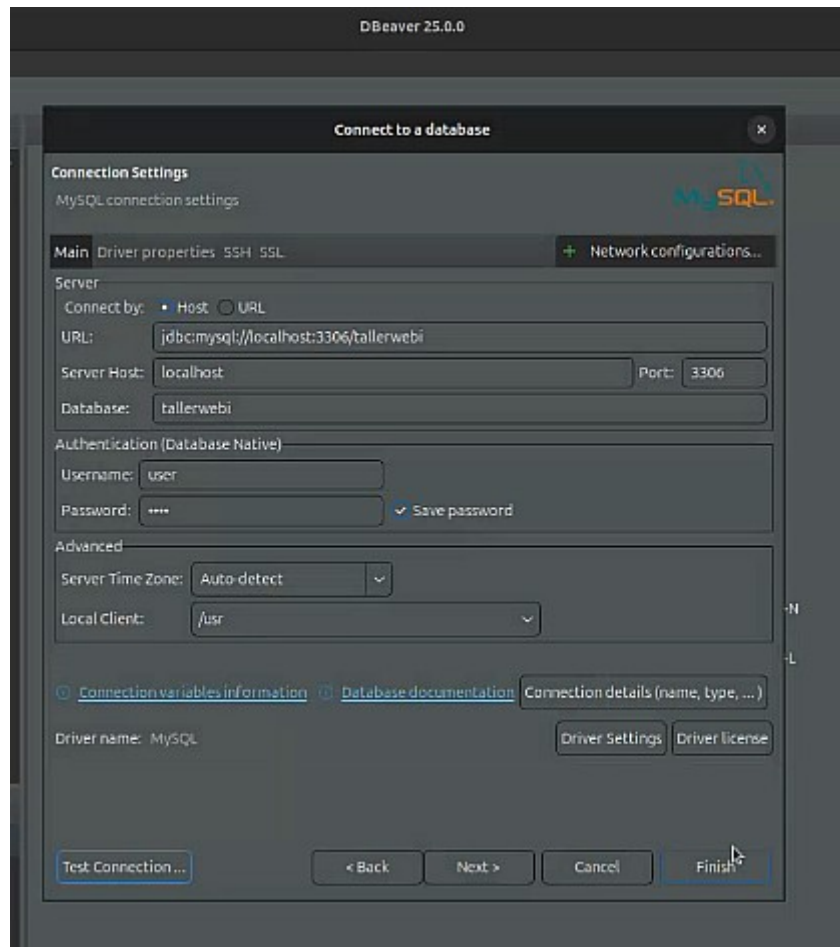
Podemos **parar** la ejecución del contenedor con **docker stop mysql-container**.  
Y para volverlo **ejecutarlo docker start mysql-container**

```
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
f4781080ce41   mysql    "docker-entrypoint.s..." 5 seconds ago  Up 5 seconds  0.0.0.0:3306->3306/tcp, 33060/tcp  mysql-container
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ ^C
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker stop mysql-container
mysql-container
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker start mysql-container
mysql-container
```

**Como puedo saber si esto esta funcionando a la hora de hacer start?**

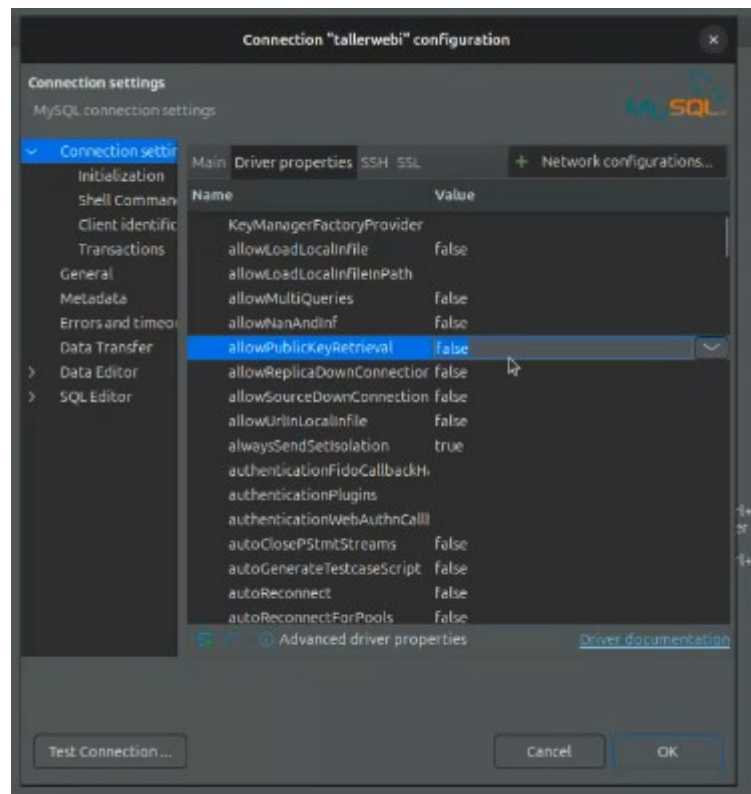
Podemos tener cualquier cliente como mysql workbench

En el caso del profe el tiene DBBiever y configura para conectarse a la bbdd



Puede ocurrir que con Dbeaver puede generar un leve error, en ese entonces tenemos que hacer:

- 1- Hacer clic derecho a la bbdd creada (tallerwebi)
- 2- Ingresar a edit connection
- 3- hacer clic a allowpublickeyretrieval



El **DockerFileJetty** de nuestro proyecto sabe levantar nuestro proyecto. Pero es el proyecto que tiene la configuracion para saber donde conectarse.

Entonces en el proyecto le damos esa configuracion. En cambio el DockerFileSQL solo sabe exponer el servicios de la bbdd.

El dockerfilejetty solo sabe exponer los servicios del proyecto.

Por lo tanto con estos dockerfiles vamos a tener 2 contenedores.

### COMO EJECUTAR PRUEBAS?

Vamos a tener pruebas que van a correr contra base de datos llamada HSQLDB en memoria para pruebas.



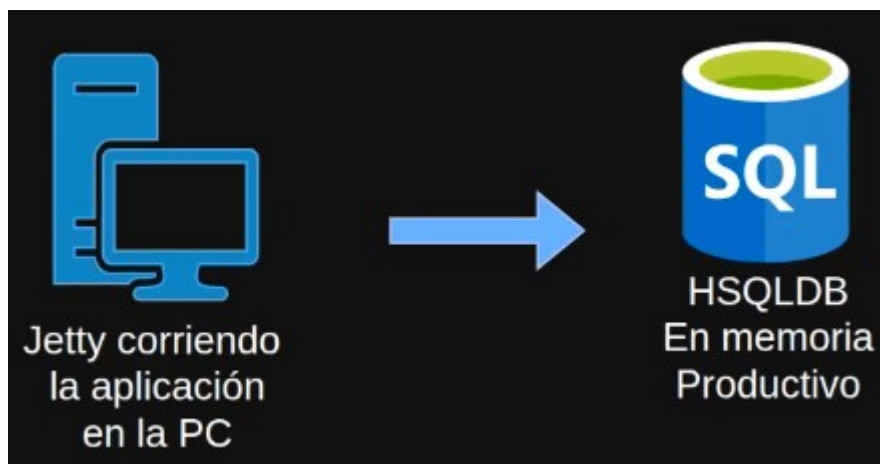
Y nuestro proyecto tambien estara corriendo sobre una bbdd HSQLDB, la cuestion es que esa base de datos actualmente esta corriendo en MEMORIA.

Que significa que corra en memoria?

Significa que se levanta un servicio que basicamente pone objetos en la memoria RAM y lo podemos usar desde ahi hasta que lo bajemos, es decir que no tiene soporte fisico. En

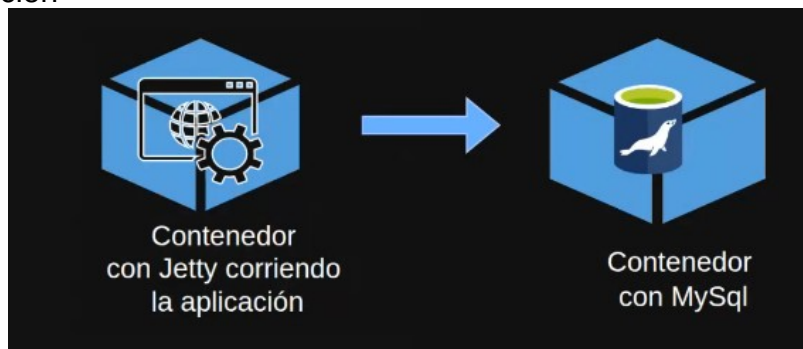


cambio MySQL y PostgreSQL dentro de sus características tiene la posibilidad de persistir información durante el tiempo.



