



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**GII 20.09 Herramienta web
repositorios de TFGII
Documentación Técnica**



Presentado por David Renedo Gil
en Universidad de Burgos — 9 de noviembre
de 2022

Tutor: Álvaro Arnaiz González y Ana Serrano
Mamolar

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	iv
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Planificación temporal	3
A.4. Estudio de viabilidad	3
Apéndice B Especificación de Requisitos	5
B.1. Introducción	5
B.2. Objetivos generales	5
B.3. Catalogo de requisitos	5
B.4. Especificación de requisitos	5
Apéndice C Especificación de diseño	7
C.1. Introducción	7
C.2. Diseño de datos	7
C.3. Diseño procedimental	7
C.4. Diseño arquitectónico	7
Apéndice D Documentación técnica de programación	9
D.1. Introducción	9
D.2. Estructura de directorios	9

D.3. Manual del programador	9
D.4. Compilación, instalación y ejecución del proyecto	16
D.5. Pruebas del sistema	24
Apéndice E Documentación de usuario	25
E.1. Introducción	25
E.2. Requisitos de usuarios	25
E.3. Instalación	25
E.4. Manual del usuario	25
Bibliografía	27

Índice de figuras

A.1. Gráfica Control chart- Sprint 0	3
D.1. Descarga de JDK 11	10
D.2. Descarga JDK 11 Licencia	11
D.3. Descargar IDE Eclipse	12
D.4. Seleccionar Eclipse	13
D.5. Eclipse marketplace	14
D.6. Plugin Vaadin	15
D.7. Copiar URL repositorio	16
D.8. Consola con Tomcat ejecutado	17
D.9. Gestor de Aplicaciones de Tomcat	18
D.10.Desplegar el archivo .war	18
D.11.Añadir servidor de Tomcat a Eclipse	19
D.12.Seleccionar carpeta contenedora de Tomcat	20
D.13.Añadir proyectos a servidor	21
D.14.Error tras desplegar el .war en el Gestor de Aplicaciones de Tomcat	22
D.15.Logs proporcionados por Tomcat	22
D.16.Logs proporcionados por Tomcat	23
D.17.Cambio de versión Dynamic Web Module	24

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En esta sección se detallará la planificación que se ha realizado, el estudio de viabilidad tanto de la parte económica, como temporal y de la legal.

A.2. Planificación temporal

Se nombrarán y explicarán brevemente las tareas realizadas a lo largo del proyecto. Estas tareas se encuentran en el [repositorio del proyecto en Github](#).

Se añadirán gráficas para una mejor comprensión del tiempo que ha supuesto cada tarea en los ciclos (*Sprints*).

Sprint 0 - Puesta a punto (5/10/22 - 19/10/22)

Puesta a punto del proyecto. Se procederá a plantear las herramientas con las que se va a trabajar, búsqueda de alternativas y toma de contacto con las herramientas nuevas que se van a emplear.

A continuación se detallarán las tareas que se realizaron durante este primer Sprint:

- Añadir la extensión ZenHub al navegador. Desde el **Chrome Web Store** de Google Chrome se añadió la extensión **ZenHub for GitHub**.

- Clonar en repositorio en local. Para clonarlo se ha utilizado la herramienta **Github Desktop**. Mediante en enlace **HTTP** que proporciona *Github*.
- Documentación sobre Vaadin. Se procederá a estudiar el *framework* Vaadin con el que se va a trabajar. A través de la página oficial de **Vaadin** se realiza la instalación en nuestro entorno IDE **Eclipse** y el aprendizaje.
- Instalación JDK 11 o superior. Para utilizar la última versión de Vaadin se descargará el **openjdk 17**.
- Importación de un proyecto Vaadin de prueba a Eclipse. Para probar el correcto funcionamiento de Vaadin descargaremos e importaremos el proyecto de **prueba**.
- Clonación e imitación del repositorio en Eclipse. Trataremos de clonar e imitar el funcionamiento de la versión **anterior del proyecto** sobre la que trabajamos. Posteriormente se descargará también el **openjdk 11** para tratar de clonar el repositorio que estaba en la anterior versión del proyecto. También debemos instalar la herramienta **Tomcat**.
- Comienzo de la documentación. Para ello hemos instalado las herramientas TexStudio y MikTex como se indica en **plantillaLatex** y se ha buscado información para iniciar la documentación.
- Actualización del README.md. Se modificó el README.md del proyecto para que refleje los cambios respecto a la versión anterior.
- Búsqueda de trabajos relacionados con la gestión de TFG/TFM. Se realizó una investigación con el fin de encontrar proyectos similares a la aplicación web, es decir, que consistan en la gestión de trabajos de fin de grado o similares. Los proyectos encontrados serán explicados en el apartado **Trabajos relacionados** de la memoria.

Sprint 1 - (19/10/22 - 2/11/22)

Se procederá a estudiar el código del repositorio y a documentar el anexo.

A continuación se detallarán las tareas que se realizaron durante este primer Sprint:

- Comienzo de la documentación del anexo. Comenzamos en este Sprint a realizar esta documentación desde TexStudio.

