Configurando un Servidor WildFly con PostgreSQL

Instalación Docker

- Windows:
  - https://docs.docker.com/desktop/install/windows-install/

- Windows: Dependerá de la distribución.

Comandos Docker (Nueva Red)

- **Crear nueva red**

docker network create mired

- **Inspeccionar red**

docker network inspect mired

Comandos Docker (PostgreSql)

- Iniciar PostgreSql (https://hub.docker.com/_/postgres)

```
docker run --network mired -it --rm -p 5432:5432 -e POSTGRES_PASSWORD=123 postgres
```

```
psql -h localhost -p 5432 -U postgres -d postgres
```

- Acceder al docker

```
docker exec -it ID_CONTENEDOR /bin/bash
```

Comandos Docker (Wildfly)

- Modo Standalone con consola

docker run -p 8080:8080 -p 9990:9990 -it quay.io/wildfly/wildfly /opt/jboss/wildfly/bin/standalone.sh -b 0.0.0.0 -bmanagement 0.0.0.0

- Dockerfile (extender la imagen)

```
FROM quay.io/wildfly/wildfly
RUN /opt/jboss/wildfly/bin/add-user.sh admin Admin#70365 –silent
CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0", "-bmanagement", "0.0.0.0"]
```

- Construir Imagen

docker build --tag=jboss/wildfly-admin .

- Correr Imagen

```
docker run  --network mired -p 8080:8080 -p 9990:9990 -it jboss/wildfly-admin
```

Comandos Base de Datos

- Crear un nuevo usuario y Base de datos
  - `CREATE USER nuevo_usuario WITH PASSWORD 'tu_contraseña';`
  - `CREATE DATABASE nueva_base_de_datos OWNER nuevo_usuario;`

- Cambiar el dueño de la base de datos:
  - `ALTER DATABASE nombre_base_de_datos OWNER TO nuevo_usuario;`

- Otros comandos:
  - https://gist.github.com/Kartones/dd3ff5ec5ea238d4c546

Mappings Hibernate

- @ManyToOne
- @OneToMany
- @ManyToMany
- @OneToOne

Mapping @ManyToOne

```
CREATE TABLE "usuarios" (
    permiso BOOLEAN,
    recursos_multimedia_id INTEGER,
    rut INTEGER PRIMARY KEY,
    nombre TEXT,
    FOREIGN KEY (recursos_multimedia_id)
REFERENCES recursos(id) )
```

```
CREATE TABLE "usuarios" (
        "permiso"       boolean,
        "recursos_multimedia_id"  integer,
        "rut"integer NOT NULL,
        "nombre"        varchar(255),
        PRIMARY KEY("rut")
);
```

```java
@Entity
@Table(name="recursos")
public class RecursosMultimedia {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String nombre;
    private String tipo;
    private boolean protegido;
    @OneToMany(mappedBy = "recursosMultimedia", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Usuario> usuarios;
```

```java
@Entity
@Table(name="usuarios")
public class Usuario {

    @Id
    private int rut;
    private String nombre;
    @Column(name="permiso")
    private boolean tienePermiso;
    @ManyToOne()
    @JoinColumn(name = "recursos_multimedia_id")
    private RecursosMultimedia recursosMultimedia;
```

Varios Usuarios pueden estar relacionados con un **único** Archivo Multimedia

Mapping @ManytoMany

CREATE TABLE Asistente (rut bigint not null, email varchar(255), nombre varchar(255), primary key (rut))

CREATE TABLE evento_asistente (id_evento bigint not null, rut bigint not null)

CREATE TABLE Evento (id bigint not null, fecha varchar(255), lugar varchar(255), nombre varchar(255), primary key (id))

```java
@Entity
public class Evento extends Subject {

    @Id
    private Long id;
    private String nombre;
    private String fecha;
    private String lugar;

    @ManyToMany
    @JoinTable(
            name = "evento_asistente",
            joinColumns = @JoinColumn(name = "id_evento"),
            inverseJoinColumns = @JoinColumn(name = "rut"))
    private List<Asistente> asistentes;
```

```java
@Entity
public class Asistente implements Observer {

    @Id
    @Column(name = "rut")
    private Long rut;
    private String nombre;
    private String email;

    @ManyToMany(fetch = FetchType.EAGER, mappedBy = "asistentes",
                cascade = {CascadeType.DETACH,
                           CascadeType.MERGE,
                           CascadeType.PERSIST,
                           CascadeType.REFRESH})
    List<Evento> eventos;
```

Muchos Eventos son asistidos por Asistentes y Muchos Asistentes asisten a Eventos

Maven

- Apache Maven es una herramienta que **estandariza la configuración de un proyecto en todo su ciclo de vida**, como por ejemplo en todas las fases de compilación y empaquetado y la instalación de mecanismos de distribución de librerías, para que puedan ser utilizadas por otros desarrolladores y equipos de desarrollo.
  - Un sistema de gestión dependencias.
  - Un mecanismo distribuido de distribución de librerías. El comportamiento distribuido es siempre desde el repositorio local de Maven hacia los repositorios que están publicados en Internet o en la red corporativa.
  - Mecanismos para ser extensible, por la creación de plugins customizables.
  - Es multi-plataforma, puede funcionar tanto en entornos Linux como Windows al ser una aplicación Java.
  - Es software libre, con lo cual es el código está disponible, se podría modificar y customizar en caso de que fuera necesario.
  - Fomenta la reutilización de código y de librerías. El hecho de que Apache Maven ofrezca repositorios oficiales y públicos de software libre, con librerías desplegadas, que toda la comunidad de desarrolladores de software utiliza, hace que este concepto también pueda trasladarse al mundo empresarial, a través de repositorios remotos corporativos, compartidos por distintos equipos de proyectos o el propio equipo de desarrollo.
  - Es compatible con múltiples IDEs.

Ciclo de Vida

compile: Compila el código fuente
test: Compila y ejecuta las pruebas
package: Empaqueta el código fuente
install: Instala el proyecto en el repositorio local
deploy: Instala el proyecto en el repositorio remoto
clean: Borra la carpeta target

Maven: POM.xml

```
1.  <project>
2.    <modelVersion>4.0.0</modelVersion>
3.    <groupId>br.com.rodrigobranas</groudId>
4.    <artifactId>my-app</artifactId>
5.    <version>1.0</version>
6.    <dependencies>
7.      <dependency>
8.        <groudId>org.seleniumhq.selenium</groupId>
9.        <artifactId>selenium-java</artifactId>
10.       <version>2.24.1</version>
11.     </dependency>
12.   </dependencies>
13. </project>
```

Trabajo de Clase

- https://github.com/IS-LAB-EIC-UCN/banco

  - Adicionar una clase cuenta.
  - Crear la relación un cliente posee muchas cuentas
  - Buscar las cuentas asociadas a un cliente (MVC).