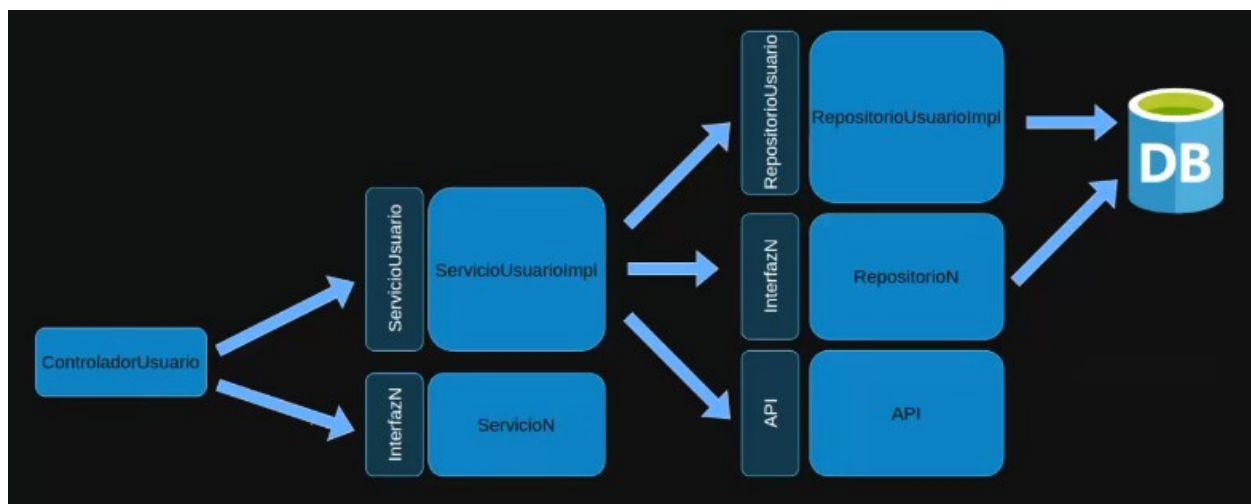
El HSQLDB nos sirve para correr nuestros test, pero para nuestro proyecto no tanto

Por lo cual nosotros vamos a crear contenedores para que funcione nuestro proyecto como en producción



Vamos a generar un contenedor que va a contener nuestra bbdd y otro contenedor que va a servir a nuestro sitio web. Este último ya va a conocer la url de conexión.

Repaso de MVC, mockear



El Controlador se comunica con el SERVICIO.

Un controlador puede tener uno o muchos servicios. El controlador va poder usar lo que necesite. Puede pasar que se necesite mas de un servicio, y lo que podemos hacer es enviar por parametros los servicios separado con una coma en el constructor lo que vamos a usar.

Pero en realidad lo que siempre deberiamos tener marcado en la mente es la relacion que existe entre controlador y la interfaz servicio de porque lo que haga la implementacion de por fuera para resolver esos metodos de la interfaz.

El alcance de este controlador que incluye metodos del servicio porque pide dame el dni tanto, el nombre tanto.

El ServicioUsuarioImplementacion va a tener metodos obtenerPorId(id), guardar(usuario). Los cuales seran usado por el controlador.

El controlador va tener su propio metodo buscarPorId() el cual va invocar al metodo obtenerPorId(id) del servicio. y ese tiene que devolver algo al metodo del controlador por lo cual necesitara comunicarse con el repositorio.

El servicio se puede comunicar con uno o muchos servicios

Y el servicio se comunica con uno o muchos repositorio.

Y el alcance del servicio va ser hasta la interfaz del repositorio.

Porque sigue con la misma cuestion, porque necesita saber como son los metodos que va usar para obtenerUsuario y la Interfaz del repositorio se lo puede decir. el como lo podra resolver sin importar la BBDD que estemos usando.

El servicio sabe que necesita consumir los metodos del repositorio, lo necesita.

En resumen, cualquier clase que este dentro de la capa de servicios va tener metodos y dentro de los mismos van invocar los metodos del repositorio.

Por ej esto:

```
@Override
public Usuario RepositorioUsuario repositorioUsuario (String password) {
    return repositorioUsuario.buscarUsuario(email, password);
}
```

Y lo que nos va a tocar mockear es la respuesta de los metodos que no es de nuestra capa.

Es decir.

Si estoy testeando el controlador. Tendre que mockear la respuesta de los metodos del servicio o sea cuando yo consuma buscarPorId(), el metodo obtenerPorId() del servicio lo unico que tiene que decir que es lo que tiene que devolver.

Para los servicios va pasar lo mismo con el repositorio. Porque tambien va consumir metodos que no son de ellos.

### Que pasa si el metodo devuelve un void?

Si el metodo void, hace uso de un metodo que no es propio de esa clase servicio, o sea hace uso de un metodo del repositorio. El metodo del repositorio lo tendre que mockear. Hay una forma de decirle al test que verifique que si se invoco el metodo con VERIFY.

Por ejemplo si estoy guardando un usuario y manejo excepciones para casos especiales y que me de una respuesta quizás no me interesa.  
Y uno lo hace void. Por lo menos verificamos con VERIFY y times que fue invocado minimamente.

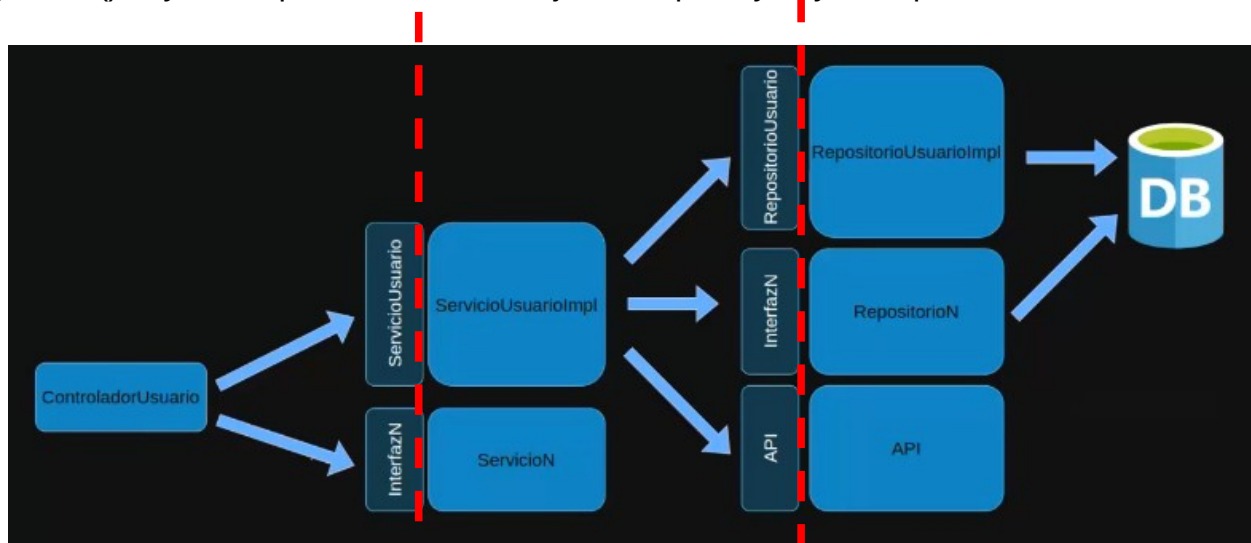
Si ese método realiza algo importante como una ACTUALIZACIÓN donde algo tiene que quedar muy puntual es testearlo.

Por ej este método habría que hacer 3 test unitarios, un test cuando entra en el if y el otro cuando no entra y otro si fue invocado o no el método del repo.

```
@Override
public void registrar(Usuario usuario) throws UsuarioExistente {
    Usuario usuarioEncontrado = repositorioUsuario.buscarUsuario(usuario.getEmail(), usuario.getPassword())
    if(usuarioEncontrado != null){
        throw new UsuarioExistente();
    }
    repositorioUsuario.guardar(usuario);
}
```

Mockito también te da la posibilidad de probar métodos PRIVADOS.

El caso del método guardar() del repo hay ORM que te devuelve el id del objeto al hacer guardar() hay otros q te devuelven el objeto completo y hay otro q no te devuelve nada



**CAPA 1:**  
Vamos a probar los métodos del controlador y a mockear los métodos del servicios

**CAPA 2:** Vamos a probar  
Los métodos de servicios pero mockear los métodos de repo.

Supongamos tengo una api y que esa no me conteste con la información.

Que hace mi servicio cuando no responde la api. Eso lo mockearemos

**CAPA 3:** Que pasara con el repositorio?  
No mockeamos nada.  
Aquí las pruebas siempre se tendrá que correrse en HSQLDB de memoria xq la levanta para correr las pruebas. Y cuando ya no la necesita más la baja. No hay posibilidad de que quede información de basura. Porque no se persiste

QUE PASA SI TENEMOS UN AMBIENTE DE PRUEBA, Ahi si usamos la bbdd?  
No, porque quedaria informacion basura de otras pruebas. Aunque hay una anotacion llamada `@Rollback` que significa que le decimos ejecuta la prueba y desace todo lo que hicimos.

De hecho deberia estar asi ejecucion de pruebas con HSQLDB xq en la configuracion de Github action tambien la ejecuta, y las corre en la bbdd en memoria.