- Estudio del código de todos los paquetes de la carpeta src. Tanto persistence, como util, ui, security y webService.
- Se procede a buscar el error que salta al intentar ejecutar el código en local.

Se puede ver el transcurso de estas tareas en la ilustración A.1.

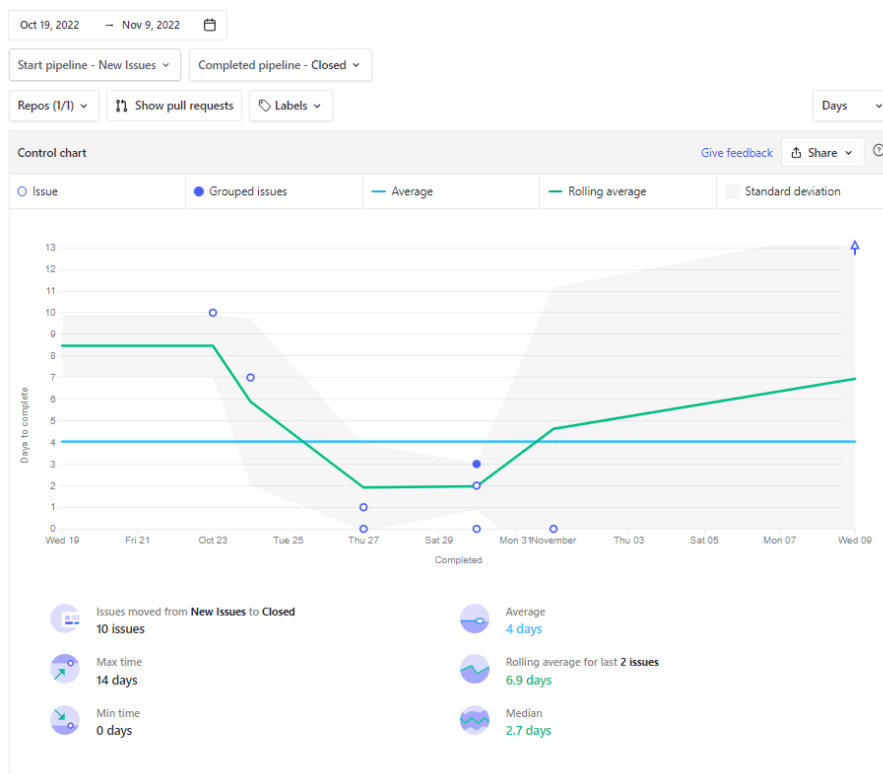


Figura A.1: Gráfica Control chart- Sprint 0

A.3. Planificación temporal

A.4. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

- B.1. Introducción
- B.2. Objetivos generales
- B.3. Catalogo de requisitos
- B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

- C.1. Introducción
- C.2. Diseño de datos
- C.3. Diseño procedimental
- C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

A continuación se detallará el proceso de instalación de los programas necesarios para el desarrollo de la aplicación.

Instalación de Java

Actualmente se sigue ejecutando con la versión de Java 11. A pesar de que se necesitará actualizar cuando migremos a la versión 23 de vaadin.

Para ello se debe descargar la [página de descargas de Oracle Java SE 11.0](#) y descargar la versión de JDK 11, correspondiente con el sistema operativo que se posea y su arquitectura, ya sea de 64 o 32 bits. Ver imagen [D.1](#).

Tras escoger la versión según el SO, se leerán y aceptarán las licencias de uso de Oracle [D.2](#), y se dará a descargar.












Java SE Development Kit 11.0.10		
This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE		
Product / File Description	File Size	Download
Linux ARM 64 Debian Package	145.64 MB	 jdk-11.0.10_linux-aarch64_bin.deb
Linux ARM 64 RPM Package	152.22 MB	 jdk-11.0.10_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	169.37 MB	 jdk-11.0.10_linux-aarch64_bin.tar.gz
Linux x64 Debian Package	149.39 MB	 jdk-11.0.10_linux-x64_bin.deb
Linux x64 RPM Package	156.12 MB	 jdk-11.0.10_linux-x64_bin.rpm
Linux x64 Compressed Archive	173.31 MB	 jdk-11.0.10_linux-x64_bin.tar.gz
macOS Installer	167.51 MB	 jdk-11.0.10_osx-x64_bin.dmg
macOS Compressed Archive	167.84 MB	 jdk-11.0.10_osx-x64_bin.tar.gz
Solaris SPARC Compressed Archive	184.82 MB	 jdk-11.0.10_solaris-sparcv9_bin.tar.gz
Windows x64 Installer	152.32 MB	 jdk-11.0.10_windows-x64_bin.exe
Windows x64 Compressed Archive	171.67 MB	 jdk-11.0.10_windows-x64_bin.zip

Figura D.1: Descarga de JDK 11

También se deberá cambiar la variable de entorno de Java del sistema.

