UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITECNICA SUPERIOR DE GANDIA

GRADO EN ING. SIST. DE TELECOM., SONIDO E IMAGEN

"Desarrollo de una aplicación web en java usando "GWT material design", para el control y mantenimiento de plantas solares fotovoltaicas"

TRABAJO FINAL DE GRADO

Autor/a:
Lucía Andreu López

Tutor/a:
José Marín-Roig Ramón
José Daniel Boscá Candel

GANDIA, 2018

AGRADECIMIENTOS

Quiero aprovechar la ocasión para agradecer a las siguientes personas que me han ayudado en el desarrollo del proyecto fin de carrera

A mi cotutor, José Daniel Boscá Candel, por cada uno de los consejos que me ha dado, pues siempre ha estado pendiente del avance del proyecto y ha tenido soluciones para los problemas y contratiempos que hayan podido ocurrir. También me gustaría dar las gracias a el tutor del proyecto José Marín-Roig Ramón, por acceder a dirigirme el proyecto pese a la cantidad de proyectos que está dirigiendo.

Y por último, quiero dedicarle este trabajo a mi familia, por todos los ánimos que me dieron.

RESUMEN

Los últimos avances tecnológicos están permitiendo el diseño de plataformas web más visuales y amigables. Y es por eso que en este TFG, se pretende concentrar los aspectos más importantes del último lanzamiento de Google en diseño tanto en plataformas móviles como web's.

Para ello se ha realizado una aplicación web con la herramienta Google Web Toolkit y la librería GWT-material, que explique la configuración del entorno de desarrollo con Eclipse, y la elaboración de una plataforma con Amcharts, chartist-js y flexbox, para que con la ayuda de un mapamundi muestre unos lugares de interés que estén relacionados con las células fotovoltaicas.

Palabras Clave: Java, Web, Google Web ToolKit, Material Design

ABSTRACT

The latest technological advances are allowing the design of more visual and friendly web platforms. In this TFG is intended to concentrate the most important aspects of the latest launch of Google design in both mobile and web platforms.

To accomplish this, for that purpose we make a web application with the Google Web Toolkit tool and the GWT-material library, which explains the situation of the development environment with Eclipse, and the development of a platform with Amcharts, chartist-js and flexbox, so that with the help of a world map it shows some places of interest that are related to the photovoltaic cells.

Key Words: Java, Web, Google Web ToolKit, Material Design

ÍNDICE

Índice de la Memoria

Índice de figuras

Índice de tablas

1. Introducción

Este proyecto ha sido desarrollado para la empresa Joboscan Projects S.L. Situada en la C/Pintor Goya, 12 en Benicull. Este proyecto surge por la necesidad de modernizar una antigua plataforma de monitorización visualmente con las librerías GWT material design e implementar la localización de instalaciones fotovoltaicas. El proyecto se instalará en la web habilitada por la empresa.[1]

1.1. Metodología

Este proyecto esta basado en una versión anterior de la empresa, de la cual hemos estudiado y partimos de ella para poder mejorarla según las especificaciones que nos ha dado la empresa. La empresa tenía la necesidad de mejorar la visualización de la aplicación, la cual había sido previamente desarrollada, pues el entorno gráfico se había quedado anticuado y no podían competir con la competencia pese a tener más calidad y solidez.

1.2. Objetivo del proyecto

El objetivo principal de este proyecto es el de modernizar una antigua plataforma de monitorización visualmente con las librerías GWT material design ,para implementar la localización de instalaciones fotovoltaicas, implementando una aplicación amigable y responsive a los usuarios desde cualquier navegador Web. Como objetivos secundarios nos enfocaremos a elaborar un turorial de los pasos a seguir para poder emplear '*GWT material design'.* De esta manera se logrará evitar que futuros proyectos relacionados con *GWT material design',* no se completen debido a la falta de guías o tutoriales.

Hemos elegido este tipo de tecnología, pues es agradable visualmente y posee un entorno bastante amigable hacia el usuario. También *'GWT material design',* nos ofrece todas las características necesarias para crear este tipo de proyecto.

Actualmente hay sistemas implementados parecidos al desarrollado en este proyecto, pero más antiguos y con otro tipo de tecnología menos amigable y visual.

1.3. Estructura del documento

En este documento se detallan todas las fases que se han ido desarrollado para la realización del proyecto llevado a cabo, para la implementación de una plataforma

web para el control y mantenimiento de plantas solares fotovoltaicas. La estructura del proyecto se divide en un capítulo específico por cada etapa que se realiza, más un anexo con información adicional para afianzar los conocimientos teóricos respecto a los contenidos y recursos empleados en el proyecto.

En primer lugar se describe la configuración de requisitos software que deben ser instalados para poder modificar e implementar la aplicación. A continuación se documentan las etapas referentes al análisis y diseño de la aplicación. En estas etapas se han añadido diagramas de estado para la especificación del código de la aplicación. Por último se mostrará como implementar la aplicación y su resultado final.

1.4. Estudio del estado del arte

Dentro del ámbito de este Proyecto Fin de Carrera se pueden encontrar diversas soluciones comerciales como pueden ser las siguientes:

Papendorf: Esta solución viene de la mano de la empresa Papendorf Software Engineering. Esta solución monitorizan, miden y gestionan respecto a la energía producida.

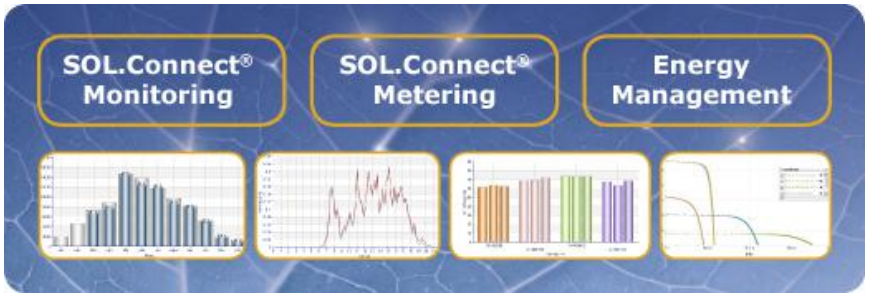

Figura 1: Captura de las funciones de Papendorf

Monsol: Esta solución esta dedicada al diseño, fabricación, suministro e integración de servicios relacionados con el sector fotovoltaico y la monitorización de equipos, señales o datos.

Figura 2: Captura de la demo de Monsol

Meteocontrol: Esta solución se encarga de la monitorización y control remoto perfectos para asegurar el rendimiento de plantas en todos los segmentos de potencia

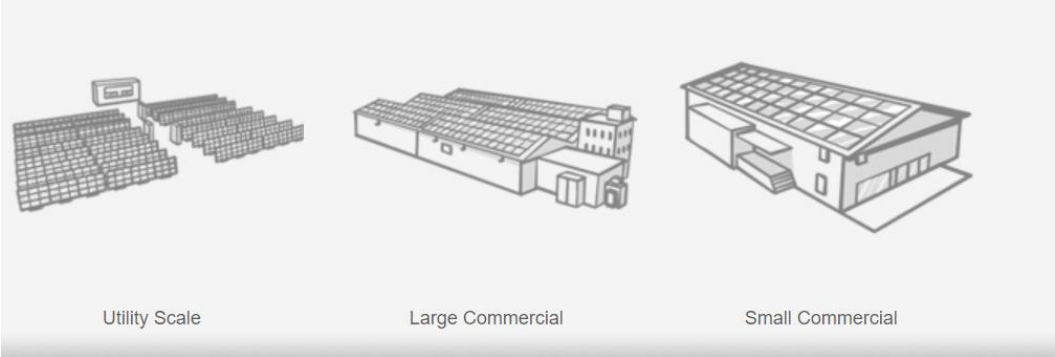

Figura 3: Captura de las soluciones de Meteocontroll

En nuestro caso hemos escogido trabajar con GWT Material Designs por su gran adaptabilidad, es un recurso más nuevo que los que emplean las demás soluciones y es más agradable y cómodo para el usuario.

2. Descripción de las herramientas de trabajo

En este apartado pasaremos a explicar las herramientas software que se van a utilizar en este proyecto.

2.1. Herramientas de compilación y automatización

2.1.1. maven

maven[2] es una herramienta open-source, su lanzamiento fue el 30 de marzo del 2002, con el objetivo de simplificar los procesos de build (compilar y generar

ejecutables a partir del código fuente).

Figura 4: Logo de maven

Antes de que maven proporcionara una interfaz común para hacer builds del software, cada proyecto solía tener a alguna persona dedicada exclusivamente a configurar el proceso de build. Además, los desarrolladores tenían que perder tiempo en aprender las peculiaridades de cada nuevo proyecto en el que participaban. Si se quería compilar y generar ejecutables de un proyecto, se tenía que analizar qué partes de código se debían compilar, qué librerías utilizaba el código, dónde incluirlas y qué dependencias de compilación había en el proyecto.

En el mejor de los casos, se empleaban unos pocos minutos para saber cómo hacer una build del proyecto. En el peor de los casos, el proceso de build era tan complejo que un desarrollador podía tardar horas en saber cómo compilar y generar los ejecutables a partir del código. Ahora, la build de cualquier proyecto maven, independientemente de sus módulos, dependencias y librerías, consiste simplemente en ejecutar el comando "mvn install".

maven es una herramienta capaz de gestionar un proyecto software completo, desde la etapa en la que se comprueba que el código es correcto, hasta que se despliega la aplicación, pasando por la ejecución de pruebas y generación de informes y documentación.

Para ello, en maven se definen tres ciclos de build del software con una serie de etapas diferenciadas. El ciclo por defecto tiene las etapas de:

- ✓ Validación (validate): Validar que el proyecto sea correcto.
- ✓ Compilación (compile).
- ✓ Test (test): Probar el código fuente usando un framework de pruebas unitarias.
- ✓ Empaquetar (package): Empaquetar el código compilado y transformarlo en algún formato tipo .jar o .war.
- ✓ Pruebas de integración (integration-test): Procesar y desplegar el código en algún entorno donde se puedan ejecutar las pruebas de integración.

- ✓ **Verificar** que el código empaquetado es válido y cumple los criterios de calidad **(verify).**

- ✓ **Instalar** el código empaquetado en el repositorio local de maven, para usarlo como dependencia de otros proyectos **(install).**

- ✓ **Desplegar** el código a un entorno **(deploy).**

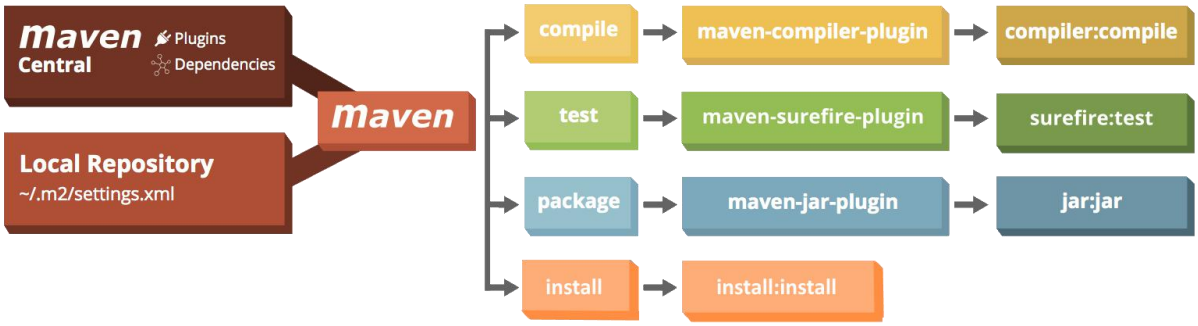

Figura 5: Esquema de las etapas de maven[3 y 4]

Para poder llevar a cabo alguna de estas fases en nuestro código, tan solo tendremos que ejecutar "mvn" y el nombre que corresponda (validate, compile, package, integration-test, verify, install y deploy). Además van en cadena, es decir, si empaquetamos el código (package), maven ejecutará desde la primera etapa de validación (validate) a empaquetación (package).

Para poder ejecutar estos comandos en primer lugar instalaremos maven y seguiremos este tutorial para instalar maven Apache[5]. Y a continuación iremos escribiendo los comandos necesarios en nuestro símbolo del sistema (cmd)[6].

2.1.2. Ant

Apache Ant,[7] fue creada en julio del 2000, es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build). Es, por tanto, un software para procesos de automatización de compilación. Desarrollado en lenguaje Java y requiere la plataforma Java, así que es más apropiado para la construcción de proyectos Java.

Ant utiliza XML para describir el proceso de generación y sus dependencias. Por defecto, el archivo XML se denomina build.xml.

Ant es un proyecto de la Apache Software Foundation. Es software open source, y se lanza bajo la licencia Apache Software. Es la predecesora de maven, actualmente

sigue en funcionamiento con las versiones 1.9.11 y 1.10.3 lanzadas el 27 de Marzo del 2018.[8]

Figura 6: Logo de Apache Ant

2.1.3. gradle

Es un sistema de automatización de construcción de código abierto, lanzado en el 2007. Introduce un lenguaje especifico del dominio (DSL) basado en Groovy para declarar la configuración de proyecto. Gradle utiliza un grafo acíclico dirigido ("DAG") para determinar el orden en el que las tareas pueden ser ejecutadas.

Determina qué partes del árbol de construcción están actualizadas, de modo que cualquier tarea dependiente a aquellas partes no necesitarán ser reejecutada.