Es decir en Github action tambien se usa una maquina que tiene un Ubuntu y tambien se ejecuta ***mvn clean test*** y esos test van a correr en esa db en memoria

Como es un metodo de infraestructura?

```
@Override
public Usuario buscarUsuario(String email, String password) {

    final Session session = sessionFactory.getCurrentSession();
    return (Usuario) session.createCriteria(Usuario.class)
        .add(Restrictions.eq("email", email))
        .add(Restrictions.eq("password", password))
        .uniqueResult();
}
```

SessionFactory es una fabrica de sesiones y tiene algo similar a una sentencia

-----

Si vamos a la pom.xml

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <jdk.version>11</jdk.version>
  <spring.version>5.2.22.RELEASE</spring.version>
  <servletapi.version>4.0.4</servletapi.version>
  <thymeleaf.spring.version>3.0.15.RELEASE</thymeleaf.spring.version>
  <webjars.version>5.2.0</webjars.version>
  <hamcrest.version>2.2</hamcrest.version>
  <junit.version>5.9.0</junit.version>
  <mockito.version>5.3.1</mockito.version>
  <hibernate.version>5.4.24.Final</hibernate.version>
  <playwright.version>1.36.0</playwright.version>
  <hsqldb.version>2.3.2</hsqldb.version>
  <mysql.version>8.0.33</mysql.version>
</properties>
```

aqui le decimos que version de mysql se usara en el proyecto spring mvc. Respecto a Hibernate vamos a trabajar con una version obsoleta

tambien vemos las dependencias que se va a descargar para hacer uso

```
<dependency>
  <groupId>org.hsqldb</groupId>
  <artifactId>hsqldb</artifactId>
  <version>${hsqldb.version}</version>
</dependency>
```

Despues tenemos el conector para mysql que es la siguiente dependencias con la version que fue indicada en el properties. Ese sera el conector necesario para poder comunicarse con la bbdd. Es el conector que conecta con mysql. Y el tipo de conector dependera el tipo de Gestor de bbdd que usaremos, si es postgres, mysql. En nuestro caso usaremos el conector version 8 de mysql esta en el properties.

```
<!-- Conector para MySQL -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>${mysql.version}</version>
  <scope>runtime</scope>
</dependency>
```

Y en nuestro contenedor Docker esta corriendo la version del servicio 5.7. mysql  
El conector de nuestro pom.xml como es una version mas actualizada soporta versiones antiguas como el de nuestro servicio.

### Y como configuramos el conector mysql con el servicio mysql del docker mas hibernate?

Dentro de la carpeta CONFIG hay un archivito llamado HibernateConfig.

```
@Configuration
@EnableTransactionManagement
public class HibernateConfig {
```

Esta clase HibernateConfig la marcamos como un archivo de @Configuration para indicarle a spring que es de suma importancia y la 2da anotacion habilita el manejo de transacciones que significa poder realizar todo o nada.

Despues define ciertos metodos con una anotacion @Bean. Un bean significa como un recurso de hibernate es como algo que se disponibiliza.

```
@Bean
DataSource com.tallerwebi.config.HibernateConfig.dataSource() {
  public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName(driverClassName:"org.hsqldb.jdbcDriver");
    // dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
    dataSource.setUrl(url:"jdbc:hsqldb:mem:db_");
    // dataSource.setUrl("jdbc:mysql://localhost:3306/tallerwebi");
    dataSource.setUsername(username:"sa");
    dataSource.setPassword(password:"");
    return dataSource;
  }
}
```

El metodo dataSource() va tener la configuracion hacia la bbdd y podemos apreciar que tiene un driver el cual esta habla del conector y por defecto tiene la db en memoria por lo cual el proyecto anda.

Lo que esta comentado es jdbc lo que permite a Java conectarse con MySQL .

Luego esta el setUrl() que me dice donde encuentro la bbdd por defecto el proyecto tiene HSQLDB pero lo que esta comentado es la url o similurl ya que LOCALHOST es como el alias de la direccion ip 127.0.0.1 como un DNS amigable.



```
@Bean
public LocalSessionFactoryBean sessionFactory(DataSource dataSource) {
    LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
    sessionFactory.setDataSource(dataSource);
    sessionFactory.setPackagesToScan(...packagesToScan:"com.tallerwebi.dominio");
    sessionFactory.setHibernateProperties(hibernateProperties());
    return sessionFactory;
}
```

El sessionFactory va ser el encargado de levantar nuestro modelo, generar la conexión y disponibilizarla. Este es el tipo que pregunta que quieres hacer? Quieres guardar? Quieres obtener? Quieres eliminar?

Por lo cual el sessionFactory tiene que conocer el origen de la conexión por eso en la siguiente línea hace `sessionFactory.setDataSource(datasource)`

Y le tenemos que decir cual es el package el paquete que tiene que revisar para encontrar las entidades, clases para la bbdd.

Hibernate tiene un vínculo muy fuerte con la bbdd.

Es un ORM, es decir el encargado de mapear clases de java con tabla de bbdd.

Y tiene 3 formas de hacerlo:

- 1) Escribir clases y que se generen tablas
- 2) Tener una db y apartir de esa que se genere las clases
- 3) Hibrido

Nosotros mayormente usamos la 1era forma o sea del mapeo la asociación, Y que estrategias hay para resolver problemas conocidos

La siguiente línea el método `setHibernateProperties()` tiene probabilidad de configuración sobre la conexión.

El `transactionManager()` controla toda la parte de transacción.

```
private Properties hibernateProperties() {
    Properties properties = new Properties();
    properties.setProperty("hibernate.dialect", "org.hibernate.dialect.HSQLDialect");
    // properties.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQL57Dialect");
    properties.setProperty("hibernate.show_sql", "true");
    properties.setProperty("hibernate.format_sql", "true");
    properties.setProperty("hibernate.hbm2ddl.auto", "create");
    return properties;
}
```

El dialecto, que es?

Es como nuestro traductor

```
@Override
public Usuario buscarUsuario(String email, String password) {

    final Session session = sessionFactory.getCurrentSession();
    return (Usuario) session.createCriteria(Usuario.class)
        .add(Restrictions.eq("email", email))
        .add(Restrictions.eq("password", password))
        .uniqueResult();
}
```



En este metodo vimos que hay algo parecido a una sentencia y lo que hace el dialecto es todo eso que hago para x bbdd sera asi. Ya que hay palabra reservada para mysql que es LIMIT en cambio para sql server es TOP.

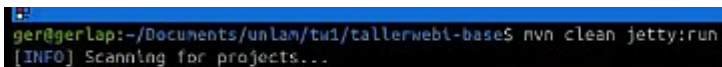
Entonces cuando usemos una bbdd el DIALECT lo que va decir esto add(Restriccion.eq) .... va funcionar de tal forma para tal bbdd

```
private Properties hibernateProperties() {  
    Properties properties = new Properties();  
    properties.setProperty("hibernate.dialect", "org.hibernate.dialect.HSQLDialect");  
    // properties.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQL57Dialect");  
    properties.setProperty("hibernate.show_sql", "true");  
    properties.setProperty("hibernate.format_sql", "true");  
    properties.setProperty("hibernate.hbm2ddl.auto", "create");  
    return properties;  
}
```

**CREATE:** El hibernate.hbm2ddl.auto, create: significa que si dejamos esta configuracion asi no mas cada vez que se levante SPRING, spring cuando tenga que comunicarse con la bbdd la va crear nueva, o sea hace drop si ya la habia creado y luego nuevamente la crea

**UPDATE:** Puedo crear un paquetes create para la presentacion darle informacion inicial a la bbdd.

Mientras estamos desarrollando podemos trabajar con update.



```
ger@gerlap:~/Documents/unlan/tw1/tallerweb1-base$ mvn clean jetty:run  
[INFO] Scanning for projects...
```

Y podemos ver que es lo que se ejecuto por la terminal

Luego podemos probar usando mysql

```

@EnableTransactionManagement
public class HibernateConfig {

    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        // dataSource.setDriverClassName("org.hsqldb.jdbcDriver");
        dataSource.setDriverClassName(driverClassName:"com.mysql.cj.jdbc.Driver");
        // dataSource.setUrl("jdbc:hsqldb:mem:db_");
        dataSource.setUrl(url:"jdbc:mysql://localhost:3306/tallerwebi");
        dataSource.setUsername(username:"user");
        dataSource.setPassword(password:"user");
        return dataSource;
    }

    @Bean
    public LocalSessionFactoryBean sessionFactory(DataSource dataSource) {
        LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
        sessionFactory.setDataSource(dataSource);
        sessionFactory.setPackagesToScan(...packagesToScan:"com.tallerwebi.dominio");
        sessionFactory.setHibernateProperties(hibernateProperties());
        return sessionFactory;
    }

    @Bean
    public HibernateTransactionManager transactionManager() {
        return new HibernateTransactionManager(sessionFactory(dataSource()).getObject());
    }

    private Properties hibernateProperties() {
        Properties properties = new Properties();
        // properties.setProperty("hibernate.dialect", "org.hibernate.dialect.HSQLDialect");
        properties.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQL57Dialect");
        properties.setProperty("hibernate.show_sql", "true");
        properties.setProperty("hibernate.format_sql", "true");
        properties.setProperty("hibernate.hbm2ddl.auto", "update");
        return properties;
    }
}

```

Para ejecutar nuestro proyecto y probarlo con mysql posta tendremos que usar docker y jetty como vimos anteriormente

The screenshot shows a terminal window with two main sections: Docker and Jetty.

**Docker Section:**