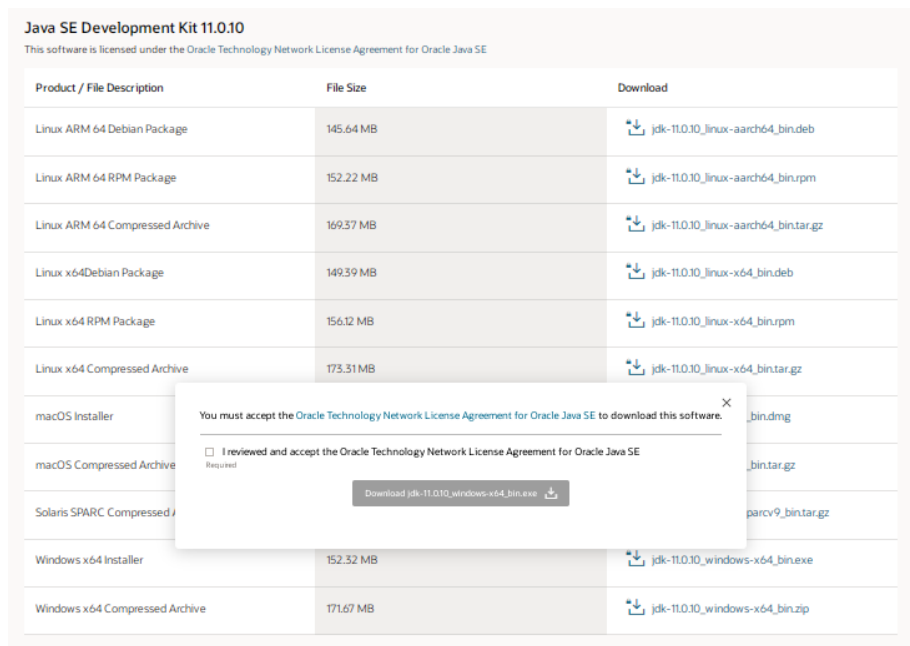


Figura D.2: Descarga JDK 11 Licencia

Instalación de Eclipse

A continuación se instalará un entorno de desarrollo integrado (IDE) para Java, en este caso se ha utilizado **Eclipse IDE for Enterprise Java Developers** en la versión 2021-12.

Para descargar el IDE se accederá a la [página de descargas de Eclipse](#) y descargar la opción correspondiente a nuestro sistema operativo del **Eclipse Installer 2021-12 R**. Ver imagen [D.3](#).

The Eclipse Installer 2021-12 R now includes a JRE for macOS, Windows and Linux.

Try the Eclipse **Installer** 2021-12 R

The easiest way to install and update your Eclipse Development Environment.

[Find out more](#)

📦 3,033,426 Installer Downloads

📦 2,923,118 Package Downloads and Updates


Download

macOS [x86_64](#) | [AArch64](#)

Windows [x86_64](#)

Linux [x86_64](#) | [AArch64](#)

Eclipse IDE 2021-12 R Packages




Eclipse IDE for Java Developers

306 MB | 1,581,141 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration

Windows [x86_64](#)
macOS [x86_64](#) | [AArch64](#)
Linux [x86_64](#) | [AArch64](#)



Eclipse IDE for Enterprise Java and Web Developers

509 MB | 972,819 DOWNLOADS

Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.

Windows [x86_64](#)
macOS [x86_64](#) | [AArch64](#)
Linux [x86_64](#) | [AArch64](#)

[Click here](#) to file a bug against Eclipse Web Tools Platform.
[Click here](#) to file a bug against Eclipse Platform.
[Click here](#) to file a bug against Maven integration for web projects.
[Click here](#) to report an issue against Eclipse Wild Web Developer (incubating).

Figura D.3: Descargar IDE Eclipse

En el caso de los sistemas operativos Windows se descargará un archivo ejecutable que se deberá ejecutar como administrador. Una vez ejecutado se deberá seleccionar la opción “***Eclipse IDE for Enterprise Java Developers***” [D.4](#).



Figura D.4: Seleccionar Eclipse

Por último seleccionaremos el JDK (11) que vayamos a utilizar y la carpeta donde queremos instalar nuestro IDE.

Instalación del *plugin de Vaadin* para Eclipse

Una vez se haya instalado Eclipse, se procederá a añadir el plugin de Vaadin para Eclipse. Esto se realizará mediante el **Eclipse Marketplace de Eclipse D.5**, el cual se encuentra en la opción de “**Help/Eclipse Marketplace...**” de la barra de herramientas.