Los plugins iniciales están principalmente centrados en el desarrollo y despliegue en Java, Groovy y Scala.[9]

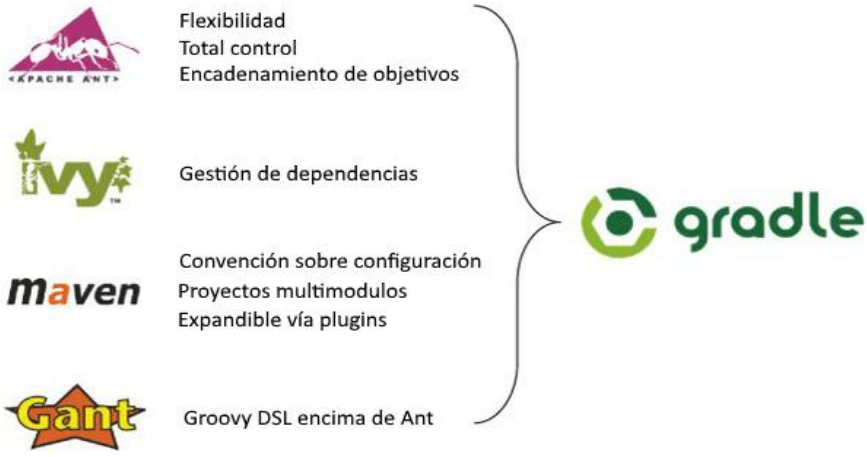

Figura 7: Integración de Gradle[10]

Gradle es el sistema más simple y flexible entre los existentes, pero en nuestro caso hemos considerado que al ser el más reciente podría ser más propenso a tener bugs y a tener menos documentación que nos pudiera resultar útil. El siguiente ejemplo es con gradle en el build.gradle.[11]

```
apply plugin: 'groovy'
```

```
Group= “info.solidsoft.rnd”
Version=”0.0.1-SNAPSHOT”

repositories {
    mavenCentral()
}
 dependencies{
compile 'org.codehaus.groovy:groovy-all:2.4.1'
testCompile 'org.spockframework:spock-core:1.0-groovy-2.4'
}

rootProject.name=’spock-10-groovy-24-gradle-maven’
```

2.1.4. Comparativa entre maven, Ant y gradle

Especificación	Ant	maven	gradle
Formato de script de compilación	XML	XML	Groovy/DSL
Dependencias	Ivy	Incorporadas	Incorporadas
Construcciones de múltiples módulos	Complejo	Simple	Simple
Estructura predefinida	Ausente	Presente	Presente
Estructura personalizable	N/a	Compleja	Simple
Verbosidad	Alta	Media	Baja
Curva de aprendizaje	Llana	Empinada	Media

Tabla 1: Comparativa entre Apache Ant, Apache maven y Gradle[12]

A continuación podemos observar el código en Ant del build:[13]

```
<project name="my-project" default="dist" basedir=".">
   <description>
      simple example build file
   </description>
   <!-- set global properties for this build -->
   <property name="src" location="src/main/java"/>
   <property name="build" location="target/classes"/>
   <property name="dist"  location="target"/>

   <target name="init">
     <!-- Create the time stamp -->
     <tstamp/>
     <!-- Create the build directory structure used by compile -->
     <mkdir dir="${build}"/>
```

```
  </target>
  <target name="compile" depends="init"

     description="compile the source " >
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}"/>
  </target>

  <target name="dist" depends="compile"
     description="generate the distribution" >
    <!-- Create the distribution directory -->
    <mkdir dir="${dist}/lib"/>

    <!-- Put everything in ${build} into the MyProject-${DSTAMP}.jar file-->
    <jar jarfile="${dist}/lib/MyProject-${DSTAMP}.jar" basedir="${build}"/>
  </target>

  <target name="clean"
     description="clean up" >
    <!-- Delete the ${build} and ${dist} directory trees -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```

El siguiente ejemplo es con maven, el archivo del pom.xml que logra el mismo resultado que el anterior código que hemos visto del build del Ant.

```
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>org.sonatype.mavenbook</groupId>
<artifactId>my-project</artifactId>
<version>1.0</version></project>
```

2.1.5. Conclusión

En nuestro caso hemos optado por emplear maven, debido a la gran facilidad que nos proporciona con el control de dependencias y posee mayor simplicidad , posee menor configuración verbosa y build scripts que ant. Además maven y gradle son

muy parecidos, las diferencias que tienen no tienen un gran impacto en el proyecto porque se van muy poco uno de otro. Por lo que se ha considerado que no importaba cual de los dos se usara.

En este caso se ha utilizado eclipse por una preconfiguración de la empresa.

2.2. Herramienta de configuración

2.2.1. Eclipse

Eclipse[14] es una plataforma de software compuesta por un conjunto de herramientas de programación open source multiplataforma, en la que se emplean herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones.

Diseñada para ser extendida de forma indefinida a través de plug-ins, fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. No tiene en mente un lenguaje específico, sino que es un IDE genérico

Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE). Esta compuesto por un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código. Algunos contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse; otros no, tales como SharpDevelop y Lazarus.

2.3. Instalaciones

2.3.1. Instalación de maven

Empezaremos asegurándonos de instalar[15] maven de la página oficial. Una vez descargado, lo siguiente que debemos hacer es descomprimir la carpeta de maven, en este caso usamos la versión apache-maven-3.5.2 por ser la más reciente en el momento en el que se ha creado el proyecto.

2.3.2. Instalación de eclipse

A continuación seguiremos descargando[16] Eclipse OXYGEN, de la página web oficial, descargaremos eclipse-inst-win64.exe. También debemos descargarnos el Java

Platform (JDK) de 64 bits, en este caso descargaremos el 9, por ser el más actual cuando se creó el proyecto.

Una vez descargados ambos ficheros, podemos ejecutar el .exe del eclipse. A continuación aparecerá el '**eclipseinstaller**' el cual nos dará distintas opciones o usos disponibles del Eclipse IDE. En nuestro caso optamos por '*Eclipse IDE for Java Developers*', pues esta opción tiene las herramientas tales como Java IDE, Git, editor de XML y maven entre otras. El último paso antes de que comience la primera instalación es aceptar la licencia de uso.

Tras la instalación te pide que reinicies eclipse, tras volverlo a abrir te pregunta por la versión del producto(32 o 64 bits), la ruta de Java(jre) y la ruta donde quieres que se instale. Tras haber configurado estos tres parámetros y haber aceptado los términos de uso comienza la instalación final.

Por último cuando termina la instalación aparece el botón ' *launch* ', el cual ejecutará eclipse, al ser la primera vez te pedirá que le indiques la ruta del directorio que quieras establecer como lugar de trabajo(workspace), el lugar que se guardarán todos los proyectos.

2.4. Configuraciones

2.4.1. Configuración del proyecto con maven

Una vez tenemos maven descargado y descomprimido, pasaremos a editar la variable de entorno PATH del sistema, para añadir maven y que los ejecutables sean accesibles a la consola y a los entornos de desarrollo instalados en el sistema. Para ello acudimos a Propiedades del sistema y hacemos clic sobre el botón Variables de entorno...

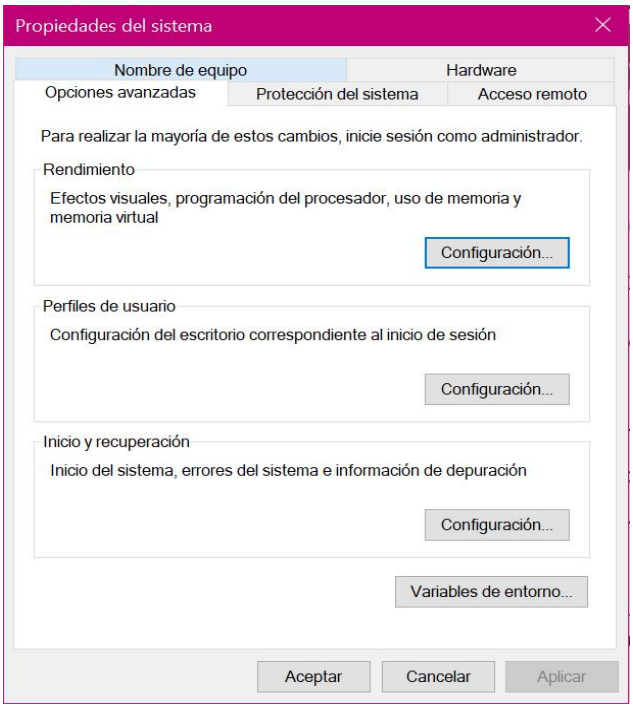

Figura 8: Propiedades del sistema

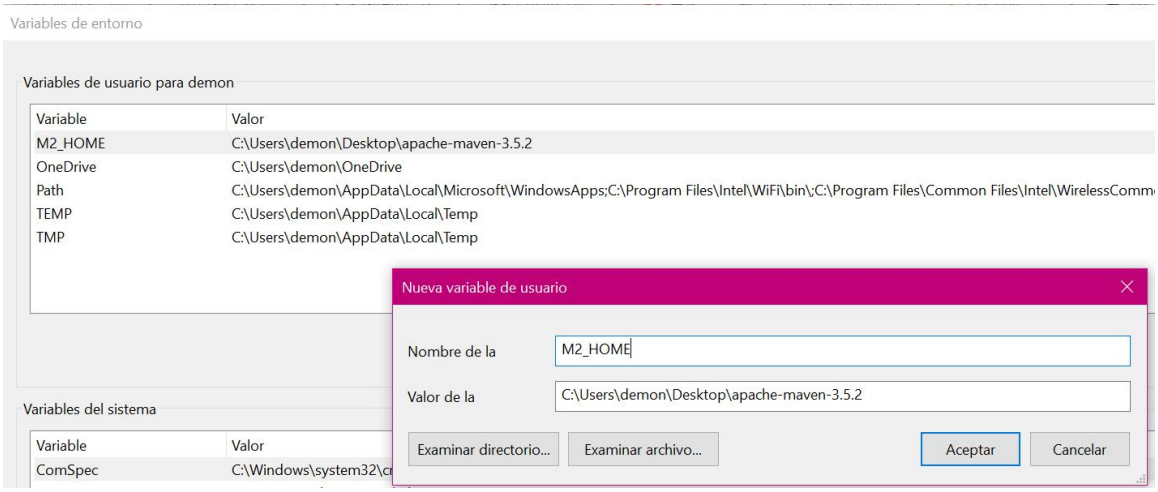

Figura 9: Creación de una nueva variable con el directorio de maven

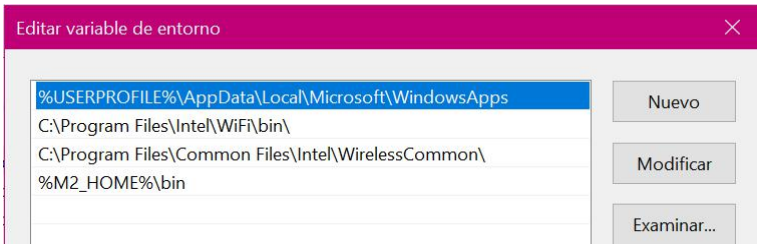

Figura 10: Llamada de la variable creada

Para comprobar la instalación en el símbolo del sistema escribimos el comando mvn -version

Figura 11: Comprobación de maven

2.4.2. Configuración del proyecto con GWT

GWT(Google Web Toolkit), es un framework (módulos de desarrollo de aplicaciones) creado por Google que permite desarrollar fácilmente aplicaciones web. Fue creado en 2006 por Google.[17] Es un kit de herramientas de desarrollo para crear y optimizar aplicaciones complejas basadas en navegador. Su objetivo es permitir el desarrollo de aplicaciones web de alto rendimiento sin que el desarrollador tenga que ser un experto en XMLHttpRequest y JavaScript. GWT es utilizado por muchos productos de Google; AdWords, AdSense, Vuelos, Buscador de hoteles, Ofertas, Monedero y Blogger. Es de código abierto, completamente gratuito.

Dentro de el kit de herramientas se encuentran el SDK y el plugin para Eclipse:

- ✓ El GWT SDK contiene las librerías de la API de Java, el compilador y el servidor de desarrollo. Permite escribir aplicaciones del lado del cliente en Java e implementarlas como JavaScript.

- ✓ El plugin para Eclipse, ofrece soporte IDE para proyectos web de GWT y App Engine.

Seguidamente buscaremos y descargaremos[18] los archivos necesarios para poder usar gwt. Tras la descarga de gwt-2.8.2.zip, se extrae, esta carpeta contiene las librerías principales, el compilador y el servidor de desarrollo que necesita para escribir aplicaciones web.

Se ha comprobado que con la versión empleada aparecía un problema entre Google App Engine SDK y GDT Pulldown, el cual si se instalaba una herramienta la otra desaparecía. Por lo que se ha optado por mantener GDT Pulldown ya que mantiene el GWT y no nos ha dado ningún problema.

2.4.3. Configuración de GWT en Eclipse

A continuación, deberemos ir a Help>Eclipse Marketplace... , y en 'find' buscaremos 'gwt' e iremos instalando todos los plugins relacionados con gwt.

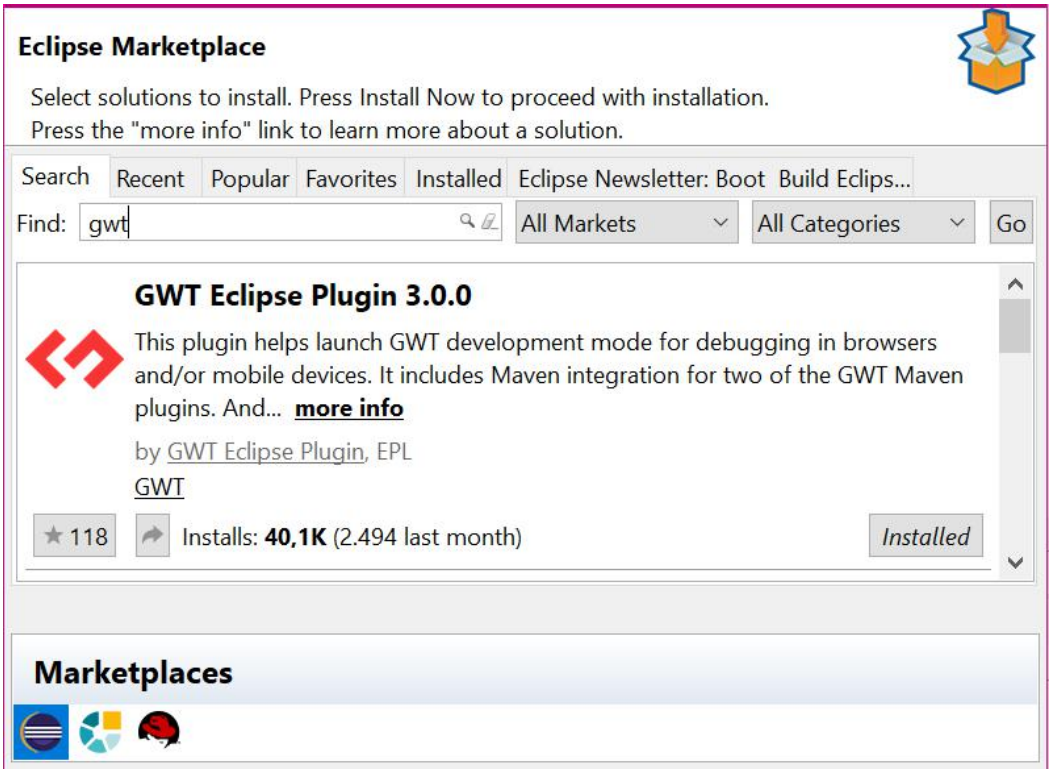

Figura 12: Búsqueda en Eclipse Marketplace de los plugins gwt

3. Guía de desarrollo

En este apartado explicaremos los pasos a seguir para poder adaptarnos y realizar una puesta en marcha en este nuevo entorno, para ello seguiremos los tutoriales oficiales[19] teniendo en cuenta las nuevas versiones y sus características para poder generar nuestro primer proyecto.

3.1. Instalación del entorno de trabajo

A continuación pasamos a generar el proyecto con el siguiente comando:

```
mvn archetype:generate -DarchetypeGroupId=com.github.gwtmaterialdesign -DarchetypeArtifactId=gwt-material-archetype -DarchetypeVersion=2.0[20].
```

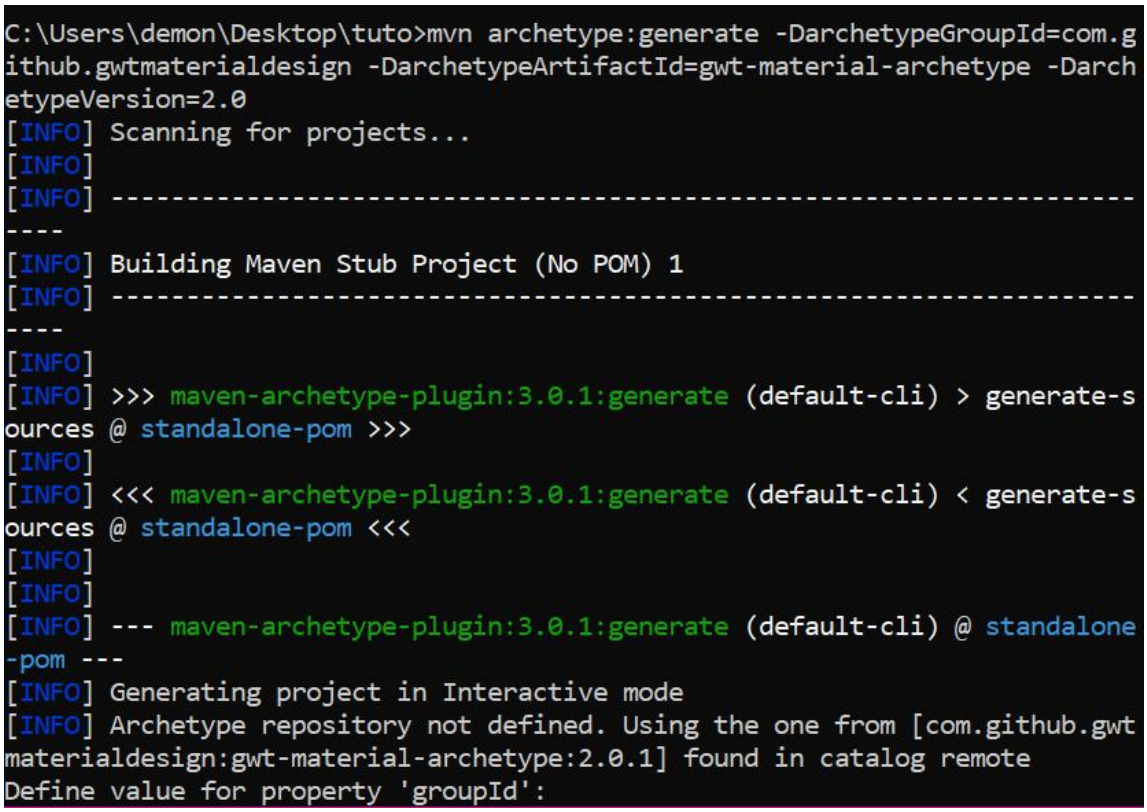

Figura 13: Creación de un proyecto ejemplo con maven

Una vez terminada la descarga nos preguntará sobre el groupId, artifactId, version y package:

- ✓ GroupId: Identificará el proyecto de forma única respecto a todos los proyectos, por lo que debemos aplicar un esquema de nomenclatura. Este esquema tiene que seguir las reglas de denominación del paquete, lo que implica que al menos tiene que ser como un nombre del dominio que el usuario controle, y se puede crear tantos subgrupos como se desee.[21]

- ✓ ArtifactId: Es el nombre del jar sin versión. Si lo has creado, puedes elegir el nombre que desees mientras contenga letras minúsculas y/o no contenga símbolos extraños. Si es un de un jar de terceros, se debe tomar el nombre del jar a medida que se distribuye.7

- ✓ Version: Si se distribuye, se puede elegir cualquier versión típica con números y puntos (1.0, 1.1, 1.0.1, ...). No se debe usar fechas, ya que suelen estar asociadas a compilaciones SNAPSHOT (nocturnas). Si se trata de un artifact de un tercero, se debe usar su número de versión, sea cual sea.

En este caso crearemos el proyecto con estos datos:

Define value for property 'groupId': com.gwtmaterial.tutorial
Define value for property 'artifactId': gwt-material-tutorial

Define value for property 'version' 1.0-SNAPSHOT: :
Define value for property 'package' com.gwtmaterial.tutorial: : com.gwtmaterial.tutorial

Y por último antes de generar el proyecto nos pedirá una confirmación:
Y: : Y

```
[INFO] Parameter: groupId, Value: com.gwtmaterial.tutorial
[INFO] Parameter: artifactId, Value: gwt-material-tutorial
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.gwtmaterial.tutorial
[INFO] Parameter: packageInPathFormat, Value: com/gwtmaterial/tutorial
[INFO] Parameter: package, Value: com.gwtmaterial.tutorial
[INFO] Parameter: gwt-material-version, Value: 2.0
[INFO] Parameter: groupId, Value: com.gwtmaterial.tutorial
[INFO] Parameter: artifactId, Value: gwt-material-tutorial
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: C:\Users\demon\Desktop\tuto\g
wt-material-tutorial
[INFO] ------------------------------------------------------------------------
----
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
----
[INFO] Total time: 02:56 min
[INFO] Finished at: 2018-05-16T13:31:22+02:00
[INFO] Final Memory: 16M/70M
[INFO] ------------------------------------------------------------------------
----

C:\Users\demon\Desktop\tuto>
```

Figura 14: Instalación correcta con maven

Si entramos en la carpeta donde hemos creado el proyecto, debería tener esteo un aspecto similar:

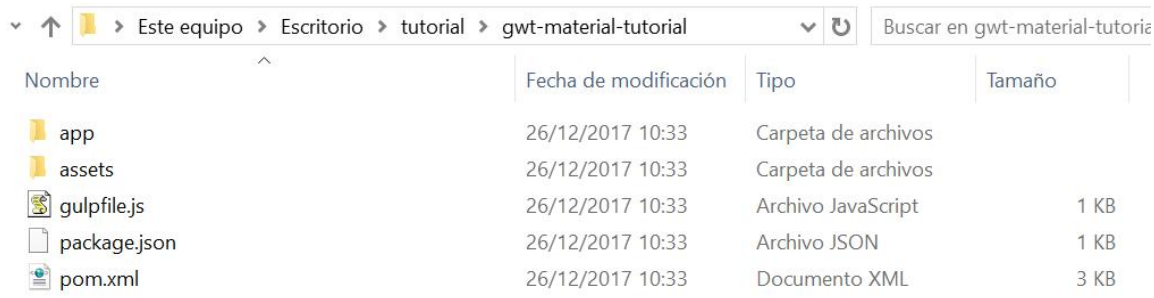

Figura 15: Estructura del proyecto de el tutorial

3.2. Configuración del proyecto con Eclipse

Una vez instalado, importamos el proyecto a Eclipse mediante la opción de : **Import>Maven>Existing Maven Projects.**

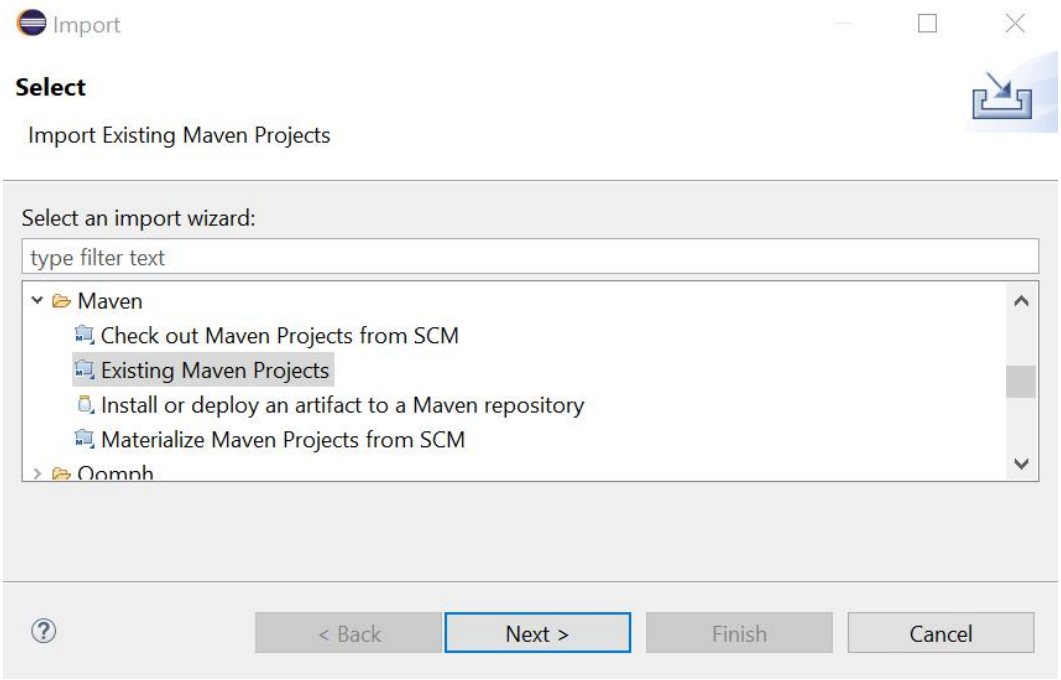

Figura 16: Importación de el proyecto creado previamente

A continuación, tras darle 'next 'le especificamos el directorio en el que se encuentra el proyecto creado, asegurándonos que seleccione el archivo 'pom.xml'.

El **'Project Object Model' o POM,** es la unidad de trabajo fundamental de maven. Es un archivo XML que contiene información sobre el proyecto y los detalles de configuración que usa maven para generar el proyecto. También contiene las metas que pueden ser ejecutadas. Cuando se ejecuta una tarea o meta, maven busca al POM en el directorio actual. Lee el POM, obtiene la información que necesita sobre la configuración, para luego ejecutar la meta.[22]

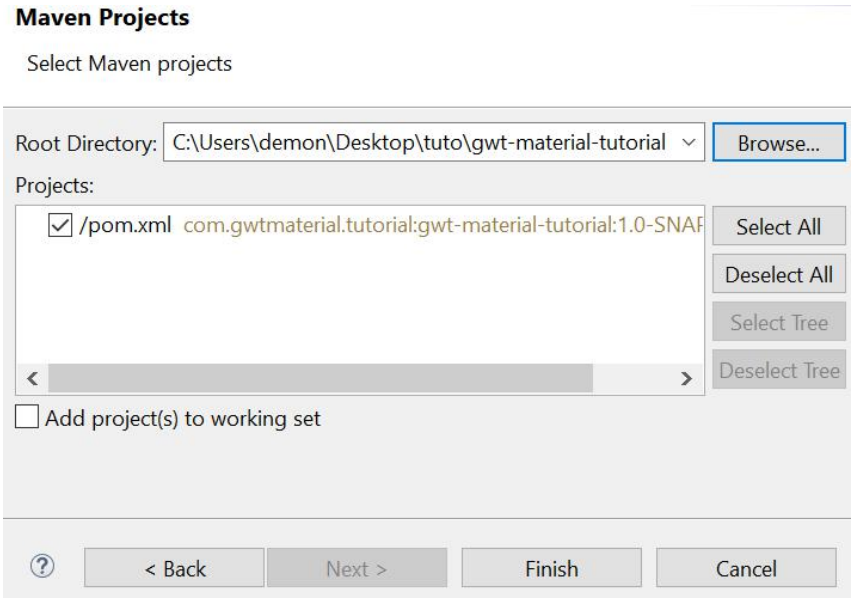

Figura 17: Importación de el proyecto, seleccionado el pom

Cuando el proyecto se ha importado es posible que se observe un error en el pom.xml '*Plugin execution not covered by lifecycle configuration: org.apache.maven.plugins:maven-antrun-plugin:1.8:run (execution: gulp, phase: generate-resources)*' el cual es un problema de M2eclipse y se soluciona poniendo '<pluginManagement>' en el pom.xml, de la siguiente forma:

```
<build>
<pluginManagement>
<plugins>
<plugin> ... </plugin>
<plugin> ... </plugin>
   ....
</plugins>
</pluginManagement>
</build>
```

En este momento la estructura del proyecto deberá tener el aspecto que se aprecia en la siguiente figura:

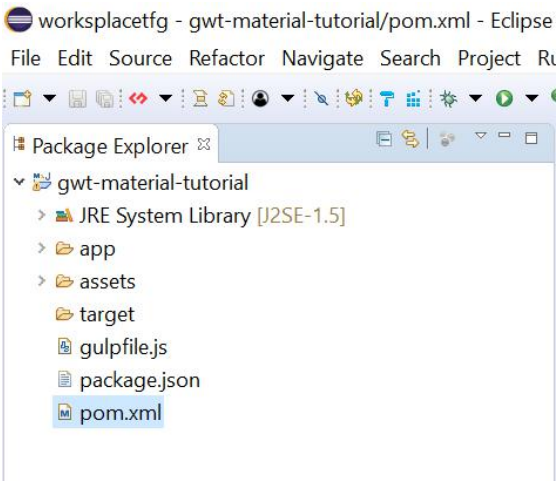

Figura 18: Estructura del proyecto en Eclipse

A continuación, continuaremos configurando mediante **'Run>Run configurations..'**. En el menú de la izquierda buscamos Maven Build, y lo seleccionamos de esta manera crearemos una nueva configuración. En el menú central completaremos con el nombre de la configuración, la dirección del directorio. En el apartado de goals deberemos poner el comando **'gwt:run'** para que el proyecto arranque, en anteriores versiones se empleaba el comando **'run'.** Terminamos con Apply y Close.

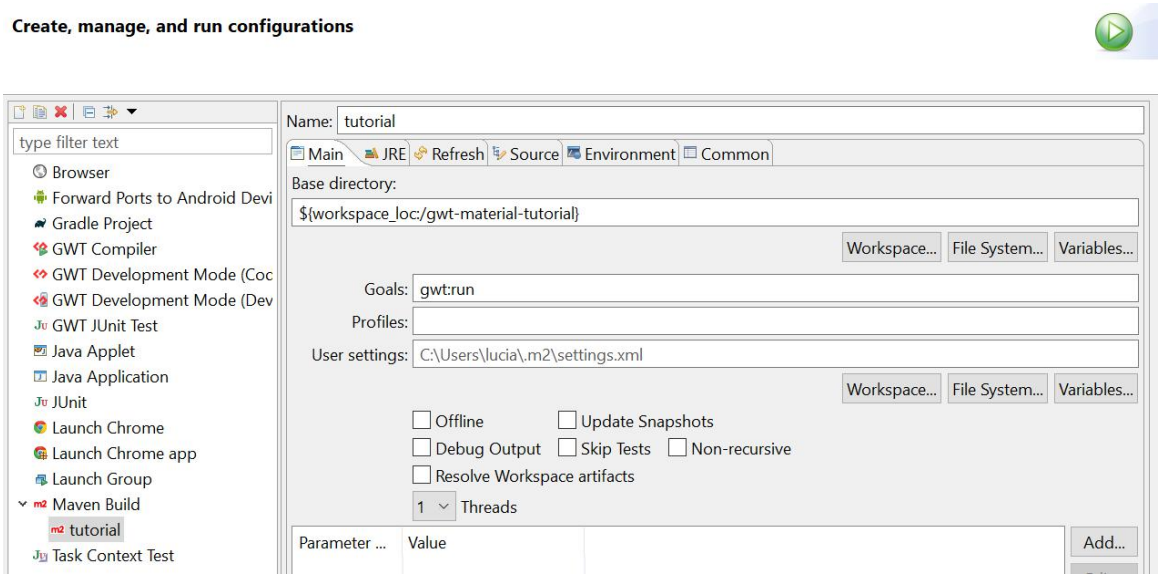

Figura 19: Modificando las configuraciones del compilador

Para evitar cualquier tipo de problema nos aseguraremos de tener instalado una versión reciente de jdk[23], y la seleccionaremos en el proyecto. Se ha comprobado que con el jdk 9 y 8 funciona.

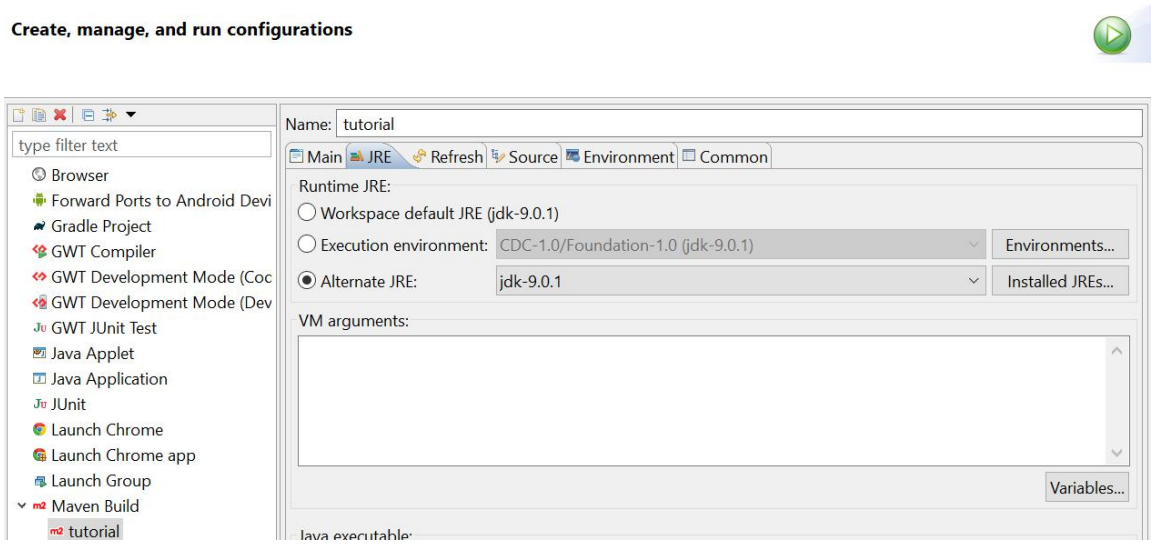

Figura 20: Modificando el Java Runtime Environment

A continuación con el botón derecho seleccionaremos gwt-material-tutorial, aparecerá un desplegable y de él debemos seleccionar **'Run As'**. Seleccionaremos las opciones : **Maven generate-sources, Maven install y Maven build .**

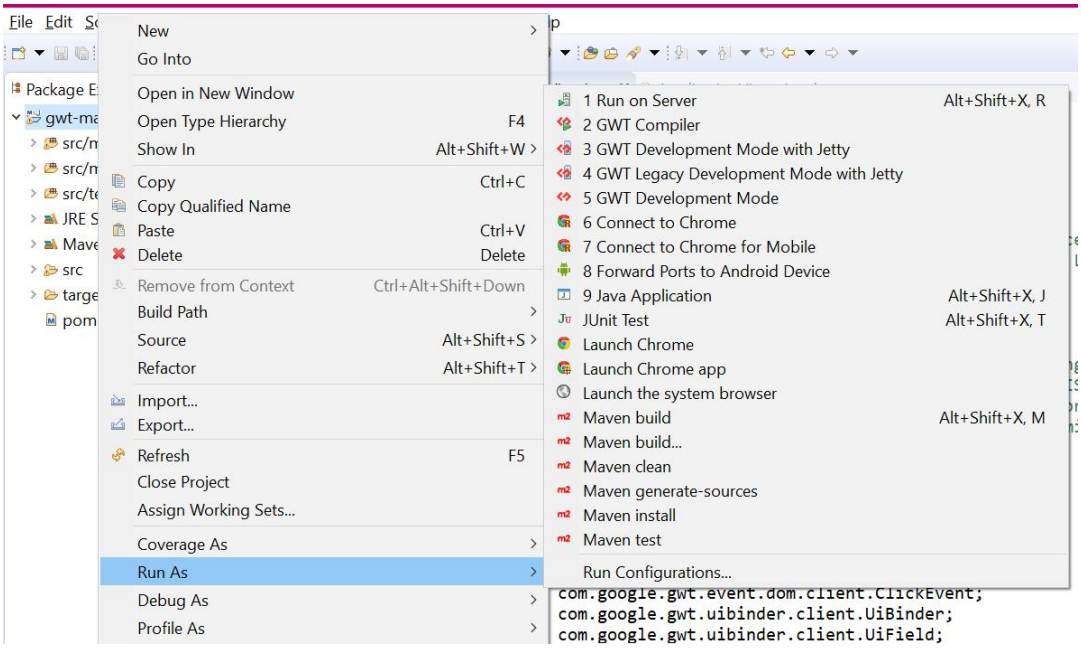

Figura 21: Generando, instalando y construyendo maven

A continuación si es todo correcto, nos saldrá una nueva ventana llamada "GWT Development Mode"

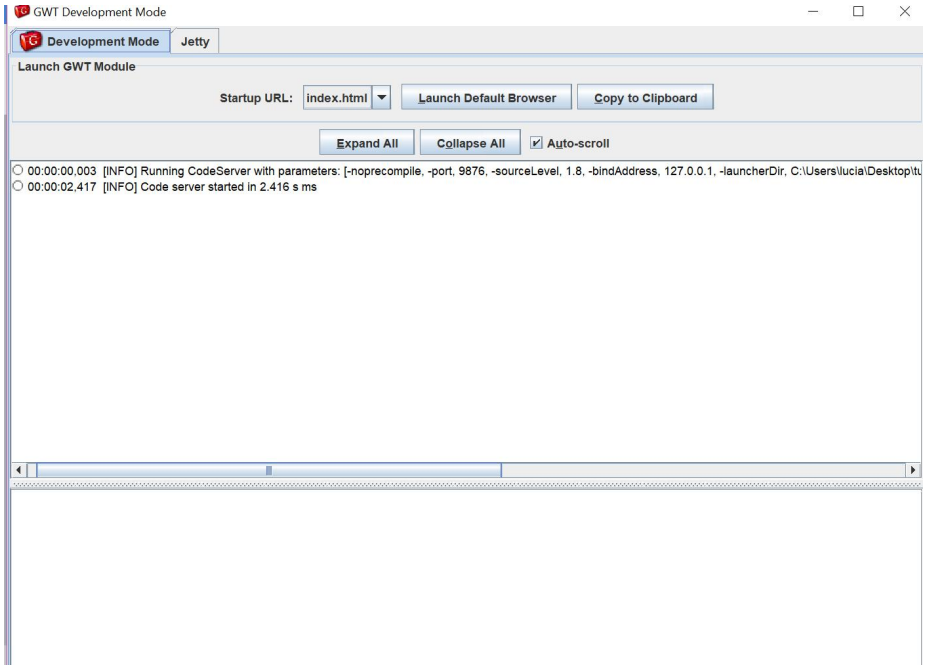

Figura 22: Development Mode

Pulsando la opción "Launch Default Browser" nos abrirá la aplicación directamente en el navegador.

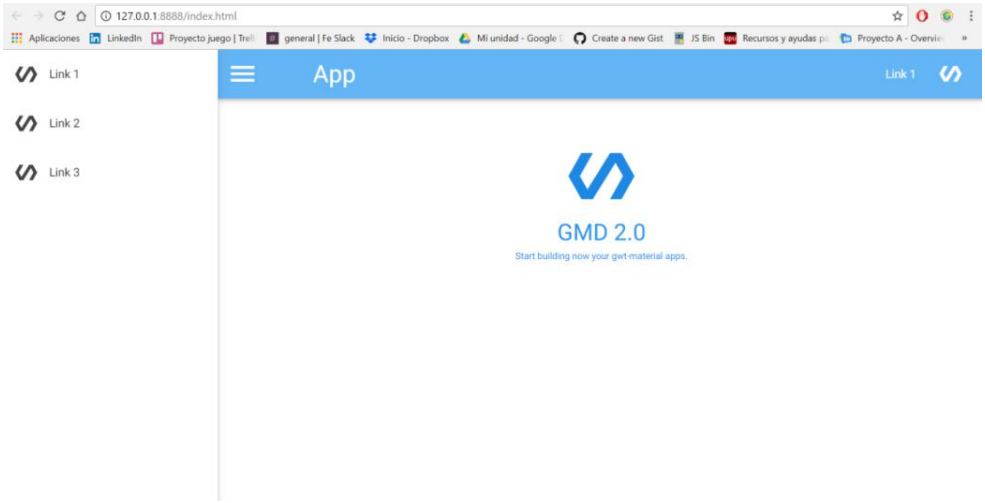

Figura 23: Resultado final del tutorial oficial

4. Creación del proyecto

En este punto se va a implementar lo que ya hemos instalado previamente, junto con lo visto en el apartado de guía de instalación del entorno de trabajo. Cuando nos aseguremos de que el resultado final del tutorial oficial nos funciona, podemos pasar a editar el código, tanto ApplicationView.java como ApplicationView.ui.xml, serán los ficheros que más modificaremos, junto a GwtMaterialBasic.gwt.xml, web.xml y

pom.xml para la configuración de varias páginas html, diferentes packages y servlets.

4.1. Descripción del proyecto realizado

En la siguiente figura podemos observar un diagrama del proyecto, con todas las rutas que dispone actualmente:

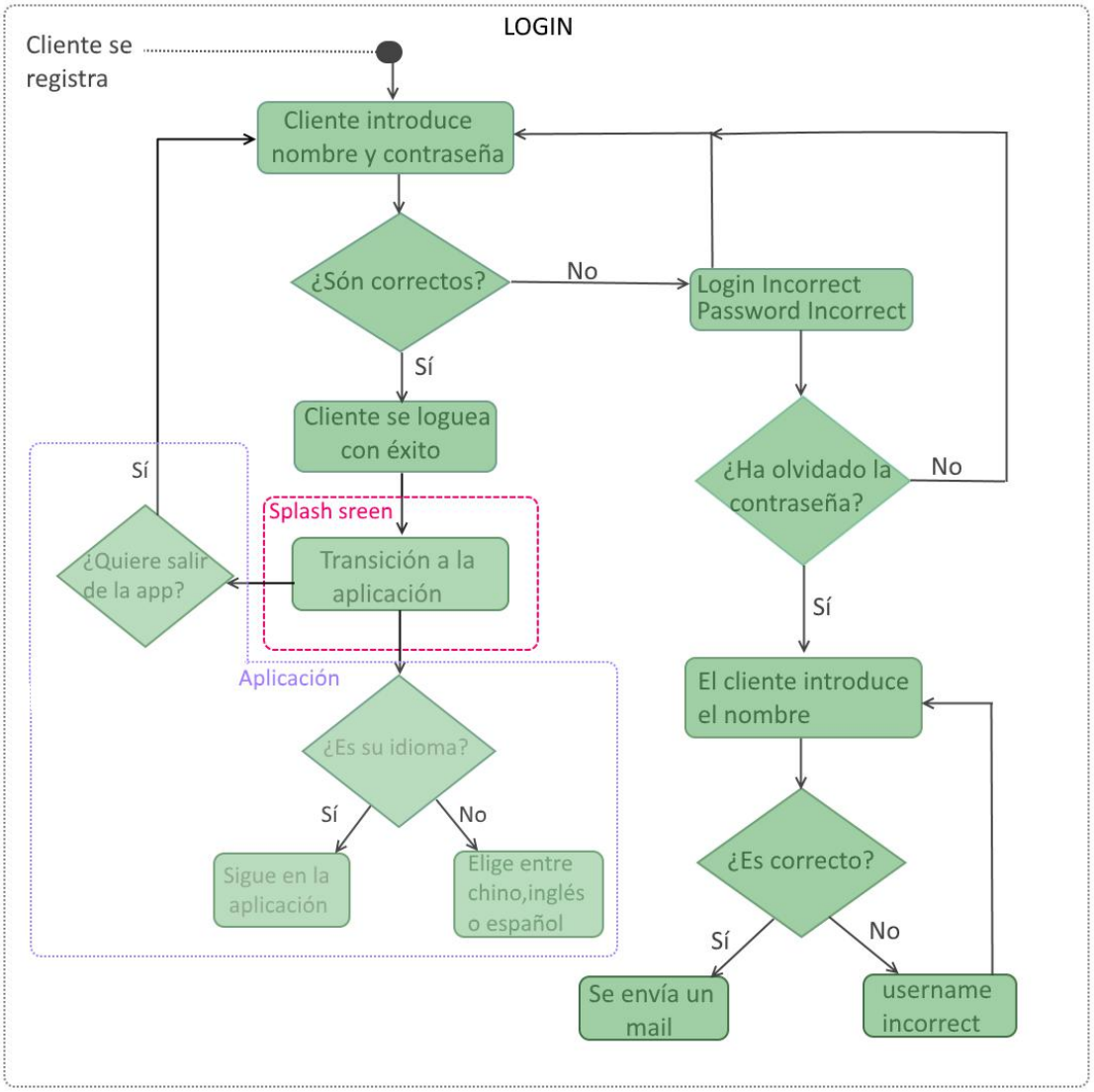

Figura 24: Diagrama del proyecto

En este proyecto, la empresa nos pedía que una vez que el usuario entrara a la página, después de haberse logueado con éxito, la aplicación le mostrara su o sus instalaciones:

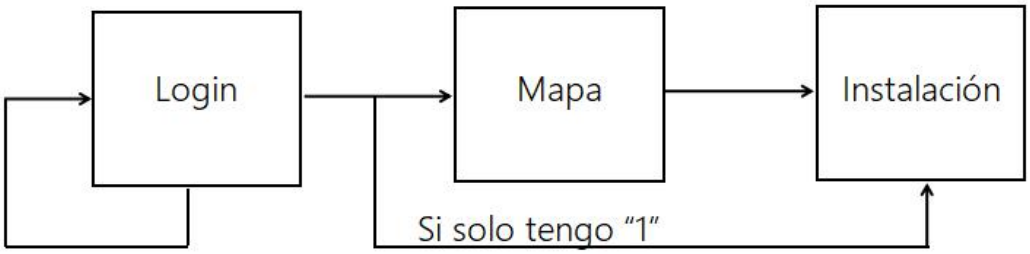

Figura 25: Esquema del proyecto

Teniendo ya en cuenta las especificaciones de la empresa, podemos distinguir tres partes:

- El **login**, que será la página inicial que dará acceso solo a los usuarios registrados en la base de datos, será capaz de almacenar la contraseña y comprobar si es un usuario registrado o no.

- La **splash screen**, o corta animación que de acceso a la aplicación.

- La **aplicación** que constará de un mapa en amcharts[24], esta plataforma se ha escogido debido a las especificaciones impuestas por la empresa. También estarán presentes un header, footer y un modulo widget que serán responsives e interactivos.

4.2. Diseño del proyecto

Para poder crear un nuevo proyecto debemos tener en cuenta la estructura para el diseño del mismo. Podemos comprobar que la estructura es muy similar a la programación en html, pues ambos tienen:

- ✓ Un **header (encabezamiento)**, es la parte superior del página. Generalmente formado por el logo de la empresa, nombre, una imagen o imágenes representativas.[25]

- ✓ **Body** en el caso de html y **Container** en caso de gwt, en ambos casos es el cuerpo y representa el contenido de la web. Sólo puede haber un elemento body en un documento, en cambio si que pueden haber varios container.[26]

- ✓ **Widget** en el caso de html y **SideNav** en caso de gwt, en ambos casos es una columna en el lado derecho o izquierdo de la web. Suele contener enlaces de acceso rápido u otros enlaces de interés.

- ✓ **Un footer (pie de página),** representa un pie de página típicamente contiene información acerca de el autor. datos de derechos de autor o enlaces a documentos relacionados. [27]

Estas partes son mostradas a continuación en la siguiente figura:

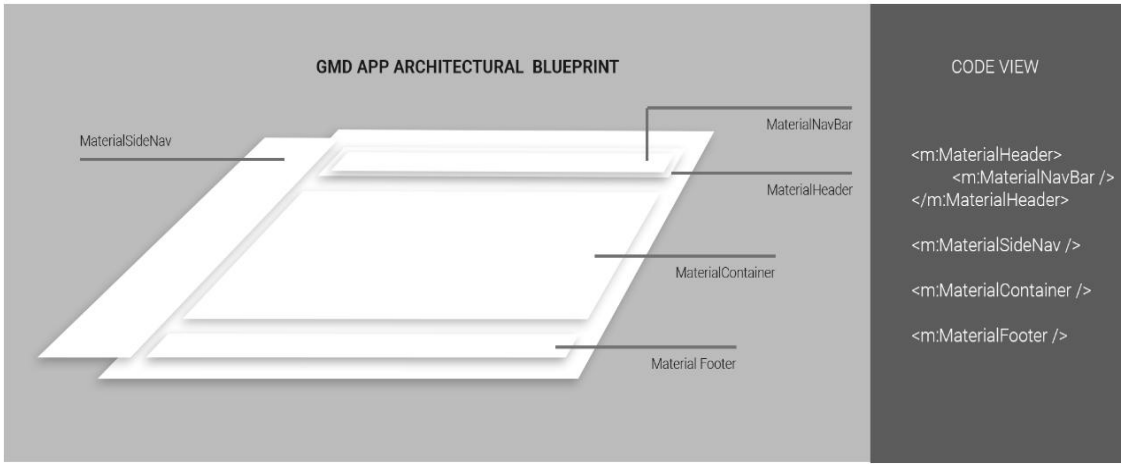

Figura 26: Estructura básica para el diseño de aplicaciones GMD.[28]

4.3.Implementación del proyecto

Para crear un nuevo proyecto emplearemos el método que hemos usado previamente para crear el tutorial. En primer lugar ejecutamos el símbolo del sistema(cmd) ,a continuación buscamos con los comandos dir y cd la carpeta donde queramos crear el proyecto. Cuando estemos en ella ejecutaremos los siguientes comandos:

```
mvn archetype:generate -DarchetypeGroupId=com.github.gwtmaterialdesign -DarchetypeArtifactId=gwt-material-archetype -DarchetypeVersion=2.0

Define value for property 'groupId': com.gwtmaterial.proyecto
Define value for property 'artifactId': gwt-material-proyecto

Define value for property 'package' com.gwtmaterial.tutorial: : com.gwtmaterial.proyecto

Y por último antes de generar el proyecto nos pedirá una confirmación:
Y: : Y
```

```
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: gwt-material-archetype:2.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: com.gwtmaterial.proyecto
[INFO] Parameter: artifactId, Value: gwt-material-proyecto
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.gwtmaterial.proyecto
[INFO] Parameter: packageInPathFormat, Value: com/gwtmaterial/proyecto
[INFO] Parameter: package, Value: com.gwtmaterial.proyecto
[INFO] Parameter: gwt-material-version, Value: 2.0
[INFO] Parameter: groupId, Value: com.gwtmaterial.proyecto
[INFO] Parameter: artifactId, Value: gwt-material-proyecto
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: C:\Users\lucia\Desktop\Nueva carpeta\gwt-material-proyecto
[INFO] ----------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ----------------------------------------------------------------------------
[INFO] Total time: 42.565 s
[INFO] Finished at: 2018-01-04T10:40:52+01:00
[INFO] Final Memory: 18M/60M
[INFO] ----------------------------------------------------------------------------
```

Figura 27: Creación de un nuevo proyecto

A continuación importamos el proyecto a Eclipse. Mediante la opción de **Import>Maven>Existing Maven Projects.** Le especificamos el directorio en el que se encuentra el proyecto creado, asegurándonos que seleccione el archivo **'pom.xml'**.

Una vez importado el proyecto deberá tener este aspecto:

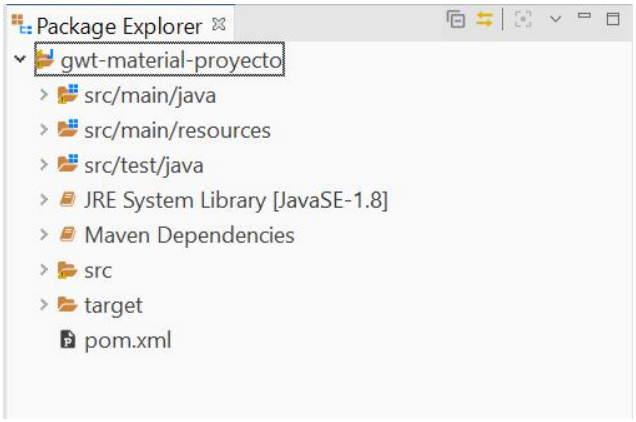

Figura 28: Estructura de el nuevo proyecto

A partir de este momento podemos ir modificando el proyecto, fijándonos sobretodo en los archivos **ApplicationView.java** y **ApplicationView.ui.xml.**

En estos ficheros empezamos a poner nuestro código, una vez que hemos decidido el diseño y estructura que vamos a seguir.

4.4. Estructura del proyecto

4.4.1. Login

Lo que contiene: Este sistema de entrada consta de un sistema de autentificación, en el cual solo los clientes de la empresa con cuentas ya creadas podrán acceder a la

aplicación introduciendo su nombre de usuario y contraseña, también disponen de un sistemas de cookies enlazado al checkbox de "REMEMBER ME" en el que al pulsarlo y entrar recordará sus datos hasta la fecha de expiración de las cookies, la cual hemos fijado en 1296000000 minutos. Si no se pulsa el checkbox al salir de la aplicación no se guardarán los datos y se deberán de poner nuevamente.

Además de las cookies este login consta del botón "LOST PASSWORD". El cual nos abrirá un modal para poder recuperar la contraseña, en caso de haberla olvidado, a través de un correo que la empresa enviará en caso de que coincida con algún correo de los clientes que la empresa tenga registrados previamente en la base de datos.

Visualización:

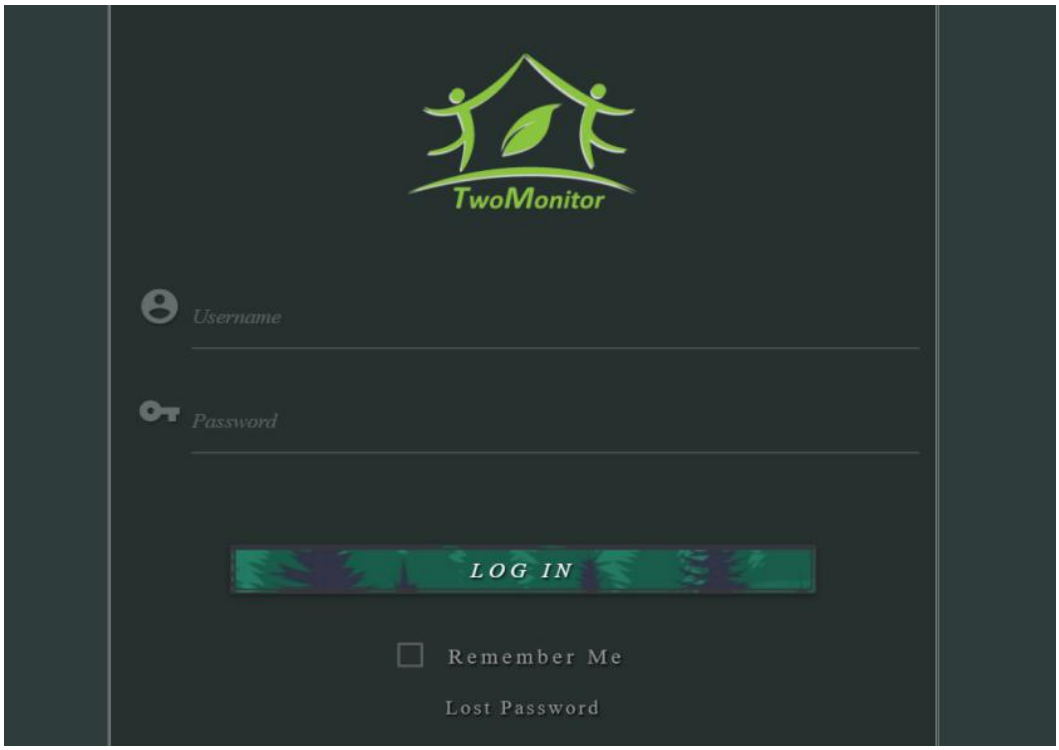

Figura 29:Visualización login

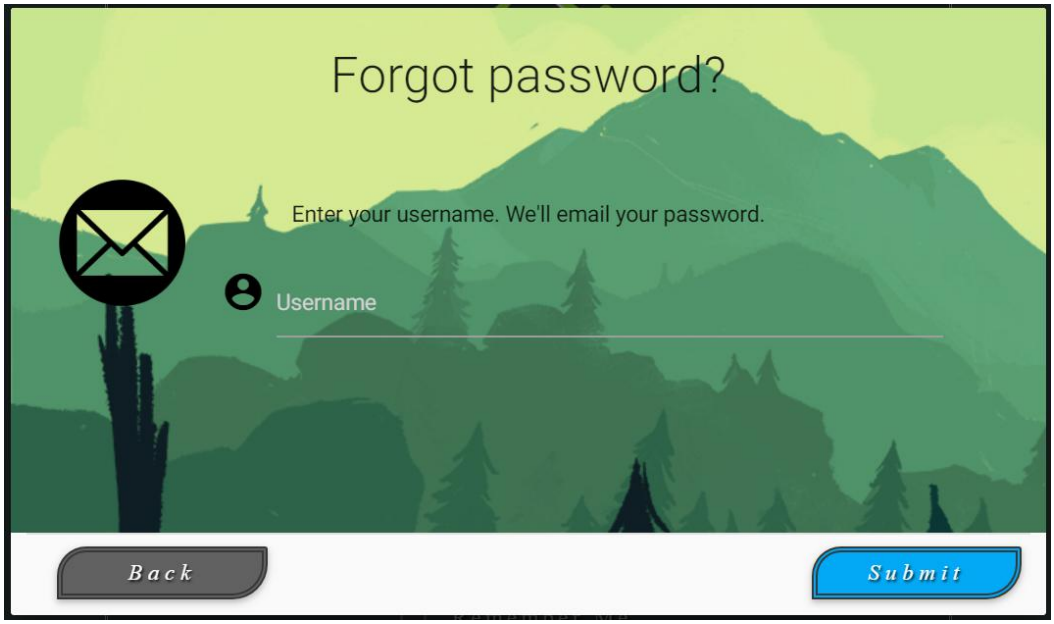

Figura 30: Visualización de modal para recuperar la contraseña

4.4.2. Splash screen

Lo que contiene: Se trata de una simple transición que ocurre cuando el usuario ha colocado su identifiación de manera correcta e intenta entrar a la aplicación con el botón de "LOG IN".

Visualización:

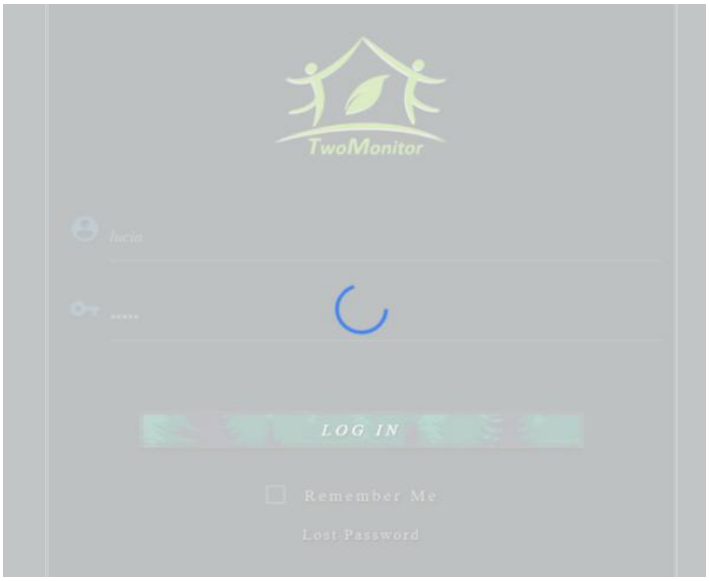

Figura 31:Visualización Splash screen

4.4.3. Aplicación

Lo que contiene: La aplicación consta de un mapamundi, una gráfica que muestra las pérdidas y el correcto funcionamiento de las instalaciones[29], y además una tabla que muestra en porcentajes las pérdidas y el correcto funcionamiento de las instalaciones[30]. Si se emplea el botón para alejar y/o acercar del mapa se puede observar la cantidad de instalaciones que posee el usuario y el lugar donde están ubicadas. También en el menú desplegable de la izquierda se encuentra una tabla con la información de las instalaciones.

Además en el header, la barra superior, hay un botón para salir (LOG OUT), que te lleva al login y un botón desplegable (dropdown), está la opción de cambiar de idioma, siendo Inglés Español y Chino los disponibles.

Visualización:

Figura 32: Visualización App(1)

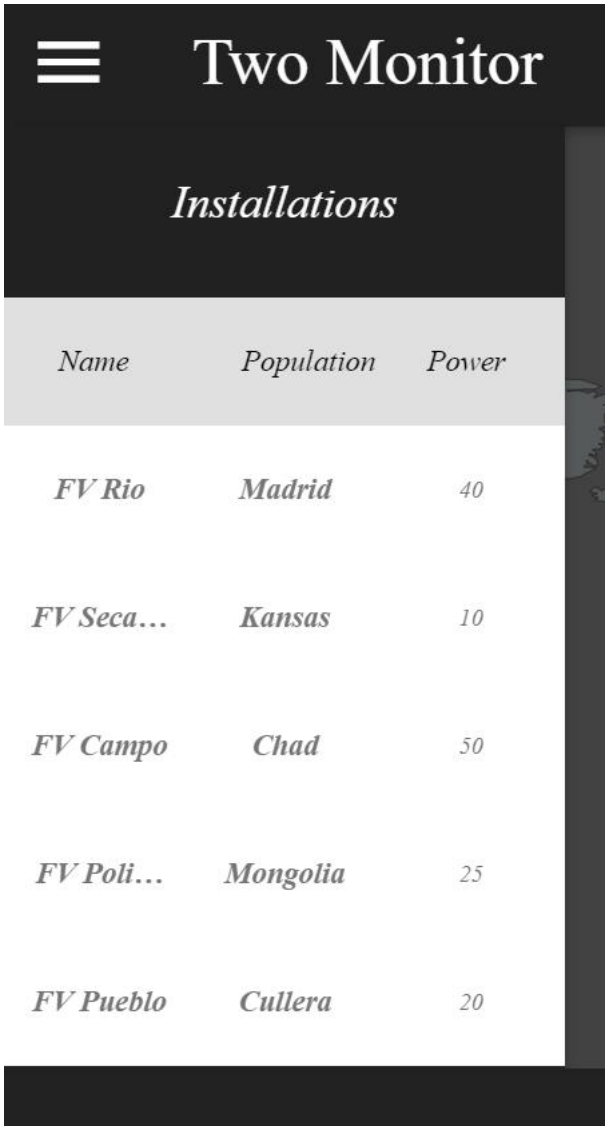

Figura 33: Visualización App(2)

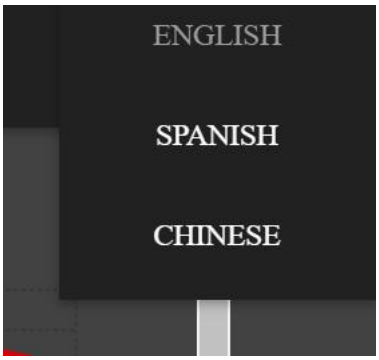

Figura 34: Visualización App(3)

4.5. Librerías empleadas

Para su correcta implementación, tendremos que añadir las librerías a nuestro proyecto. A continuación de descargarnos las librerías las pondremos en el proyecto. Mediante el botón derecho en target>WEB-INF>lib, y seleccionar build path> configure build path.

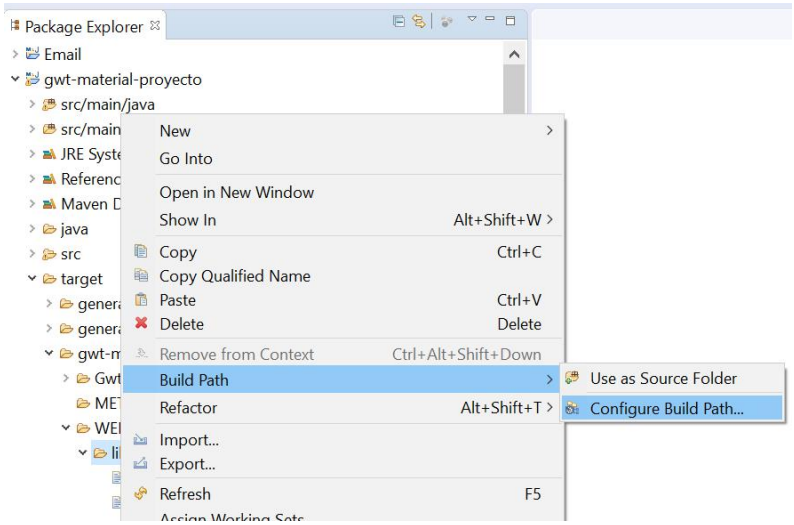

Figura 35: Configurar las librerías en el proyecto

Cuando nos aparezca la ventana de Properties for gwt-material-proyecto ,debemos seleccionar la pestaña de Libraries. En esta pestaña podremos ver las librerías actuales del proyecto y añadir las librerías que queramos mediante Add JARs

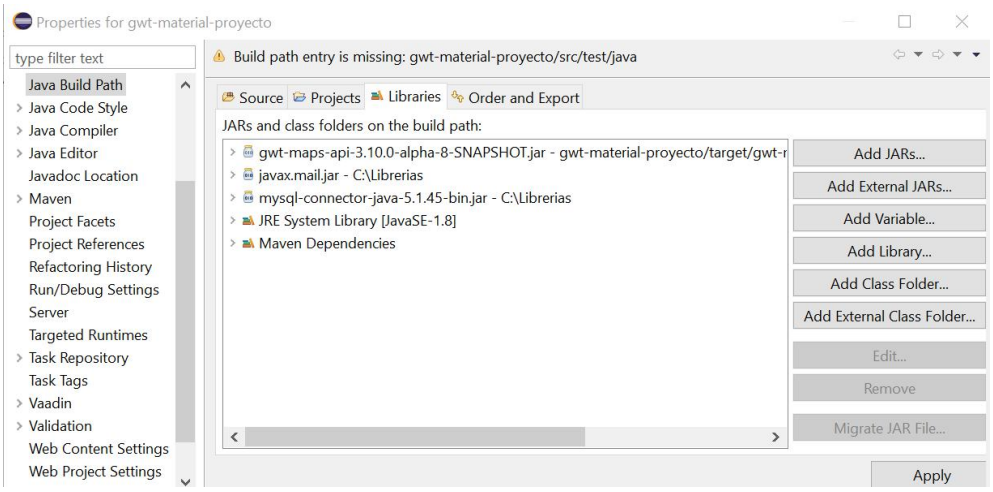

Figura 36: Ver e insertar librerías en el proyecto

Además de instalar las librerías, también deberemos ponerlas en WEB-INF>lib. Estas librerías deben estar reflejadas con maven repository[31] en el pom.xml .

Estas son las librerías que hemos utilizado:

Javax.mail-1.6.1.jar[32] , el archivo jar de referencia JavaMail, incluye los proveedores de protocolo SMTP, IMAP y POP3 [33]

gwt-servlet-2.8.2.jar[34], necesaria para dar soporte al protocolo de comunicación RPC[35]

gwt-material-themes-2.0.jar[36], necesaria para importar los estilos predefinidos [37]