```

ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
f4781080ce41   mysql    "docker-entrypoint.s..." 42 seconds ago Up 5 seconds  0.0.0.0:3306->3306/tcp, 33060/tcp   mysql-container

ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
f4781080ce41   mysql    "docker-entrypoint.s..." 4 minutes ago  Up 3 minutes  0.0.0.0:3306->3306/tcp, 33060/tcp   mysql-container

ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker stop mysql-container
mysql-container
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ docker start mysql-container
mysql-container
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$

```

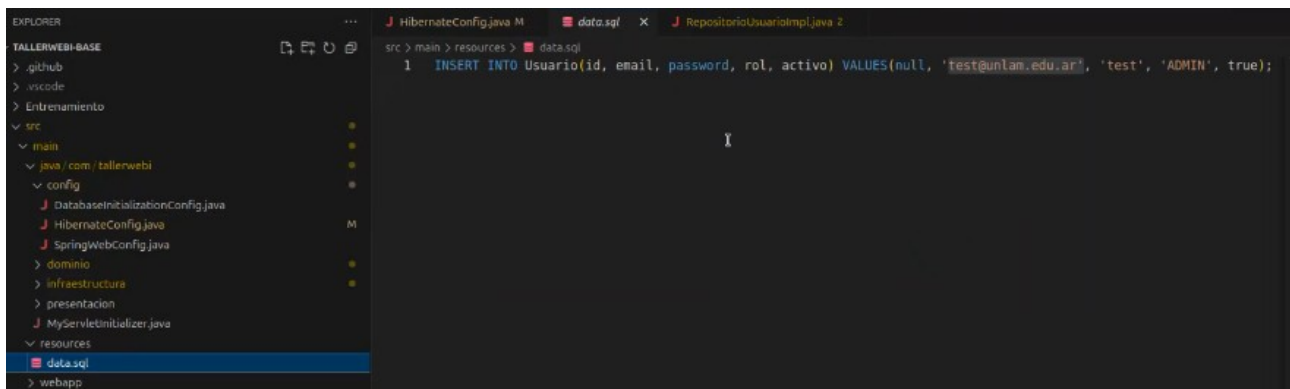
**Jetty Section:**

```

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:05 min
[INFO] Finished at: 2025-05-29T20:39:33-03:00
[INFO] -----
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ ^C
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ ^C
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ ^C
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ ^C
ger@gerlap:~/Documents/unlan/tw1/tallerwebi-base$ mvn clean jetty:run
[INFO] Scanning for projects...
[INFO] -----< com.tallerwebi:tallerwebi-base >-----
[INFO] Building spring web mvc 1.0-SNAPSHOT
[INFO] from pom.xml

```

Al hacer las pruebas de cambiar create, update podemos ver que se duplica los datos ya que en nuestro proyecto estamos ejecutando un script en resources → data.sql, una sentencia o script sql y no hace drop a la tabla vinculada que ya existe



Este data.sql es un script que se ejecuta todo el tiempo por lo tanto cada vez que levantamos el proyecto insertara el mismo dato. Podemos agregar info lo que queramos ya que este puede funcionar como base para mis pruebas.

### Y este data.sql como esta siendo ejecutado?

Esta siendo ejecutado por la clase DatabaseInitializationConfig{} y basicamente lo que hace es leer ese script.

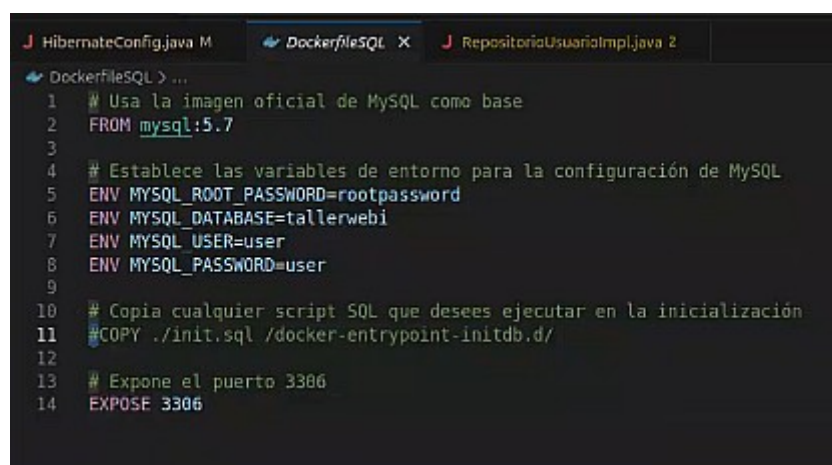
Y basicamente lo que hace es leer ese script y conoce el datasource y lo ejecuta con el metodo setDataBasePopulator().

El profe dario hablo sobre sql inyeccion y German dice que hibernate igualmente tiene una capa de seguridad que protege de sql inyeccion.

Los ORM agrega una buena capa seguridad por eso tambien esta bueno que se agrega soporte y actualizaciones de las versiones porque siempre va haber alguien que va arreglarlo y parcheando.

Esta bueno porque ademas de sanitizar las entradas, revisemos el origen de la informacion. Por ej: laravel tiene enloquient, otros entity framework.

### ----- EN EL DOCKERFILESQL



Tenemos la linea 11 **COPY ./init.sql/docker-entrypoint-initdb.d** la cual dice donde estamos parado copia el archivo init.sql en la carpeta docker-entrypoint...para ejecutarlo al inicio.

Hace la misma accion que el DataInitialization solo que aqui lo hace DOCKER

AHORA VIENDO ENTIDADES....



```
hibernateConfig.java M DockerfileSQL Usuario.java X J Rep
> main > java > com > tallerwebi > dominio > Usuario.java > Usuario
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String email;
    private String password;
    private String rol;
    private Boolean activo = false;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getRol() {
        return rol;
    }

    public void setRol(String rol) {
        this.rol = rol;
    }

    public Boolean getActivo() {
        return activo;
    }
}
```

Que es lo que convierte una clase cualquiera a una entidad? La anotacion @Entity. Esta es la que va decir a hibernate que justo esta en la carpeta de dominio y que busque todas las clases que lleve esa anotacion.

Estan obligadas a tener un ID y se marca de esta forma @Id este es el decorador o anotacion que nos marca la parte pk de la tabla de bbdd

Despues esta la STRATEGY.IDENTITY que lo que hace es indicar que el id va ser autoincremental y es la mas usada de las estrategias.

Todas las clases que usan para formar las base de datos tienen que tener @Entity. Si no lo ponemos tendremos una excepcion porque va intentar levantar toda la configuracion y va encontrar que una parte no era, lo mismo pasa con el atributo. Le podemos decir a hibernate que este atributo no lo mapee.

DATO CURIOSO: **git diff pom.xml** : sirve para ver las diferencias que estamos subiendo al repo como ver el antes y despues antes de hacer **git add** .

## PRACTICA

```
public class ControladorCarta {  
    private ServicioCartaImpl servicioCarta;  
    public ControladorCarta(ServicioCartaImpl servicioCarta) {  
        this.servicioCarta = servicioCarta;  
    }  
    public ModelAndView crearCarta(CartaDto carta) {  
        ModelMap modelMap = new ModelMap();  
        Boolean creada = this.servicioCarta.crear(carta);  
        String mensaje = "Error al crear carta";  
        if(creada) {  
            mensaje = "Carta creada correctamente";  
        }  
        modelMap.put("mensaje", mensaje);  
        modelMap.put("carta", carta);  
        return new ModelAndView(viewName:"crear-carta", modelMap);  
    }  
}
```

### Cual es la ventaja de usar una interfaz como servicioCarta?

Que quizás no tenga implementación aun, no se como hace tal cosa? No se como es el cuerpo? Pero se que van a estar los metodos, se que voy a tener operaciones que voy a poder consumir de otra clase, invocar y eso la interfaz lo puede decir.

Cuando hacemos uso de una clase implementación estamos saltando una parte del SOLID.

El principio de SOLID te invita tener siempre una interfaz que responda por las operaciones publicas de la clase implementación.

Uno solo debería saber el contrato...

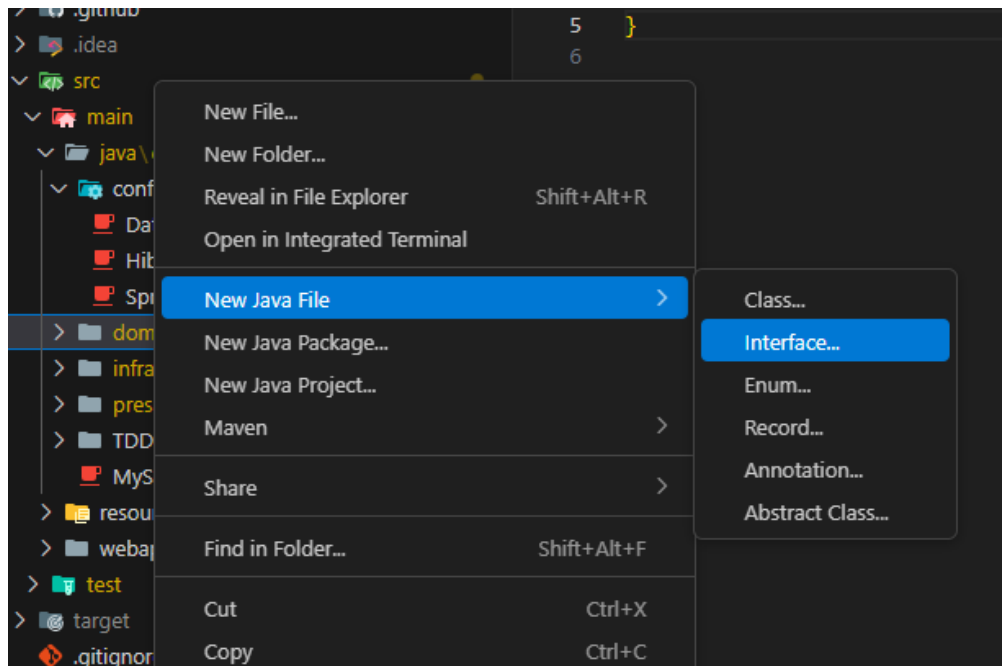
Es como ver el control remoto. Y decir che que operaciones puedo usar? Y sepan que hay botones pero como lo resuelven no lo sabemos hasta que presionemos el boton.

ENTONCES UNA DE LAS VENTAJAS DE USAR INTERFAZ.

Es que yo podría trabajar tranquilamente con el test del controlador sin haber hecho la implementación de servicio, o sea con tan solo saber que operaciones tenemos me alcanza para testear xq yo puedo mockear esto del servicio y me va mockear el metodo crearCarta() y le puedo decir a mockito que es lo que puede devolver

DATO CURIOSO:

Esta la opción new java file me permite crear clases e interfaces en VSC.



Se puede crear carta pero deben tener nombre. Sin nombre no es posible crear una carta.

Entonces tenemos dos escenarios validos que son: TieneNombre o NoTieneNombre en el uso de tu sistema.

Si tiene nombre que me la cree, si no lo tiene que no me la cree