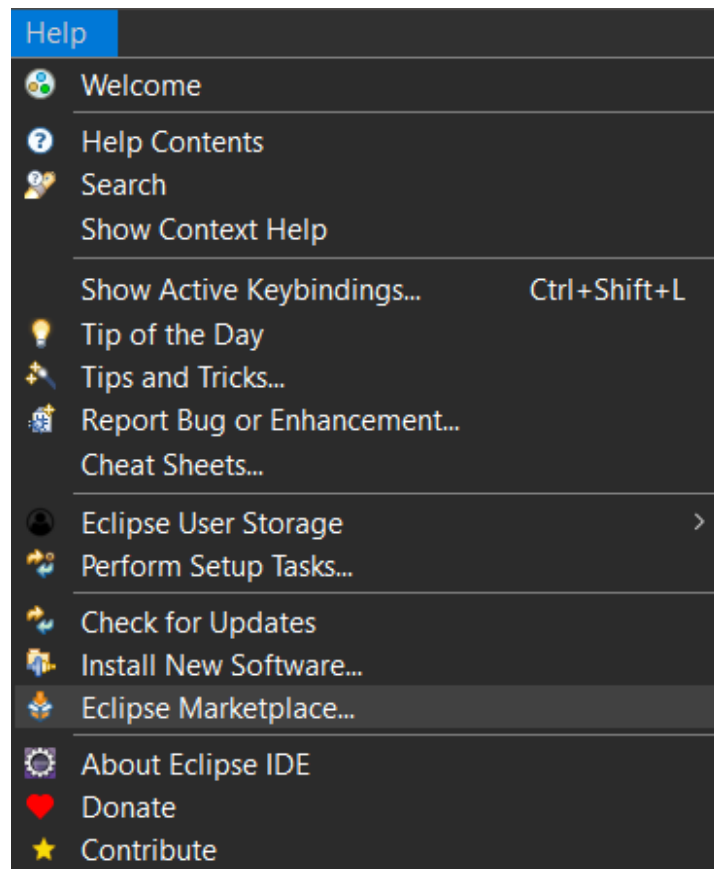


Figura D.5: Eclipse marketplace

Una vez en el Eclipse Marketplace, se buscará “**Vaadin**” y se pulsará “**Go**”. Tras salir el plugin “***Vaadin Plugin for Eclipse***”, se dará a “**Install**” y comenzará la instalación del plugin [D.6](#). En la imagen ya se muestra una vez instalado.