gwt-material-table-2.0.jar[38] , necesaria para poder utilizar tablas en el proyecto[39]

gwt-material-amcharts-2.1-SNAPSHOT.jar[40] , necesaria para importar AmCharts a gwt-material, y así poder utilizar los gráficos y mapas de AmCharts en el proyecto.[41]

gwt-material-addins-2.0.jar[42] ,necesaria para poder utilizar el cambio de ventanas, las animaciones, el menú, y algunas utilidades más que hemos utilizado en nuestro proyecto.[43]

gwt-material-2.0.1.jar[44] , versión en la que esta hecho el proyecto ya que es la última versión finalizada[45]

Activation-1.1.jar[46], necesaria pues contiene el marco de activación[47]

dom4j-1.6.1.jar[48], se trata de una librería de código abierto fácil de usar para trabajar con XML, XPath y XSLT en Java utilizando Java Collections Framework y con soporte completo para DOM, SAX y JAXP[49]

hibernate-commons-annotations-5.0.1.Final.jar[50] , es un proyecto de utilidad utilizado por varios proyectos de Hibernate, no lo hemos utilizado directamente. Su primer objetivo es dar soporte a los descubrimiento de tipos de Java Generics. Su segundo objetivo es respaldar la anulación de anotaciones Java a través de archivos XML[51]

hibernate-core-5.2.16.Final.jar[52] , es donde esta definida la funcionalidad ORM[53]

hibernate-jpa-2.1-api-1.0.0.Final.jar[54], este es el JAR que contiene la API, proporciona todas las interfaces y clases concretas que la especificación define como API pública. Dicho de otro modo, puede usar este JAR para arrancar cualquier implementación de proveedor JPA. [55]

hibernate-testing.jar[56] , empleada para testear la base de datos de la aplicación[57]

mysql-connector-java-5.1.15-bin.jar[58] , se trata de la librería que permite acceder mediante java a la base de datos mysql[59]

4.6. MySQL Workbench y MySQL server

En primer lugar para poder crear una base de datos y el servidor usaremos los recursos de MYSQL[60] Workbench y MYSQL server[61]. En este caso se ha escogido esta solución porque de esta manera podemos tener servidor[62] con un programa único ejecutándose en segundo plano y con el MySQL Workbench modificaremos e interactuaremos con la base de datos.

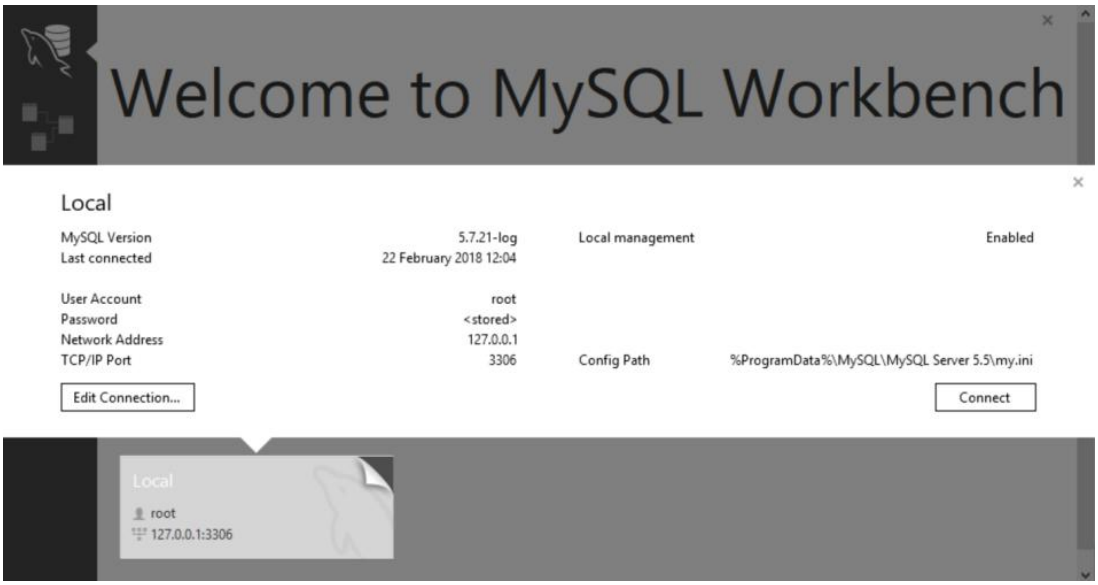

Figura 37: Creación de una base de datos Local

Si nos colocamos para ver el diseño de la base de datos podemos ver todas las interacciones entre clases y requisitos

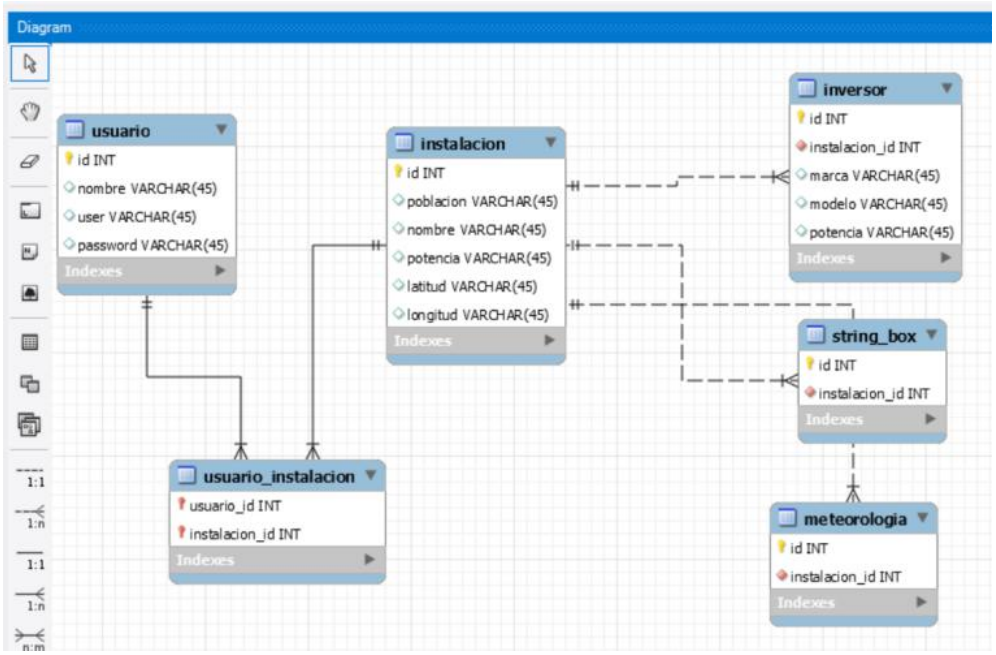

Figura 38: Visualización de la base de datos

Para poder rellenar la base de datos o ver lo que contiene, entraremos en Local con un doble click. Una vez dentro en SCHEMAS buscaremos en este caso "Twomonitor", a continuación desplegaremos las tablas que contiene y nos fijaremos en este caso en "instalacion".Con el botón derecho seleccionaremos "Select Rows - Limit 1000 ". A continuación nos iremos situando sobre cada columna y la iremos rellenando. En este caso, las instalaciones serán los datos que aparecerán en la tabla de la aplicación.

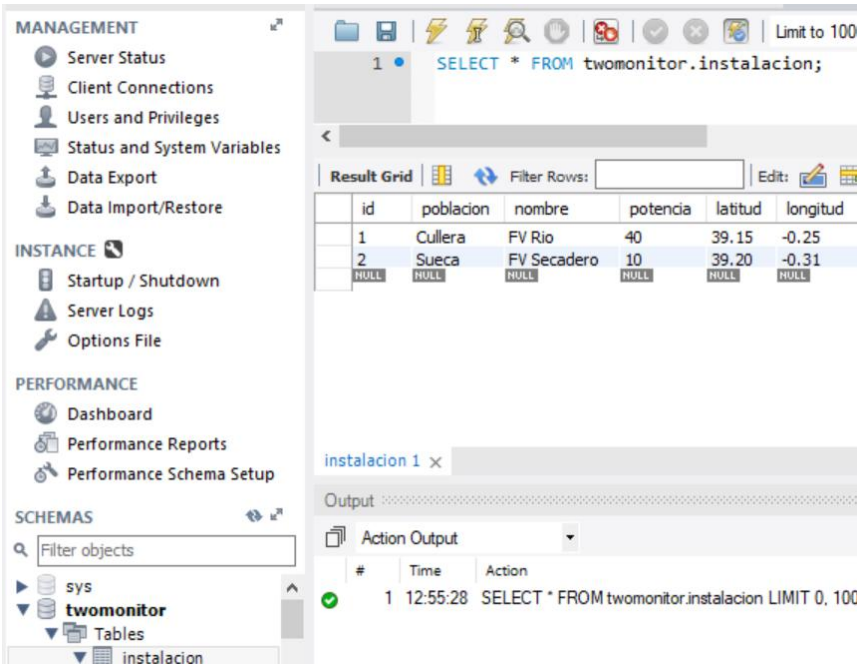

Figura 39: Visualización del interior de la base de datos

Definimos las siguientes interacciones entre clases de tal forma que, un único usuario pueda tener varias instalaciones, a la vez que cada instalación posea un inversor, un string_box y una clase denominada meteorología. De esta manera desde usuario podremos acceder a cualquier dato de la base de datos.

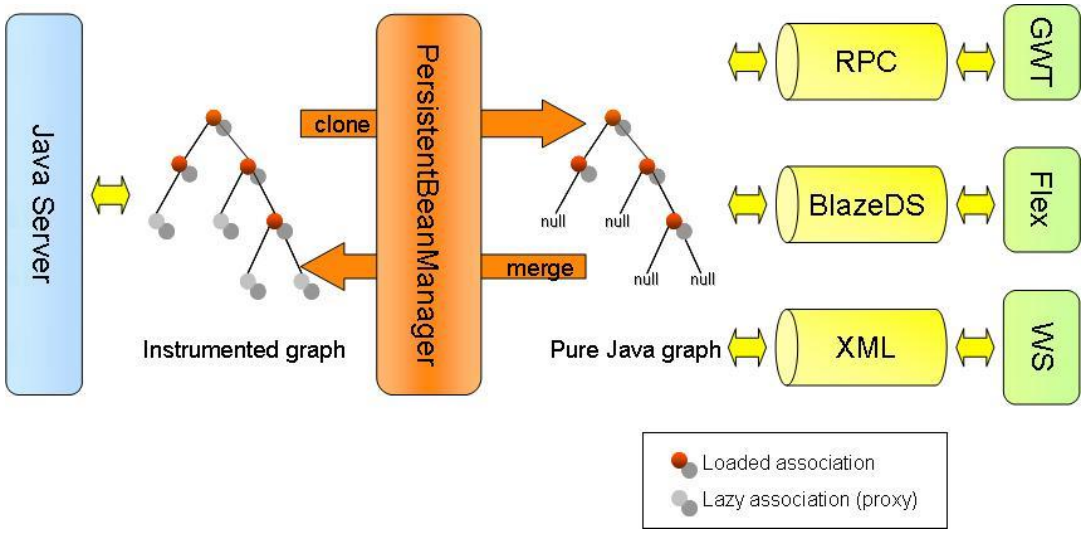

Figura 40: Esquema de un proyecto[63]

En este proyecto no lo tendremos en cuenta, debido a que se maneja una base de datos pequeña, pero para base de datos más extensas el método de con un usuario se pueda acceder a todos los datos de la base de datos provoca lucha latencia.

Esta latencia sería debida a la búsqueda de entre todos los usuarios de la base de datos, la extracción de todos los datos que tenga el usuario, que en nuestro caso irían hasta los inversores, string_box y meteorologías que poseen cada instalación que tenga el usuario. Para evitar la latencia respecto a este tema es conveniente emplear persistence en los objetos de hibernate. De esta manera conseguimos obtener solo lo que necesitamos en cada momento.

4.7. Hibernate

Hibernate es una herramienta de mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Hibernate[64] es software libre, distribuido bajo los términos de la licencia GNU LGPL.[65]

En primer lugar pasaremos a instalarlo buscándolo en Eclipse "*Help>Eclipse MarketPlace"* buscaremos y descargaremos los siguientes plugins:

- JBoss Tools Usage Reporting
-Hibernate Tools
-Jboss Maven Hibernate Configuration
-Jboss Maven Integration

A continuación vamos a *"Src/main/java >new> others> Hibernate> Hibernate Configuration File (cfg.xml)"*

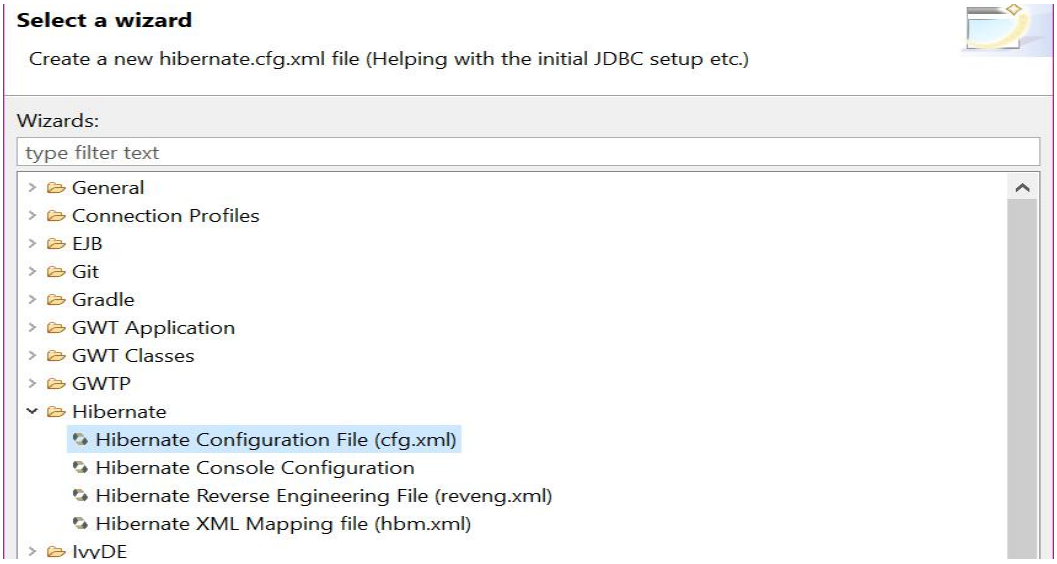

Figura 41: Configuración de Hibernate

A continuación le indicamos la ruta de java de nuestro proyecto para después ponerle la connection url, que es la dirección de nuestra base de datos. También nos aseguraremos de añadir en ese momento, también podremos añadirlos más tarde, los archivos hbm.xml de cada clase POJO, de esta manera podremos comunicarnos con la base de datos a través del mapping que hará hibernate empleando estos archivos. La contraseña y nombre de usuario que vayamos a usar en la base de datos en MySQL Workbench y MYSQL server. Una vez terminado nos quedará de esta manera en el proyecto:

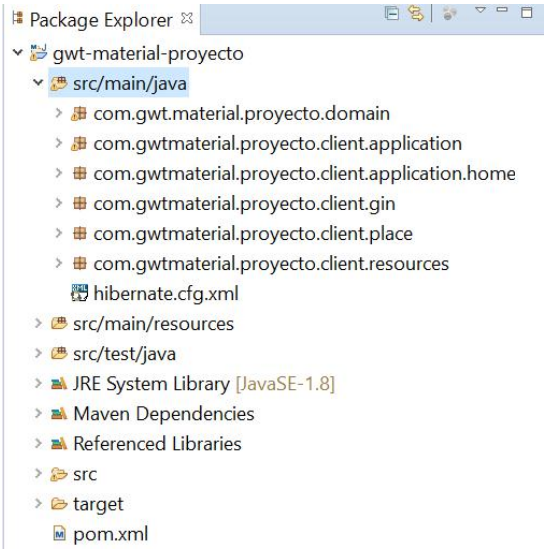

Figura 42: Visualización de Hibernate en Eclipse

Si entramos dentro de hibernate.cfg.xml lo que vemos es la configuración que le acabamos de indicar anteriormente.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">org.gjt.mm.mysql.Driver</property>
        <property name="hibernate.connection.password">[redacted]</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost/twomonitor</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>
    </session-factory>
</hibernate-configuration>