```
public class ControladorCarta {
    private ServicioCarta servicioCarta;
    public ControladorCarta(ServicioCarta servicioCarta2) {
        this.servicioCarta = servicioCarta2;
    }
    public ModelAndView crearCarta(CartaDto carta) {
        ModelMap modelMap = new ModelMap();
        Boolean creada = this.servicioCarta.crear(carta);
        String mensaje = "Error al crear la carta";
        if (creada) {
            mensaje = "Carta creada correctamente";
        }
        modelMap.put("mensaje", mensaje);
        modelMap.put("carta", carta);
        return new ModelAndView("crear-carta", modelMap);
    }
}
```

```
modelMap.put("carta", carta);
```

Si a mi modelMap en un metodo get le doy un objeto no vacio llenara mis input con esos valores. Ya que esos datos los vincula.

Si tiene el metodo de mi controlador try catch, por cada catch agregarle un test por cada catch.

**Todo lo que sea externo se mockea.**



En el test de integracion ahi si usaremos implementacion, porque probaremos la integracion del todo. En cambio en los test unitarios solo lo especifico el controlador si fuese el controlador, o el servicio solo, asi con el resto y se mockea lo externo.

El profe pregunta

Esta el grupo que elige que el servicio le llegue una cartaDto o le llegue una entidad

En que se diferencia DTO vs Entidad

Se diferencia en que El DTO no va a la bbdd. La entidad sirve para guardar la informacion que va a la bbdd y el dto me sirve para mostrar deternimanda informacion q yo quiera mostrar. El dto no se guarda a la bbdd.

El DTO significa DATA TRANSFER OBJECT es muy habitual de ver que se construya a partir de la entidad carta y tambien poder volver a partir de una entidad generar un dto.

El dto le va a llevar al controlador, con los inputs que se maneja con la vista. Quizas no tenga que mostrarte toda la informacion de la entidad.

**LO QUE ENTENDI**, no se si esta bien. En caso de mostrar los datos por ejemplo para editar con el request UPDATE es conveniente traer la info de entidad y setearla al dto para mostrar. Y no confiar en lo que obtengo al guardar en hibernate.

**Hibernate por ejemplo si tengo una coleccion asociada a un objeto te inserta los registros relacionados. Si vos lo sacaste puede ser que lo elimine**

```
@Override
public Boolean crear(CartaDto carta) {
    I
    Carta carta = this.repo.obtener(1);
    // return !carta.getNombre().isEmpty()
    return null;
}
```

**Es bueno tener un dto por cada entidad que necesitemos?** Si, de hecho podrias tener un dto que no se llame igual a la entidad pero que sea un dto que se devuelva **y pueda estar formado por varias entidades**.

**Al final del día el dto es como un nexo de comunicacion.**

**Este dto va a contestar a alguien?** Si → tiene que darle lo que necesita. Si va estar formado por uno o varias, sera problema de el.

**Que es REPOSITORIO?** Es la clase que se va a comunicar con la base de datos para realizar sentencias.

**ENTIDAD** es la unidad de la tabla, voy hacer una accion contra esta tabla.

**Siempre tener dto desde el lado del controlador, al menos para comunicarse con el html.**  
**No deberia llegar la entidad de la bbdd al html.**

Un dto puede formarse por la informacion de una entidad o varias. COMO? Apartir del constructor recibiendo como parametro la entidad y luego dentro del constructor hacer getter...

. Y a su vez al revez que el dto pueda formar una entidad. Como? Exponiendo un metodo partiendo de una cartaDto.

```
public Carta obtenerEntidad(){
    Carta cartaEntidad = new Carta();
    cartaEntidad.setId(this.id);
    cartaEntidad.setNombre(this.nombre);
    return cartaEntidad;
}
```

Este tendra que hacer un camino inverso, o sea donde podria recibir los datos para formarse o usar setter para completar la entidad.

El this.id se refiere al objeto dto actual que sabe convertirse en una entidad

Si el DTO venia formado por varias cosas como datos random o aleatorios o no precisos pero puedo formar con esto metodo de obtener entidad le decimos.

Si tenemos un mapeo raro que no coincide por ej al como name y yo tengo nombre, yo le mandare name eso es MAPEAR.

Es traducir lo que la cartaDTO tiene a lo que carta entidad necesita.

Entidad se refiere para que sea para tu bbdd, no?

Exacto esa clase que [tiene @Entity](#) es nuestra clase de dominio es parte de la construccion que va a parar de la bbdd.

Luego tenemos nuestra clase DTO que es una clase de comunicacion con lo de afuera.

La entidad es una clase de uso INTERNO y DTO de uso EXTERNO.

Por ejemplo nosotros nos muestra el contenido y lo usas.

Si consumimos un endpoint vamos a obtener una informacion, yo no se como se construyo todo eso que nos muestra la api pokedex por ejemplo.

Hasta incluso en DTO podemos permitir que otros pueda construir objeto vacios y esta bien pero tambien esta bien no dejar construir objetos vacios o sea hacer `private CartaDto(){}` tambien esta bien, por ej: la clase Math no te deja crear objetos Math

```
public final class Math {
    public static final double E = 2.718281828459045;
    public static final double PI = 3.141592653589793;
    public static final double TAU = 6.283185307179586;
    private static final double DEGREES_TORadians = 0.017453292519943295;
    private static final double RADIANS_TODegrees = 57.29577951308232;
    private static final long negativeZero = -0L;
    private static final long negativeZeroLong = -0L;
    static double twoToTheDoubleScaleUp = 2.0;
    static double twoToTheDoubleScaleDown = 0.5;

    private Math() {
    }

    @IntrinsicCandidate
    public static double sin(double a) {
        return StrictMath.sin(a);
    }
}
```

Las clases estaticas es una clase que solo tiene metodos estaticos por lo tanto te niegan la construccion porque no tiene sentido. Porque solo vamos a consumir metodos.

Uno lo hace propio por lo cual uno como programado obligamos a la gente del como usarlo de esta manera.

Por ejemplo creo una herramienta y esa herramienta le doy a alguien yo voy a querer que lo use de la manera correcta.

Como dexter, si uno deja el boton rojo la gente lo toca. Si lo puede romper, lo va a romper.

**Nuestro trabajo de desarrollo es garantizar que eso no suceda.**

**Y por eso los test son importante.**

**Si tenemos casos de pruebas bien controlados vamos a mitigar un monton de errores, la falla, la posibilidad que venga alguien a tocar el boton y se rompa.**

Tambien podemos hacer un test de existencia. O sea hago un new y me fijo que en el id me venga lo que le pase. Respecto a los test de getter y setter no se suele hacer mucho.