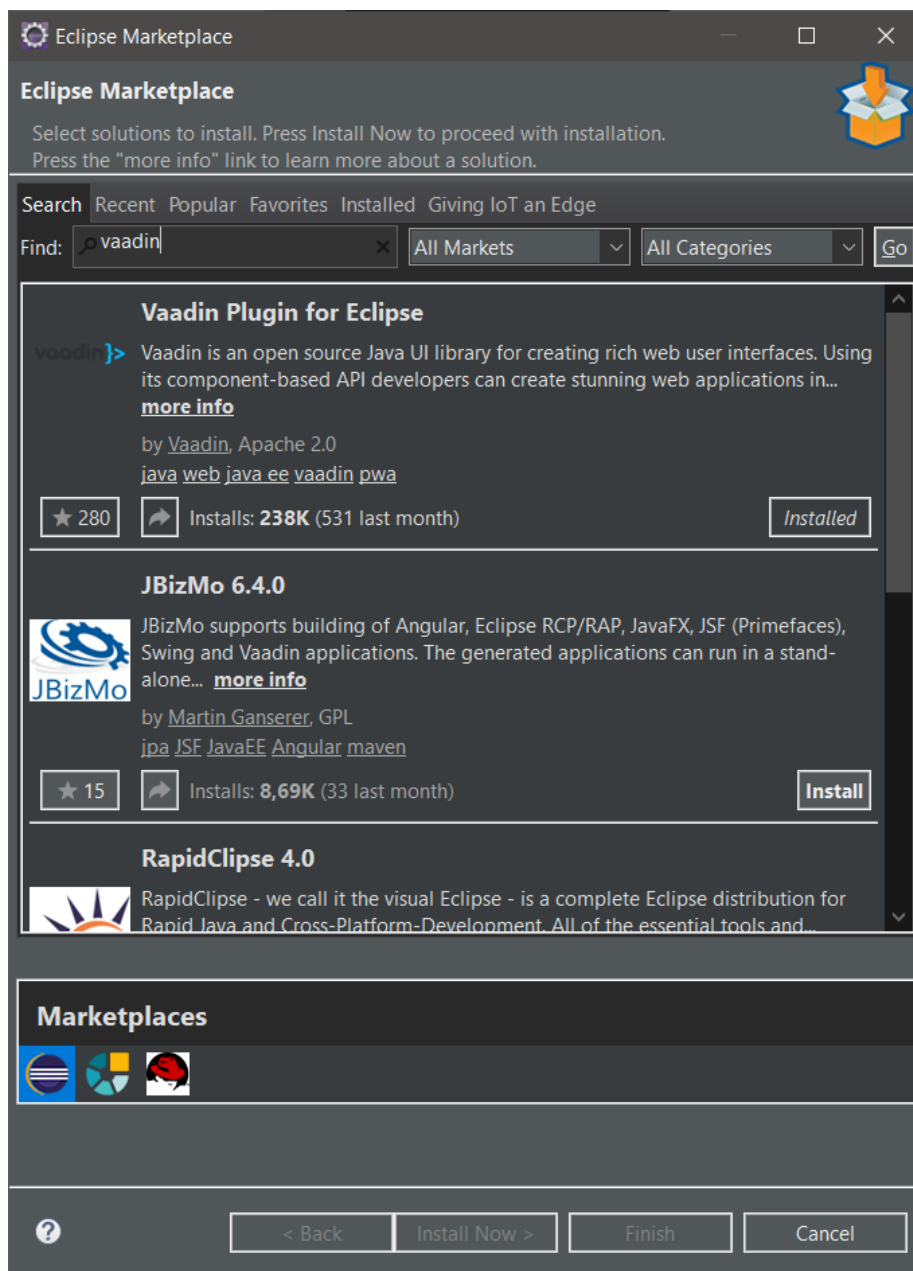


Figura D.6: Plugin Vaadin

D.4. Compilación, instalación y ejecución del proyecto

Se explicará como compilar, instalar y ejecutar el proyecto. En el caso de la ejecución, se detallará como hacerlo desde un terminal y mediante Eclipse (IDE).

Descarga del repositorio

El código fuente se encuentra en el **repositorio del proyecto** en GitHub. Para descargarlo se deberá hacer click en “**Code**” y copiar la URL que aparece en el apartado de “**HTTP**”. Con esta URL deberemos ir al “**GitHub Desktop**” y clonar el repositorio **D.7**.

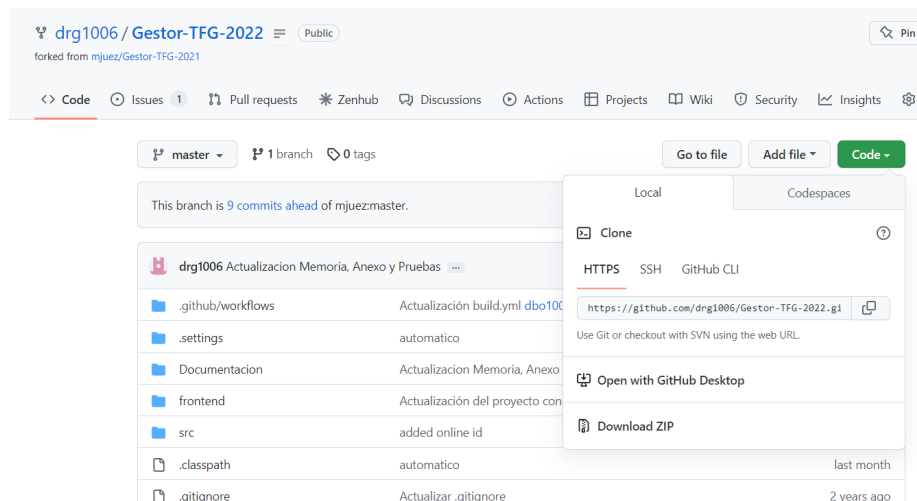


Figura D.7: Copiar URL repositorio

Si se desea tener código en local se deberá descargar el zip “**Download ZIP**” en la opción “**Code**” anteriormente mencionada. Una vez descargado el zip se descomprimirá y abrirá con Eclipse.

Compilación del proyecto

Para compilar el proyecto en local desde terminal se usará:

- Limpiar las dependencias: “**mvn clean**”.
- Instalar dependencias y compilar: “**mvn install**”.

- Instalar en modo producción (para desplegar): “**mvn package -Pproduction**”.
- Ejecutar test: “**mvn test**”.

Ejecución del proyecto desde local

Para la ejecución del proyecto en local desde terminal se usará:

- Entrar en la terminal que utilizemos.
- Acceder a la carpeta donde tenemos nuestro servidor tomcat instalado y entrar en la carpeta **/bin**.
- Ejecutar nuestro servidor local mediante **startup** [D.8](#).
- Entrar en el nuestro navegador en la direccion **localhost:8080**.
- Pulsar en la opción Manage App [D.9](#).
- Iniciamos sesión como manager-gui. (Indicado en el archivo `/conf/tomcat-users.xml`).
- Llegaremos a la pantalla [D.10](#) y seleccionaremos el archivo `.war` que hemos creado al compilar nuestro proyecto con “**mvn package -Pproduction**”.

```
C:\Programas\apache-tomcat-9.0.68\bin>startup
Using CATALINA_BASE:   "C:\Programas\apache-tomcat-9.0.68"
Using CATALINA_HOME:   "C:\Programas\apache-tomcat-9.0.68"
Using CATALINA_TMPDIR: "C:\Programas\apache-tomcat-9.0.68\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk-11.0.16"
Using CLASSPATH:       "C:\Programas\apache-tomcat-9.0.68\bin\bootstrap.jar;C:\Programas\apache-tomcat-9.0.68\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""
```