```

Figura 43:Visualización de la configuración de Hibernate

A continuación vamos a *"Scr/main/java> new> package"* y creamos el package domain, que es donde irán nuestras clases java tipo POJO.

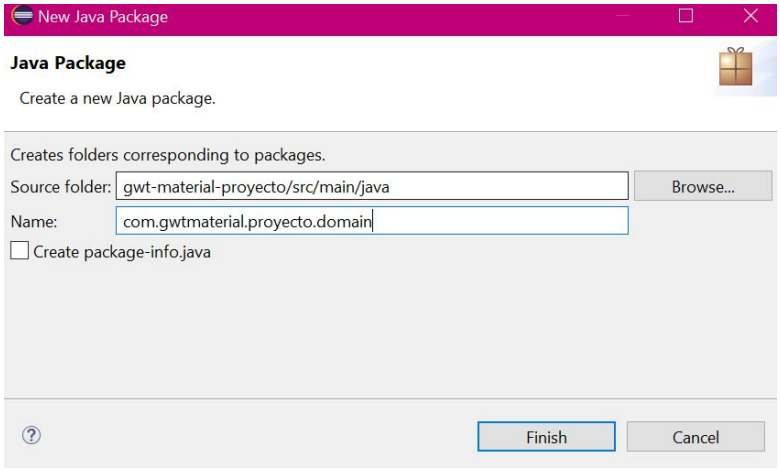

Figura 44: Creación del package domain

A continuación vamos a *"Src/main/resources > com> gwtmaterial> proyecto> GwtMaterialBasic.gwt.xml" y le añadimos "<source path='domain'>" para que pueda leer e identificar el package domain.*

Después vamos a *"Hibernate Code Generation Configurations..."* donde creamos una nueva configuración dándole doble click, y tras indicarle la ruta y el package de donde queremos que vaya le daremos a run, de esa manera nos generará las clases java tipo POJO creadas anteriormente en la base de datos.

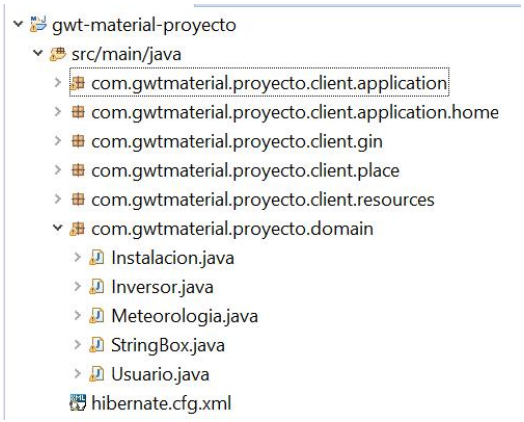

Figura 45: Clases POJO

De esta manera ya podremos referenciar el código de la aplicación a la base de datos.

4.8. Servicios

A continuación, para poder acceder a la base de datos debemos crear servicios. En nuestro caso los crearemos para el Login y para mandar un correo si al usuario se le ha olvidado su contraseña.

En primer lugar emplearemos el protocolo de comunicación RPC, el cual consiste en llamadas a procedimientos remotos. Gracias a AJAX, podemos actualizar solamente una parte, la parte deseada, de una web, no es necesaria refrescar toda la página web.[66]

A continuación crearemos el interfaz síncrono llamado LoginServicio, el cual extiende RemoteService. Esta interfaz síncrona es la versión definitiva de las especificaciones del servicio. Cualquier implementación de este servicio en el lado del servidor debe extender RemoteServiceServlet e implementar esta interfaz de servicio.

Seguidamente, antes de que podamos intentar realizar una llamada remota desde el cliente, crearemos la interfaz asíncrona llamada LoginServicioAsync, basada en la interfaz de servicio original, LoginServicio.

La naturaleza de las llamadas al método asíncrono requiere se le pase un objeto callback que puede ser notificado cuando se completa una llamada asíncrona, ya que por definición no puede ser bloqueado hasta que la llamada finalice. Por la misma razón, los métodos asíncronos no tienen tipos de devolución; generalmente vuelven vacíos. Si se desea tener más control sobre el estado de una solicitud pendiente, se tiene que devolver la Solicitud en su lugar. Después de que se realice la llamada asíncrona, toda la comunicación va de vuelta a la persona que ha realizado la llamada a través del objeto callback que se ha pasado anteriormente.

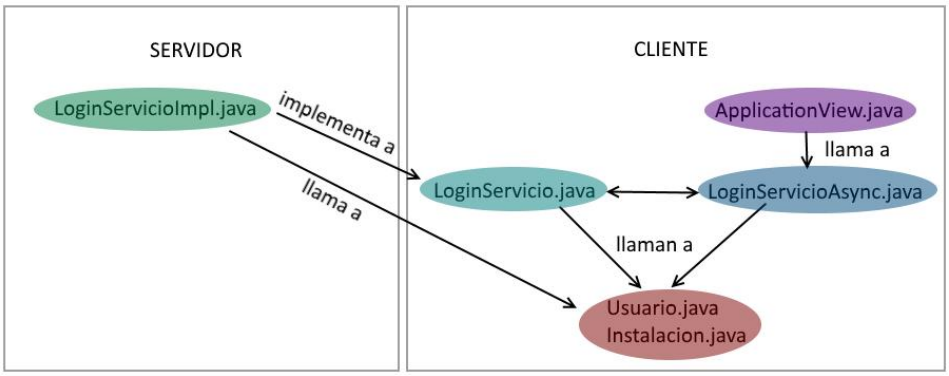

Figura 46: Estructura del servicio de login

A continuación, testearemos los servicios durante el desarrollo. El modo de desarrollo GWT incluye una versión integrada de Jetty que actúa como un contenedor de servlets. De esta manera, es capaz de depurar tanto el código del lado del servidor como el código del lado del cliente, cuando ejecuta su aplicación en modo de desarrollo usando un depurador Java. Para cargar automáticamente la implementación del servicio, se debe configurar los servlets necesarios en el archivo web.xml.

```
<!-- Servlet -->
<servlet>
<servlet-name>LoginServicioImpl</servlet-name>
<servlet-class>com.gwtmaterial.proyecto.server.LoginServicioImpl</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>LoginServicioImpl</servlet-name>
<url-pattern>/GwtMaterialBasic/LoginServicio</url-pattern>
</servlet-mapping>
```

Cuando probemos el código del lado del cliente y del lado del servidor en modo de desarrollo, debemos asegurarnos de colocar una copia de gwt-servlet.jar en el directorio war> WEB-INF > lib y asegurarnos de que el directorio de salida Java esté configurado a war > WEB-INF > classes. De lo contrario, el servidor Jetty incrustado no podrá cargar nuestros servlet correctamente.