```
public Carta obtenerEntidad(){
    Carta carta = new Carta();
    return this.obtenerEntidad(carta);
}

public Carta obtenerEntidad(Carta cartaEntidad){
    cartaEntidad.setId(this.id);
    cartaEntidad.setNombre(this.nombre);
    return cartaEntidad;
}
```

Tenemos la posibilidad de sobrecargas de metodos en mi dto, usar un metodo vacio o decirle dame basado en este objeto cartaEntidad.

Cada vez que hacemos uso del new se crea una instancia y que cualquier cosa que le asigne. Por ej:

```
Carta carta = new Carta();
Carta c = carta;
```

En realidad el obj c No crea un nuevo objeto. Si no le dice que esta variable c va a apuntar donde esta la direccion de objeto carta.

Y de los 2 metodos el mas usado es el primero.

La Carta puede tener una sobrecarga de metodos, pero se requiere para Hibernate mapee un constructor vacio si o si

```

@Entity
public class Carta {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = 100, nullable = false)
    private String nombre;

    public Carta(){}

    public Carta(Long id, String nombre){
        this.id = id;
        this.nombre = nombre;
    }
}

```

al atributo nombre podemos marcarlo como un campo a configurar con @Column y que le vamos a configurar? Le diremos cuanto tiene que ser el largo por ejemplo 100 y de ese modo hacer que cuando se cree la tabla de bbdd no tenga el valor maximo de largo (255), Tambien podemos decirle que ese campo es requerido o sea NOT NULL usando nullable=false.

Puede haber casos que estamos trabajando con una tabla ya existente y suponete que en esa tabla tiene campo name y nombre, Entonces podemos darle al atributo nombre

**@Column(name="name")**

Tambien podemos decirle que ademas de ser requerido que no puede haber nombres repetidos con **@Column(unique="true")**

Esto del unique esta muy presente en el registro de usuario cuando dice que no debe haber mail repetido.

La entidad va tomar todos los atributos que tenga y lo va a mapear en la tabla salvo que le diga **@Transient** es para decir que eso no quiero que se guarde en la db.

**Tambien tenemos @Table(name= "card")**

Que me permite vincular la entidad con una tabla ya existente que suponete se llame en ingles card o cambiar el nombre de la tabla en la db porque el admin de db me lo solicito.

LA TABLA SE GENERA AUTOMATICAMENTE CON @ENTITY?

Si, Spring junto con hibernate hacer eso, cuando lo levanta todo el contexto de la sesion.

```

@Bean
public LocalSessionFactoryBean sessionFactory(dataSource) {
    LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
    sessionFactory.setDataSource(dataSource);
    sessionFactory.setPackagesToScan(...packagesToScan: "com.tallerwebi.dominio");
    sessionFactory.setHibernateProperties(hibernateProperties());
    return sessionFactory;
}

```

El setPackageToScan() lo que hace es decirle a hibernate en que paquete tiene que buscar aquellas clases q esten marcadas como una entidad y lo hacen de manera automaticamente.

Anotacion de @Service en este momento esa clase se pone a disposicion como servicio posible en inyectar en controladores

**@Transactional**

que hace esto?

Le provee al servicio la posibilidad de manejar transacciones contra la bbdd

Si mi servicio usara dos repositorios, por ejemplo guarda una parte en uno y guarda la otra parte en otro.

Pero si una parte falla no quiero que guarde la otra.

Si dejamos la anotacion @Transactional en la interfaz le estamos diciendo que cualquiera implementacion que implemente esa interfaz va ser transaccional.

Simpre lo que sea Spring @Service, @Repository lo tiene que tener.

@Autowired no es necesario que siempre este presente pero el autowired es la anotacion que nos permite dejar que el objeto, la instancia que se envia como parametro en el constructor pueda ser manejada por spring.

Al menos te pide que tenga almenos una implementacion de la interfaz.

El test del repositorio tiene una particularidad para construirse, si los otros recibian la clase la cual debian llamar.

**Que tendria que recibir este Repotest?**

Entidades no son, aunque si trabajaran con una o muchas por nombre o filtrado.

El repositorio hace uso del sessionFactory, ese sessionFactory fue configurado con el @Bean en la clase HibernateConfig.

En ese momento podemos decir que nace el contexto.

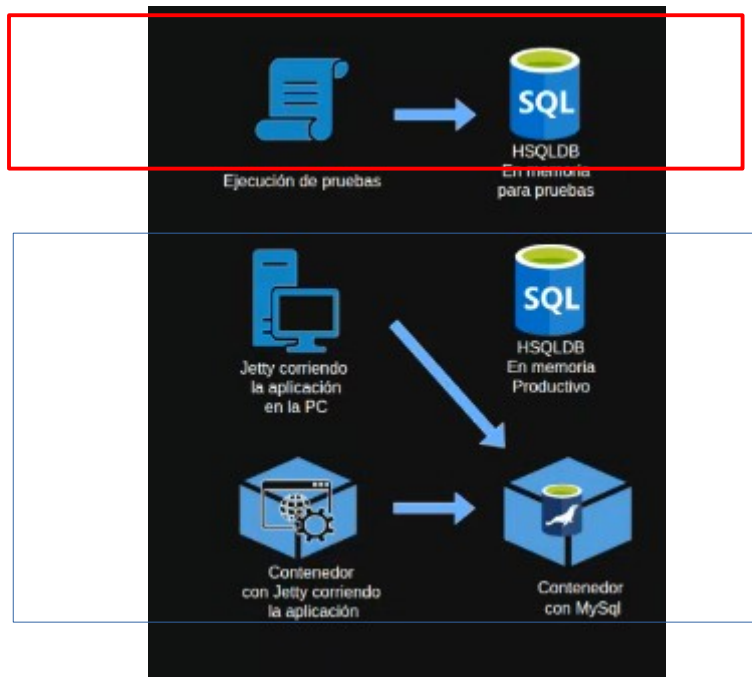
El contexto es levantar todo y lo dejo disponible para que lo puedas usar. Que es todo?

Configuracion de tabla, repositorios.

**Como hacemos que la clase RepositoryTest pueda levantar datos ajena y particular a la que tiene en configuracion productiva?**

Lo hacemos apartir de una clase que ya tenemos hecha en la carpeta Test → integracion → config llamada HibernateTestConfig.java

Y de ese modo usar la base de datos de memoria para los test.



Esa parte roja de configuración se mantiene así.

En el desarrollo podemos usar la base de datos de memoria o el contenedor con MySQL. Si no queremos el contenedor con MySQL podemos correr el servicio MySQL en nuestra PC.

Como le decimos al test que esa configuración remarcada en rojo se mantenga así? Primero hay que darle un archivo de configuración de Hibernate que puede ser el mismo que nos copiamos (está en la carpeta test → integración) o copiarlo en nuestra carpeta de test.

Y para decirle que use ese archivo en particular nuestro test tenemos que tener en cuenta en 2 anotaciones.

```
@ExtendWith(SpringExtension.class)
```

Te da la posibilidad de ir contra una base y un poquito más

```
@ContextConfiguration(classes = HibernateInfraestructuraTestConfig.class)
```

Esta es la anotación importante es la que le dice a mi test que archivo de configuración tiene que usar.

La línea `sessionFactory.getCurrentSession()` será necesitado en todos los métodos del reposicionamiento y no arriba porque sino va estar siempre abierta y eso no queremos.

### Que métodos posibles vamos a tener?

Podemos obtener uno o muchos, hacer un update.

El método `save()` funciona como insert into o update



Y ahora bien como hacemos para agregar restricciones?

Podemos usar criteria o sino hql (Hibernate Query Language)

```
@Override
public Usuario buscarUsuario(String email, String password) {

    final Session session = sessionFactory.getCurrentSession();
    return (Usuario) session.createCriteria(persistentClass:Usuario.class)
        .add(Restrictions.eq(propertyName:"email", email))
        .add(Restrictions.eq(propertyName:"password", password))
        .uniqueResult();
}
```

En este metodo le decimos que crea un criterio de busqueda basado en la clase Usuario y (.add(Restriction.eq)) se refiere que agregue ciertas restricciones como que el mail sea igual al parametro q le estoy enviado a comparar y que sea un unico resultado.

Cuando tengamos que comparar dos objetos en test debemos hacer override en la clase Carta el hashCode y equal porque instancia distinta no son la misma.

En conclusion:

Si yo estoy probando traer informacion primero la tengo que guardar y luego buscarla

Si yo estoy probando guardar tambien puedo buscar

```
@Test
@Transactional //metelo en una transaccional
@Rollback//Deshacelo para luego no tocar otros test
public void dadoQueExisteUnaCartaEnLaBDCuandoLaObtengoPorIdMeDevuelveLaCartaCorrespondiente(){
    //Preparacion
    Carta carta = new Carta();
    carta.setNombre(nombre:"carta");
    this.sessionFactory.getCurrentSession().save(carta);
    //ejecucion
    Carta obtenida = this.repositorioCarta.obtenerPorId(id:1L);

    //validacion
    assertEquals(obtenida, carta); //para que funcione el equal en tdd tenemos q hacer un ov
}
```

En este test estamos dependiendo de dos metodos save(), obtenerPorId() algunos lo ven que es valido testear pero va haber momentos que no y querra que uses otro modo en este caso usar HQL

```
// @Table(name = "carta") // vinculo una tabla existente o nueva
public class Carta {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = 100, nullable = false)
    private String nombre;

    @OneToOne
    private Tipo tipo;

    @Transient // Este atributo no se persistira en la tabla
}
```

De que lado puedo poner el onetoone?

Si lo necesita de los dos lados seran de los dos.

Si tiene que traer el tipo de pokemon con el pokemon, aunque seria medio raro, entonces tambien se lo ponen en el tipo esa anotacion

y si agregamos la anotacion @Column(nullable = false) lo hace not null el tipo o sea requerido  
Pero tendre que ajustar en mi test que ademas exista la carta tambien exista el tipo

MI EXTRA SI QUIERO TERMINAR CON SERVICIO MYSQL PUEDO USAR LO SIGUIENTE  
COMANDO

```
C:\Windows\system32>sc query state=all | findstr /I "mysql"
NOMBRE_SERVICIO: MySQL80
NOMBRE_MOSTRAR : MySQL80

C:\Windows\system32>sc query mysql
[SC] EnumQueryServicesStatus:OpenService ERROR 1060:
El servicio especificado no existe como servicio instalado.