Figura D.8: Consola con Tomcat ejecutado

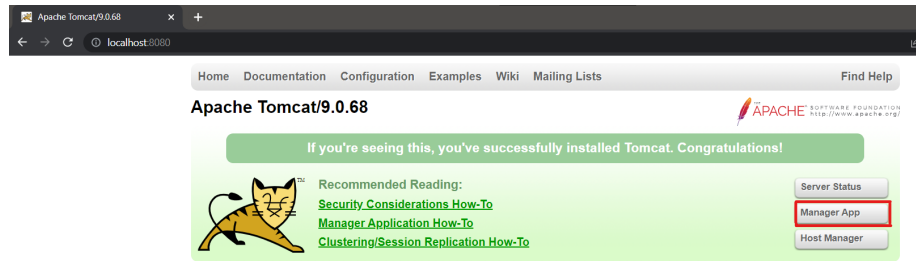


Figura D.9: Gestor de Aplicaciones de Tomcat

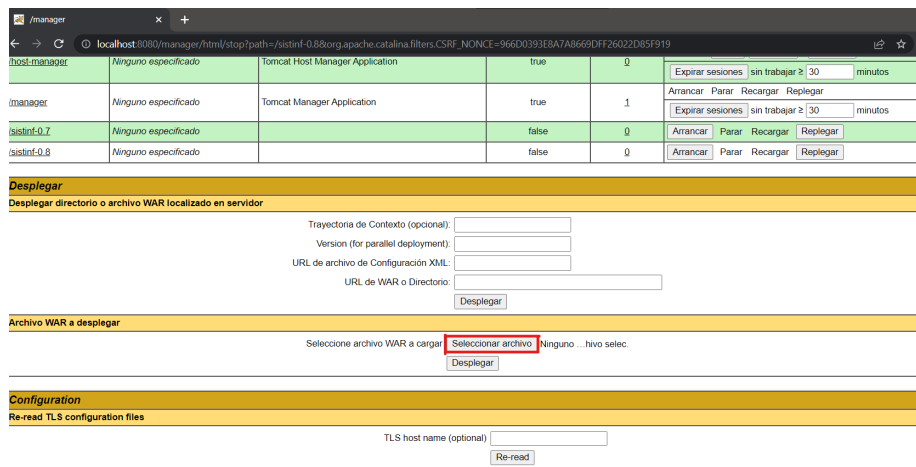


Figura D.10: Desplegar el archivo .war

Ejecución del proyecto desde Eclipse IDE

Para la ejecución del proyecto en local desde Eclipse primero debemos importar como proyecto Maven, con el pom.xml. Utilizaremos también un servidor local de de **Apache Tomcat**, en concreto, la versión 9. Se puede descargar en **la página oficial de Apache Tomcat**.

Una vez descargado y descomprimido, se creará un servicio de Tomcat **D.11** con la ruta donde se tiene descargado Tomcat y se le dará un nombre **D.12**. Por último, se añadirá el proyecto principal “sistinf” **D.13**.