4.8. Cronograma del proyecto

Acontinuación vamos a mostrar la duración del proyecto y el camino óptimo, el cual buscará el camino que menos días emplee, con la ayuda de los diagramas de Gantt y Roy.

Tareas	noviembre	diciembre					enero			febrero				marzo	abril		mayo
	8	10	15	16	18	19	10	14	15	20	10	22	26	13	17	26	14
Reunión con la empresa																	
Confirmación del proyecto																	
Investigar y configurar las herramie																	

ntas que se vayan a emplear																	
Login																	
Splash screen																	
Aplicación																	
Integración BBDD																	
Testeo																	
Reunir documentación																	
Redactar la memoria																	

Tabla 2: Diagrama de Gantt

Tareas	Número	Antecesor	Predecesor	Duración
Reunión con la empresa	1	-	2,3,6,7	1 día
Confirmación del proyecto	2	1	3	1 día
Investigar y configurar las herramientas que se vayan a emplear	3	1	4.1	5 días
Login	4.1	3	4.1	3 días
Splash screen	4.2	4.1	4.2	1 día
Aplicación	4.3	4.2	4.3	113 días
Integración BBDD	4.4	4.3	4.4	78 días
Testeo	5	4.4	-	116 días
Reunir documentación	6	1	7	117 días
Redactar la memoria	7	1,6	-	122 días

Tabla 3: Diagrama de Roy

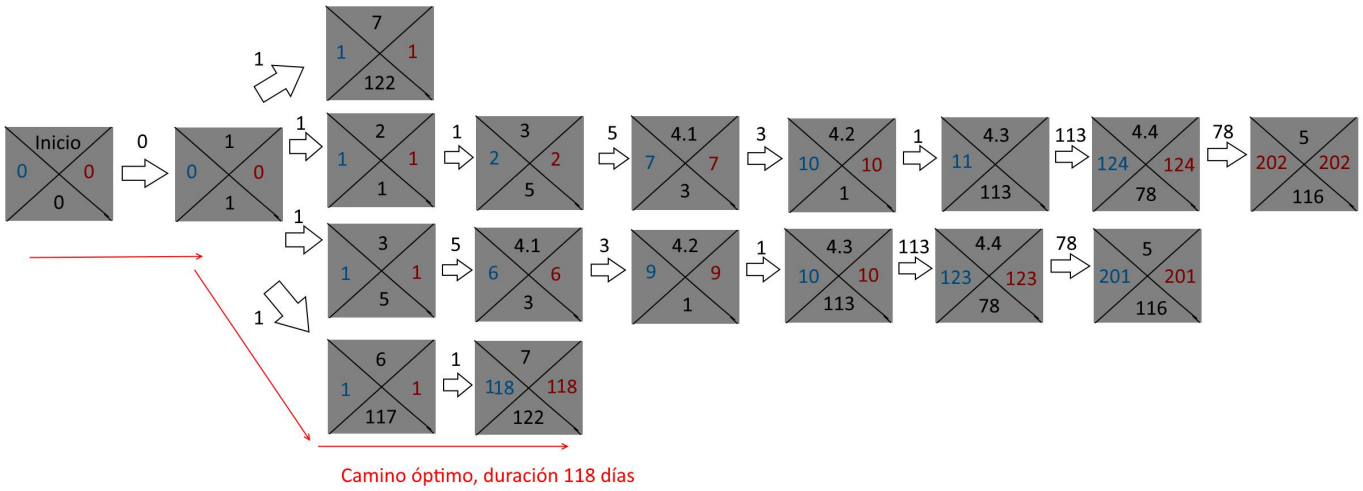

Figura 47: Diagrama de Roy, camino óptimo

5. Conclusiones

Podemos verificar que el proyecto se ha realizado con éxito, pues se han cumplido con las especificaciones que la empresa daba.

Los administradores disponen de una aplicación Web en la que puedan crear, gestionar, borrar y modificar las instalaciones y usuarios de una manera sencilla. Usando la base de datos creada con MySql Workbench y la página html para acceder al js. Podemos concluir que este apartado ha sido superado con éxito, ya que después del testeo, no se ha detectado ninguna anomalía significativa.

El usuario tendrá una aplicación en la que tras loguearse, acceda a sus instalaciones, y pueda observar su estado en un mapa. Podemos concluir que este apartado ha sido superado con éxito, ya que después del testeo, no se ha detectado ninguna anomalía significativa.

También cabe mencionar que la página oficial gwtmaterialdesign[67] contiene información obsoleta, y se ha tenido que emplear diferentes lenguajes de programación tales como html, java, js, xml y css para continuar con el proyecto.

Como posibles ampliaciones y usos, podríamos añadir datos de producción fotovoltaica, un pop-up con una gráfica de la información individual de cada instalación. Y alguna animación y/o alerta para cuando la instalación tenga niveles muy bajos de potencia.

6. Bibliografía

[1] *TwoMonitor* <http://www.twomonitor.com/>[Consulta: 10 de diciembre de 2017]

[2] Apache Maven Project. <http://maven.apache.org/> [Consulta: 16 de diciembre de 2017]

[3] zeroturnaround <https://zeroturnaround.com/rebellabs/maven-cheat-sheet/> [Consulta: 16 de diciembre de 2017

[4] vikastechblog <https://www.vikastechblog.com/misc/maven/maven-build.html> [Consulta: 16 de diciembre de 2017

[5] YOUTUBE, "How To Install Tutorials - Installing Apache Maven [Windows]" en Youtube <https://www.youtube.com/watch?v=Nn8cmBVdYDs> [Consulta: 16 de diciembre de 2017]

[6] GitHub.<https://github.com/LearnLib/learnlib/wiki/Getting-Started-Building-Your-Own-Project> [Consulta: 16 de diciembre de 2017]

[7] Wikipedia<https://es.wikipedia.org/wiki/Apache_Ant>[Consulta: 10 de diciembre de 2017]

[8] ant.apache <https://ant.apache.org/>[Consulta: 10 de diciembre de 2017]

[9] Wikipedia <https://es.wikipedia.org/wiki/Gradle>[Consulta: 10 de diciembre de 2017]

[10] Quora <https://www.quora.com/What-is-Gradle-exactly-in-Android-Studio>[Consulta: 12 de diciembre de 2017]

[11] Solidsoft <https://solidsoft.wordpress.com/2015/03/09/spock-1-0-with-groovy-2-4-configuration-comparison-in-maven-and-gradle/>[Consulta: 12 de diciembre de 2017]

[12] slideshare <https://www.slideshare.net/HannoEmbregts/migrating-25k-lines-of-ant-scripting-to-gradle>[Consulta: 10 de diciembre de 2017]

[13] splessons <http://www.splessons.com/lesson/ant-vs-maven/>[Consulta: 10 de diciembre de 2017]

[14] *Eclipse* <https://www.eclipse.org/> [Consulta: 16 de diciembre de 2017]

[15] *Apache Maven Projects.* <https://maven.apache.org/download.cgi#>[Consulta: 16 de diciembre de 2017]

[16] *Eclipse* <http://www.eclipse.org/downloads/> [Consulta: 16 de diciembre de 2017]

[18] *Gwtproject.*<http://www.gwtproject.org/download.html> [Consulta: 16 de diciembre de 2017]

[19]GWTMaterial.<https://gwtmaterialdesign.github.io/gwt-material-demo/#gettingstarted> [Consulta: 16 de diciembre de 2017]

[20] *GitHub.*<https://github.com/GwtMaterialDesign/gwt-material-archetype> [Consulta: 16 de diciembre de 2017]

[21] Apache Maven Project. <https://maven.apache.org/guides/mini/guide-naming-conventions.html > [Consulta: 16 de diciembre de 2017]

[22] Apache Maven Project.<https://maven.apache.org/guides/introduction/introduction-to-the-pom.html#What_is_a_POM > [Consulta: 16 de diciembre de 2017]

[23] ORACLE.<http://www.oracle.com/technetwork/java/javase/downloads/jdk9-downloads-3848520.html > [Consulta: 16 de diciembre de 2017]

[24] Amcharts<https://www.amcharts.com/>[Consulta: 26 de abril de 2017]

[25] Super Diseño Web (2015) "Diseño web: ¿Qué es el header?" en Preguntas Frecuentes,2015.<http://www.superdweb.com.ar/preguntas-frecuentes/item/649-diseno-web-ique-es-el-header.html> [Consulta: 16 de diciembre de 2017]

[26] Developer.mozilla (2005) "<body>" en Referencia de Elementos HTML, 2016. <https://developer.mozilla.org/es/docs/Web/HTML/Elemento/body> [Consulta: 16 de diciembre de 2017]

[27] Developer.mozilla (2005) "footer" en Referencia de Elementos HTML, 2014. <https://developer.mozilla.org/es/docs/Web/HTML/Elemento/footer> [Consulta: 16 de diciembre de 2017]

[28] GWTMaterial.<https://gwtmaterialdesign.github.io/gwt-material-demo/#blueprint> [Consulta: 16 de diciembre de 2017]

[29] Chartist-js<http://gionkunz.github.io/chartist-js/examples.html>[Consulta:5 de abril de 2017]
[30] Tympanus<https://tympanus.net/codrops/css_reference/flexbox/>[Consulta: 5 de abril de 2017]

[31] mvnrepository<https://mvnrepository.com/>[Consulta: 14 de enero de 2018]

[32] mvnrepository<https://mvnrepository.com/artifact/com.sun.mail/javax.mail/1.6.1>[Consulta: 15 de enero de 2018]

[33] Javaee <https://javaee.github.io/javamail/>[Consulta: 10 de febrero de 2018]

[34] mvnrepository<https://mvnrepository.com/artifact/com.google.gwt/gwt-servlet/2.8.2>[Consulta: 15 de enero de 2018]

[35] Gwtproject <http://www.gwtproject.org/release-notes.html>[Consulta: 10 de febrero de 2018]

[36] mvnrepository<https://mvnrepository.com/artifact/com.github.gwtmaterialdesign/gwt-material-themes/2.0-rc4>[Consulta: 15 de enero de 2018]

[37] Github <https://github.com/GwtMaterialDesign/gwt-material-themes>[Consulta: 10 de diciembre de 2017]

[38] mvnrepository<https://mvnrepository.com/artifact/com.github.gwtmaterialdesign/gwt-material-table/2.0-rc5>[Consulta: 15 de enero de 2018]

[39] Github <https://github.com/GwtMaterialDesign/gwt-material-table>[Consulta: 15 de enero de 2018]

[40] GitHub<https://github.com/GwtMaterialDesign/gwt-material-amcharts>[Consulta: 26 de abril de 2017]

[41] GitHub <https://github.com/GwtMaterialDesign/gwt-material-amcharts/blob/master/README.md>[Consulta: 20 de enero de 2018]

[42] GitHub<https://github.com/GwtMaterialDesign/gwt-material-addins/releases/tag/gwt-material-addins-2.0>[Consulta: 7 de marzo de 2018]

[43] GitHub <https://github.com/GwtMaterialDesign/gwt-material-addins>[Consulta: 18 de diciembre de 2017]

[44] GitHub<https://github.com/GwtMaterialDesign/gwt-material>[Consulta: 16 de diciembre de 2017]

[45] GitHub <https://github.com/GwtMaterialDesign/gwt-material> [Consulta: 10 de diciembre de 2017]
[46] mvnrepository<https://mvnrepository.com/artifact/javax.activation/activation/1.1> [Consulta: 7 de febrero de 2018]

[47] Versioneye <https://www.versioneye.com/java/javax.activation:activation/1.1.1>[Consulta 15 de diciembre de 2017]

[48] mvnrepository <https://mvnrepository.com/artifact/dom4j/dom4j/1.6.1>[Consulta: 10 de enero de 2018]

[49] dom4j <http://www.dom4j.org/dom4j-1.6.1/>[Consulta: 10 de febrero de 2018]

[50] mvnrepository <https://mvnrepository.com/artifact/org.hibernate.common/hibernate-commons-annotations/5.0.1.Final>[Consulta: 26 de febrero de 2018]

[51] Github <https://github.com/hibernate/hibernate-commons-annotations> [Consulta: 26 de febrero de 2018]

[52] mvnrepository <https://mvnrepository.com/artifact/org.hibernate/hibernate-core/5.2.16.Final>[Consulta: 26 de febrero de 2018]

[53] Github <https://github.com/hibernate/hibernate-orm/blob/master/hibernate-core/hibernate-core.gradle>[Consulta: 26 de febrero de 2018]

[54] mvnrepository <https://mvnrepository.com/artifact/org.hibernate.javax.persistence/hibernate-jpa-2.1-api/1.0.0.Final>[Consulta: 26 de febrero de 2018]

[55] docs.jboss <https://docs.jboss.org/hibernate/entitymanager/3.6/reference/en/html_single/> [Consulta: 26 de febrero de 2018]

[56] mvnrepository <https://mvnrepository.com/artifact/org.hibernate/hibernate-testing>[Consulta: 26 de febrero de 2018]

[57] Dzone <https://dzone.com/articles/testing-databases-junit-and>[Consulta: 26 de febrero de 2018]

[58] mvnrepository <https://mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.15> [Consulta: 22 de febrero de 2018]

[59] Versioneye <https://www.versioneye.com/java/mysql:mysql-connector-java/5.1.15> [Consulta: 13 de marzo de 2018]

[60] MySQL<https://dev.mysql.com/downloads/workbench/>[Consulta: 22 de febrero de 2018]

[61] MySQL <https://dev.mysql.com/downloads/file/?id=474803>[Consulta: 22 de febrero de 2018]

[62] Lineadecodigo.com <http://lineadecodigo.com/java/conectar-mysql-java/>[Consulta: 14 de mayo de 2018]

[63]Gwtproject.org <http://www.gwtproject.org/articles/using_gwt_with_hibernate.html>[Consulta: 17 de abril de 2018]

[64] HIBERNATE <http://hibernate.org/> [Consulta: 26 de febrero de 2018]

[65] Wikipedia. <https://es.wikipedia.org/wiki/Hibernate>[Consulta: 26 de febrero de 2018]

[66] Gwtproject <http://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideRemoteProcedureCalls> [Consulta: 20 de enero de 2018]

[67] Gwtmaterialdesign.github <https://gwtmaterialdesign.github.io/gwt-material-demo/#gettingstarted>[Consulta: 10 de diciembre de 2017]

Clodoaldo Robledo, P. et al. (2006). Google Web Toolkit. Gobierno de España. Ministerio de educación

Luciano Fiandesio, P. et al. (2007). GWT in Action Easy Ajax with the Google Web Toolkit. Estados Unidos de América: Manning Publications Co.