C:\Windows\system32>sc query MySQL80
NOMBRE_SERVICIO: MySQL80
        TIPO                : 10  WIN32_OWN_PROCESS
        ESTADO                : 4   RUNNING
                                (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        C  D_SALIDA_WIN32     : 0   (0x0)
        C  D_SALIDA_SERVICIO : 0   (0x0)
        PUNTO_COMPROB.       : 0x0
        INDICACI  N_INICIO   : 0x0
```

```
C:\Windows\system32>net stop MySQL80
El servicio de MySQL80 est   deteni  ndose....
El servicio de MySQL80 se detuvo correctamente.
```

CLASE 19-06-2025

## Como conectar a ambos contenedores?

Opcion 1: averiguar como hacer una red interna entre contenedores en chatgpt

Opcion 2:

Comandos nuevos

**docker-compose up:** levanta los contenedores hacen la conexion.

```
docker-compose.yml x
docker-compose.yml
1  version: '3.8'
2
3  services:
4    mysql:
5      build:
6        context: .
7        dockerfile: DockerfileMySQL
8      args:
9        MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
10       MYSQL_DATABASE: ${MYSQL_DATABASE}
11       MYSQL_USER: ${MYSQL_USER}
12       MYSQL_PASSWORD: ${MYSQL_PASSWORD}
13      container_name: tallerwebi-mysql
14      environment:
15        MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
16        MYSQL_DATABASE: ${MYSQL_DATABASE}
17        MYSQL_USER: ${MYSQL_USER}
18        MYSQL_PASSWORD: ${MYSQL_PASSWORD}
19      ports:
20        - "3306:3306"
21      volumes:
22        - mysql_data:/var/lib/mysql
23      networks:
24        - tallerwebi-network
25      restart: unless-stopped
26
27    jetty-app:
28      build:
29        context: .
30        dockerfile: DockerfileJetty
31      container_name: tallerwebi-jetty
32      ports:
```

El docker compose.yml es una herramienta donde me lee los archivos dockerfile y lo que hace es levantar los 2 contenedores a la vez y ya lo levanta en un contexto en el que ambos se puedan comunicar en donde conectan los contenedores de jetty y mysql. Y lo hace con "network"

```
49
50 networks:
51   tallerwebi-network:
52     driver: bridge
```

Estos son los pasos para crear el package y ejecutar los contenedores de nuestra aplicacion

```
''' shell
mvn clean package
# Invoco a docker-compose para que me genere contenedores de todos los servicios especificadas
docker-compose up --build

# Invoco a docker para que elimine los contenedores creados
docker-compose down
'''
```

En unas lineas de codigo del docker-compose.yml esta las configuracions del puerto, usuario, contrasenas de la bbdd las cuales estaran en las variables de entorno

```
restart: unless-stopped
environment:
  DB_HOST: ${DB_HOST}
  DB_PORT: ${DB_PORT}
  DB_NAME: ${DB_NAME}
  DB_USER: ${DB_USER}
  DB_PASSWORD: ${DB_PASSWORD}
volumes:
```

## QUE SON LAS VARIABLES DE ENTORNOS?

Las variables de entorno de un sistema son esas variables que se lo puede usar en cualquier parte del SO.

**docker-compose up:** levanta los contenedores

## AHORA COMO COMPILAMOS Y EJECUTAMOS NUESTRO PROYECTO?

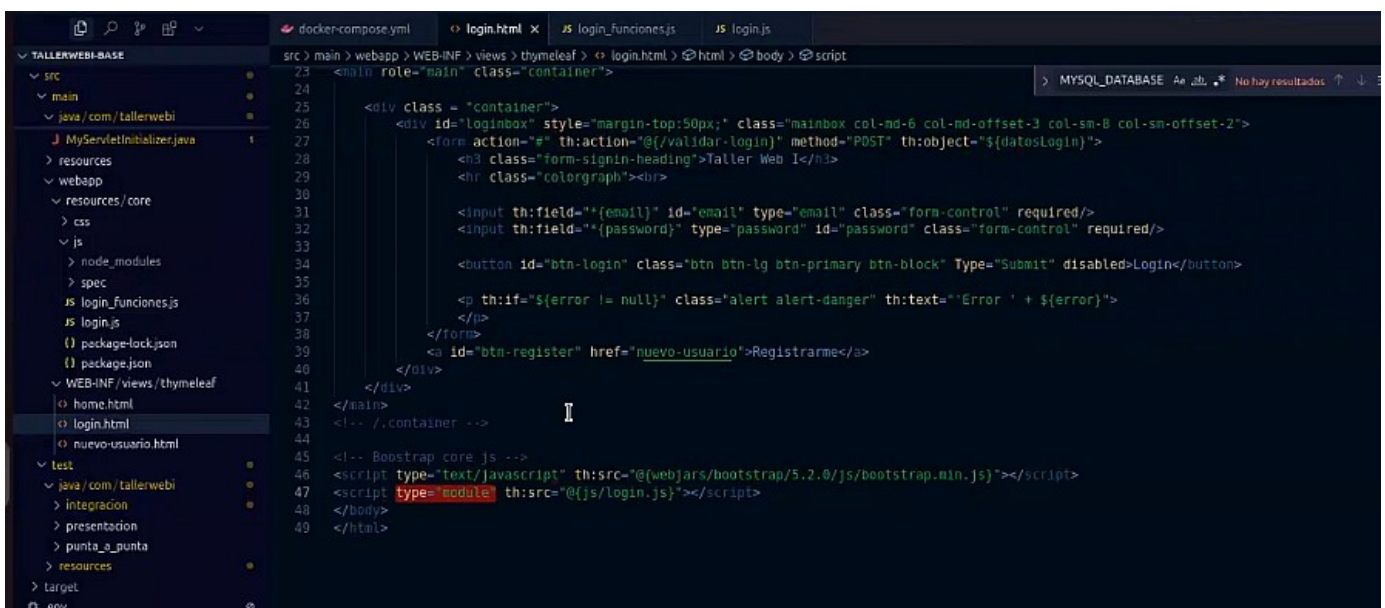
**mvn clean package**

**docker-compose down**

**docker-compose up --build**

**Nota:** cuando borro los contenedores se deja de ver los logs

## LOGIN ACTUALIZADO DEL PROFE



```
src > main > webapp > WEB-INF > views > thymeleaf > login.html > html > body > script
23 <main role="main" class="container">
24
25 <div class="container">
26 <div id="loginbox" style="margin-top:50px;" class="mainbox col-md-6 col-md-offset-3 col-sm-8 col-sm-offset-2">
27 <form action="#" th:action="@{/validar-login}" method="POST" th:object="${datosLogin}">
28 <h3 class="form-signin-heading">Taller Web I</h3>
29 <hr class="colorgraph"><br>
30
31 <input th:field="${email}" id="email" type="email" class="form-control" required/>
32 <input th:field="${password}" type="password" id="password" class="form-control" required/>
33
34 <button id="btn-login" class="btn btn-lg btn-primary btn-block" type="submit">Login</button>
35
36 <p th:if="${error != null}" class="alert alert-danger" th:text="Error ' + ${error}">
37 </p>
38 </form>
39 <a id="btn-register" href="#nuevo-usuario">Registrar</a>
40 </div>
41 </div>
42 </main>
43 <!-- /.container -->
44
45 <!-- Bootstrap core js -->
46 <script type="text/javascript" th:src="@{webjars/bootstrap/5.2.0/js/bootstrap.min.js}"></script>
47 <script type="module" th:src="@{js/login.js}"></script>
48 </body>
49 </html>
```

..

..