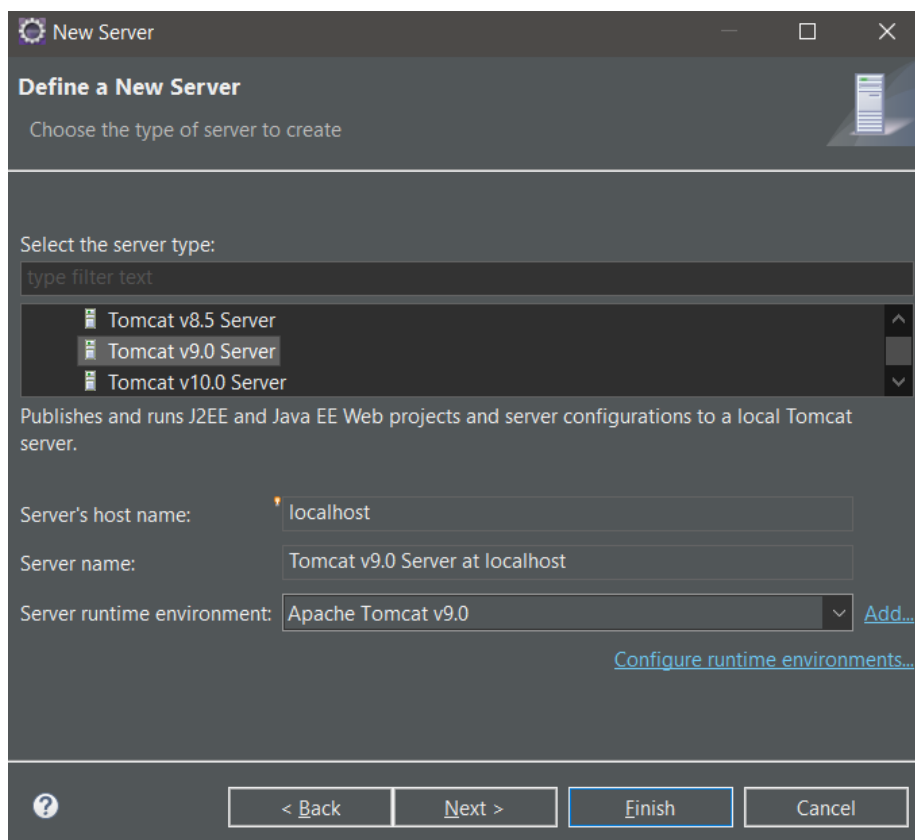


Figura D.11: Añadir servidor de Tomcat a Eclipse

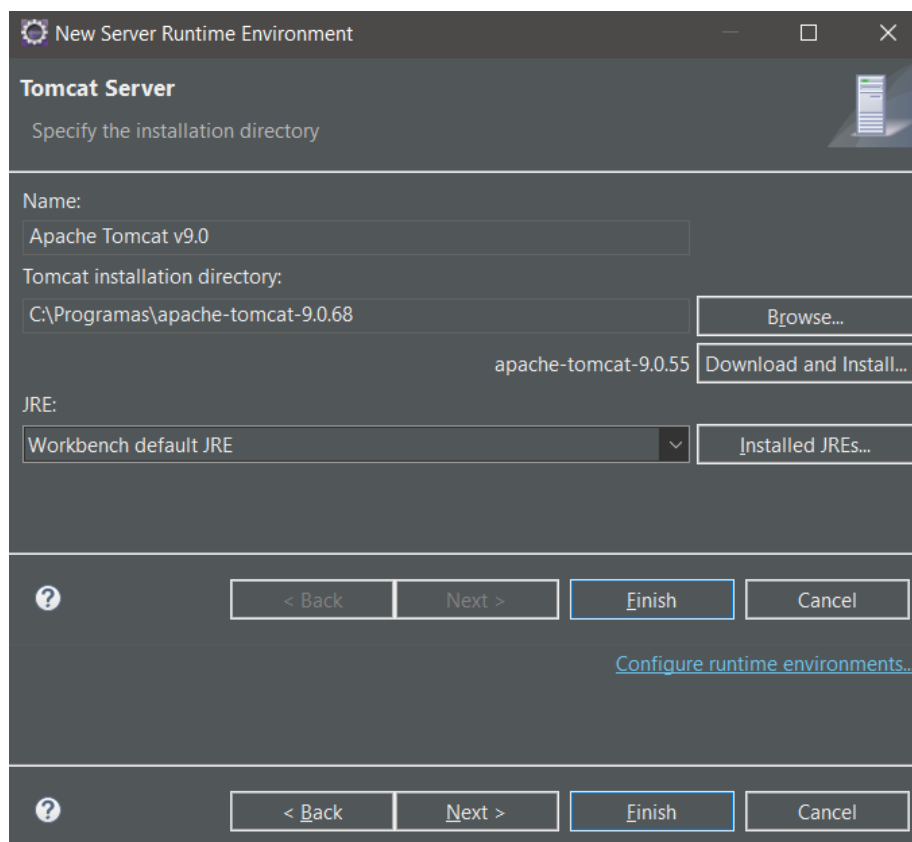


Figura D.12: Seleccionar carpeta contenedora de Tomcat

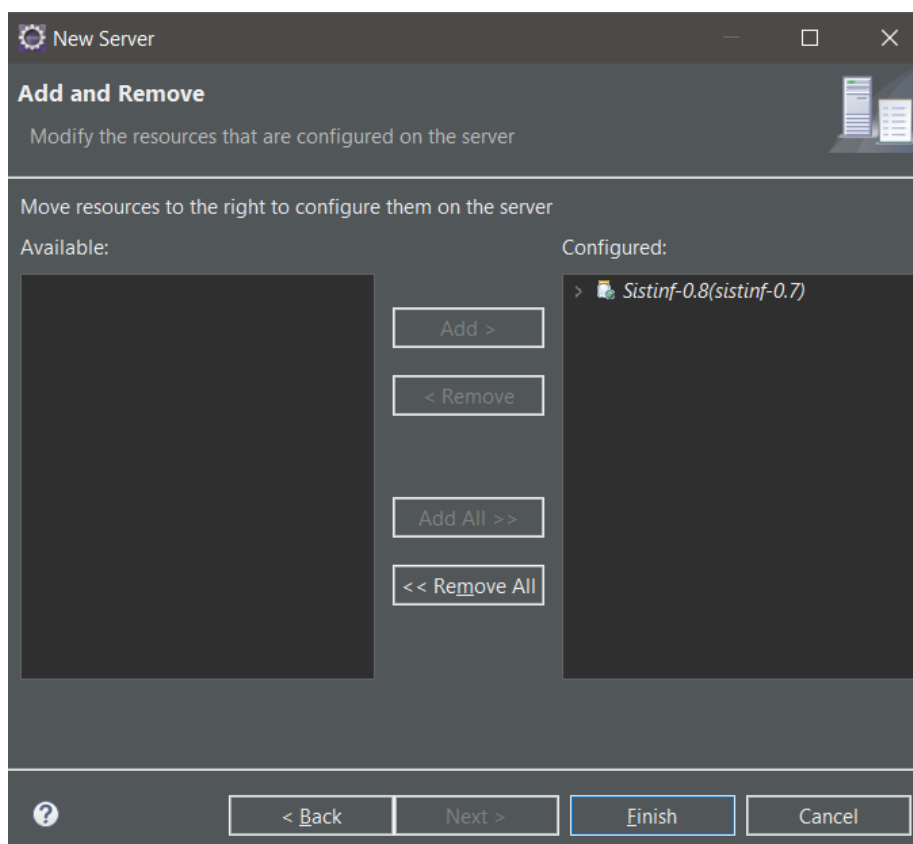


Figura D.13: Añadir proyectos a servidor

Para ejecutarlo desde eclipse debemos también seguir todos los pasos de compilación anteriormente mencionados.