```
docker-compose.yml login.html JS login_funciones.js JS login.js x
src > main > webapp > resources > core > js > JS login.js > inputEmailNode.addEventListener("keyup") callback
1 import { validarCamposDeLogin } from "../login_funciones.js";
2
3 const btnLoginNode = document.getElementById("btn-login");
4 const inputEmailNode = document.getElementById("email");
5 const inputPasswordNode = document.getElementById("password");
6
7 inputEmailNode.addEventListener("keyup", (event) => {
8     const inputEmailValue = event.target.value;
9     const inputPasswordValue = inputPasswordNode.value;
10    btnLoginNode.disabled = validarCamposDeLogin(inputEmailValue, inputPasswordValue);
11 });
12
13 inputPasswordNode.addEventListener("keyup", (event) => {
14     const inputEmailValue = inputEmailNode.value;
15     const inputPasswordValue = event.target.value;
16     btnLoginNode.disabled = validarCamposDeLogin(inputEmailValue, inputPasswordValue);
17 });
```

KK

```
docker-compose.yml login.html JS login_funciones.js JS login.js
src > main > webapp > resources > core > js > JS login_funciones.js > validarCamposDeLogin
1 // function validarCamposDeLogin(inputEmailValue, inputPasswordValue) {
2     return inputEmailValue?.length > 0 && inputPasswordValue?.length > 0;
3 }
```

## TESTING DEL LADO JS con JASMINE

```
docker-compose.yml login.html JS login_funciones.js JS login_funciones.test.js M x J ControladorLoginTest.java 2 JS login.js
src > main > webapp > resources > core > js > spec > JS login_funciones.test.js > ...
1 /**
2  * DOCU DE JASMINE, https://jasmine.github.io/api/5.0/global
3  */
4 import { validarCamposDeLogin } from "../login_funciones.js";
5
6 describe("Login Funciones", function() {
7     it("debe devolver true cuando el email y la contraseña son validos", function() {
8         expect(validarCamposDeLogin("test@unlam.edu.ar", "test")).toBe(true);
9     });
10
11     it("debe devolver false cuando el email es 'test@unlam.edu.ar' y la contraseña es '', function() {
12         expect(validarCamposDeLogin("test@unlam.edu.ar", "")).toBe(false);
13     });
14
15     it("debe devolver false cuando el email es '' y la contraseña es 'test', function() {
16         expect(validarCamposDeLogin("", "test")).toBe(false);
17     });
18
19     it("debe devolver false cuando el email es '' y la contraseña es '', function() {
20         expect(validarCamposDeLogin("", "")).toBe(false);
21     });
22
23     it("debe devolver false cuando el email es null y la contraseña es null", function() {
24         expect(validarCamposDeLogin(null, null)).toBe(false);
25     });
26
27     it("debe devolver false cuando el email es null y la contraseña es 'test', function() {
28         expect(validarCamposDeLogin(null, "test")).toBe(false);
29     });
30
31     it("debe devolver false cuando el email es 'test@unlam.edu.ar' y la contraseña es null", function() {
32         expect(validarCamposDeLogin("test@unlam.edu.ar", null)).toBe(false);
33     });
34 }
```



## COMO HAGO RUN LOS TESTS UNITARIOS DE JASMINE??

```
damian@damian-Latitude-3400: ~/.../js
damian@damian-Latitude-3400: ~/.../tallerwebi-base x damian@damian-Latitude-3400
login_funciones.js login.js node_modules package.json package-lock.json spec
damian@damian-Latitude-3400:~/.../js$ npm install
npm up to date, audited 84 packages in 2s
20 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
damian@damian-Latitude-3400:~/.../js$ npm run test
tallerwebi@1.0.0 test
  jasmine
jasmine started

Login Funciones
✓ debe devolver false cuando el email es null y la contraseña es null (0.01 sec)
✓ debe devolver false cuando el email es '' y la contraseña es '' (0.002 sec)
✓ debe devolver false cuando el email es '' y la contraseña es 'test' (0.003 sec)
✓ debe devolver false cuando el email es '' y la contraseña es null (0.001 sec)
✓ debe devolver false cuando el email es 'test@unlam.edu.ar' y la contraseña es '' (0.001 sec)
✓ debe devolver false cuando el email es 'test@unlam.edu.ar' y la contraseña es null (0 sec)
✓ debe devolver false cuando el email es null y la contraseña es 'test' (0 sec)
✓ debe devolver true cuando el email y la contraseña son validos (0.001 sec)

Executed 8 of 8 specs SUCCESS in 0.049 sec.
Randomized with seed 83025.
damian@damian-Latitude-3400:~/.../js$
```

NOTA: instalar node.js y npm previamente de ejecutar el comando npm install

el npm install lo que hace es instalar todas las dependencias que se requiere para hacer test unitarios en js las cuales se indico en el package.json.

En la carpeta js hay un mini proyecto en donde tenemos el archivo package.json el cual funciona muy parecido al pom.xml

En el package.json tenemos las dependencias que me permite instalar las librerias de jasmine para los test.

En esta parte de js debemos saber como separar las capas de responsabilidades el login.js se encarga cosas del DOM y login\_funciones.js se encarga en cosas de validaciones

El test de integracion continua se basa a lo de github action se lo agrego tambien



```
github > workflows > ! ci.yml
8 jobs:
9   Explore-GitHub-Actions:
10    runs-on: ubuntu-latest
11    steps:
12      - run: echo "🚀 Esta tarea (job) fue lanzado automaticamente por el evento ${github.event_name}"
13      - run: echo "🌐 Esta tarea esta corriendo en un servidor ${runner.os} por GitHub!"
14      - run: echo "📁 El nombre de la rama es ${github.ref} y tu repositorio es ${github.repository}."
15      - name: Descargando el codigo
16        uses: actions/checkout@v3
17      - run: echo "💡 El repositorio ${github.repository} ha sido clonado."
18      - run: echo "🏠 El workflow esta listo para testear el codigo"
19      - name: Corriendo las pruebas
20        run: |
21          mvn clean test
22      - run: echo "🍏 El estado de la tarea es ${job.status}."
23      - name: Instalar paquetes de node
24        run: |
25          cd src/main/webapp/resources/core/js
26          npm install
27      - name: Ejecutar pruebas de javascript
28        run: |
29          cd src/main/webapp/resources/core/js
30          npm run test
```

## QUE ES TEST DE PUNTA A PUNTA?

Es decirle al codigo al test que actue como si fuese un humano y que deje de darle importancia desde el lado de javascript o java por ej: si hubo un nullpointer o se desconecto la bbdd u otros errores tecnicos.

Pero esas pruebas es desde punta de vista del usuario. Que es lo que ve el usuario en la pagina? Al usuario no le importa que esta pasando con la bbdd, el usuario quiere saber desde el cartel si se inserto o no se inserto de eso confia.

Estos se le agrega al final de su nombre E2E (end to end) para que el pipeline de github action no lo tome en cuenta para probar.

Se utiliza Playwright q es de microsoft. Tambien esta cypress. El test de punta a punta es tambien conocido como test automatizado o prueba de caja negra ya que es un test q no es tecnico y no usamos @Rollback por lo tanto si registro un usuario por medio de este test queda ahi en la bbdd ya que prueba desde la punta front hasta el back.

```
> java > com > lauterweb > punta_a_punta > VistaLogineZE.java > VistaLogineZE > @CrearContextoYPagina()
@BeforeAll
static void abrirNavegador() {
    playwright = Playwright.create();
    browser = playwright.chromium().launch();
    //browser = playwright.chromium().launch(new BrowserType.LaunchOptions().setHeadless(false).setSlowMo(500));
}

@AfterAll
static void cerrarNavegador() {
    playwright.close();
}

@BeforeEach
void crearContextoYPagina() {
    ReiniciarDB, limpiarBaseDeDatos();
```

Comentamos browser = playwright.chromium() para que no se ejecute automaticamente la ventana del buscador de google y asi poder probar los test unitarios sin que se abra cada rato el navegador ya que si hacemos varios test e2e puede demorar en ejecutarse.

Por eso el profe creo una clase auxiliar para borrar las tablas de la bd.

```
4
5 class ReiniciarDB {
6     public static void limpiarBaseDeDatos() {
7         try {
8             String dbHost = System.getenv("DB_HOST") != null ? System.getenv("DB_HOST") : "localhost";
9             String dbPort = System.getenv("DB_PORT") != null ? System.getenv("DB_PORT") : "3306";
10            String dbName = System.getenv("DB_NAME") != null ? System.getenv("DB_NAME") : "tallerweb1";
11            String dbUser = System.getenv("DB_USER") != null ? System.getenv("DB_USER") : "user";
12            String dbPassword = System.getenv("DB_PASSWORD") != null ? System.getenv("DB_PASSWORD") : "user";
13
14            String sqlCommands = "DELETE FROM Usuario;\n" +
15                                "ALTER TABLE Usuario AUTO_INCREMENT = 1;\n" +
16                                "INSERT INTO Usuario(id, email, password, rol, activo) VALUES(null, 'test@unlam.edu";
17
18            String comando = String.format(
19                "docker exec tallerweb1-mysql mysql -h %s -P %s -u %s -p%s %s -e \"%s\"",
20                dbHost, dbPort, dbUser, dbPassword, dbName, sqlCommands
21            );
22        }
23    }
24 }
```

le decimos a java viste el comando que te pase ejecutalo en una terminal de bash y por que bash y no en cmd? Porque docker tiene una version de linux ligera corriendo.

```
        dbHost, dbPort, dbUser, dbPassword, dbName, sqlCommands
    );
}

Process process = Runtime.getRuntime().exec(new String[]{"/bin/bash", "-c", comando});
int exitCode = process.waitFor();

if (exitCode == 0) {
    System.out.println("Base de datos limpiada exitosamente");
} else {
    System.err.println("Error al limpiar la base de datos. Exit code: " + exitCode);
}

} catch (IOException | InterruptedException e) {
    System.err.println("Error ejecutando script de limpieza: " + e.getMessage());
    e.printStackTrace();
}
```

NOTA: Si quiero reiniciar la bbdd tendre que borrar todas las tablas de bbdd o bien tener otro docker de prueba con mysql no productivo y truncar todo, desactivar la fk, y empezar todo