Una vez tengamos compilado nuestro código debemos ejecutarlo (click derecho en el proyecto>“Run As”>“Run on Server”).

Si no aparece la vista de los servicios se puede añadir desde la barra de herramientas>“Window”>“Show View”. Para configurar la ruta donde se ejecuta la aplicación, por defecto en `localhost:8080/` o en ciertos casos `localhost:8080/sistinf`.

Problemas a la hora de ejecutar el proyecto

A la hora de ejecutar el proyecto anterior surgieron una serie de problemas tanto para la ejecución por terminal como desde Eclipse.

Cuando quise ejecutarlo mediante la terminal desplegando el archivo .war generado tras compilar me surgía el siguiente error [D.14](#)

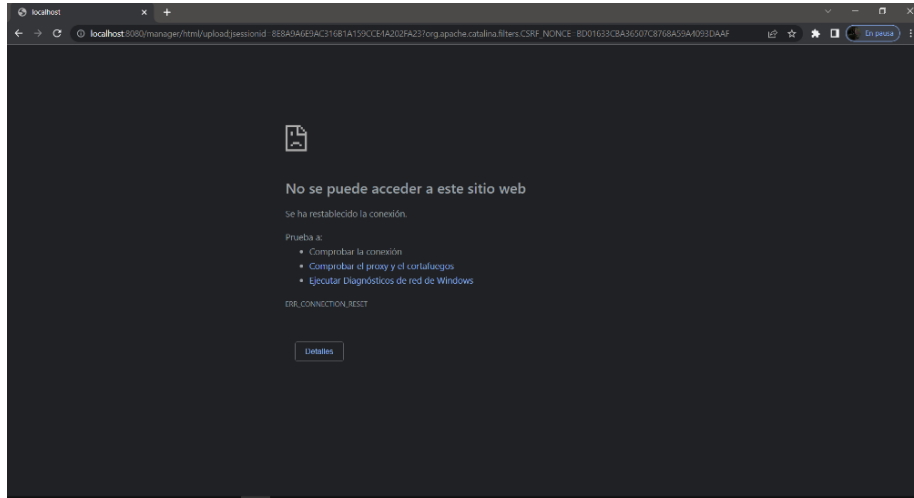


Figura D.14: Error tras desplegar el .war en el Gestor de Aplicaciones de Tomcat

Tras buscar información sobre el posible error, se descubre en los logs que proporciona tomcat lo siguiente D.15. En el que se informa que se intenta ejecutar un proyecto con un tamaño mayor al que tenemos configurado en tomcat.

```
08-Nov-2022 19:57:41.644 INFO [http-nio-8080-exec-2] org.apache.catalina.core.ApplicationContext.log HTMLManager: info: Associated with Deployer 'CatalinaType-Deployer,host=localhost'
08-Nov-2022 19:57:41.644 INFO [http-nio-8080-exec-2] org.apache.catalina.core.ApplicationContext.log HTMLManager: info: Global resources are available
08-Nov-2022 19:57:41.651 INFO [http-nio-8080-exec-2] org.apache.catalina.core.ApplicationContext.log HTMLManager: List: Listing contents for virtual host 'localhost'
08-Nov-2022 19:58:02.486 SEVERE [http-nio-8080-exec-4] org.apache.catalina.core.ApplicationContext.log HTMLManager: Failed - Failed Carga de Despliegue, excepción: [org.apache.tomcat.util.http.fileupload.impl.SizeLimitExceededException: the request was rejected because its size (72771942) exceeds the configured maximum (52428800)]
java.lang.IllegalStateException: org.apache.tomcat.util.http.fileupload.impl.SizeLimitExceededException: the request was rejected because its size (72771942) exceeds the configured maximum (52428800)
    at org.apache.catalina.connector.Request.parseParts(Request.java:2074)
    at org.apache.catalina.connector.Request.getParameter(Request.java:1262)
    at org.apache.catalina.connector.Request.getParameter(Request.java:1141)
    at org.apache.catalina.connector.RequestFacade.getParameter(RequestFacade.java:181)
    at org.apache.catalina.filters.CorsPreventionFilter.doFilter(CorsPreventionFilter.java:127)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:189)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
    at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:189)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
    at org.apache.catalina.filters.HttpHeaderSecurityFilter.doFilter(HttpHeaderSecurityFilter.java:126)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:189)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
    at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:159)
    at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:97)
    at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:688)
    at org.apache.catalina.valves.RequestFilterValve.process(RequestFilterValve.java:378)
    at org.apache.catalina.valves.RemoteAddrValve.invoke(RemoteAddrValve.java:36)
    at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:135)
    at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92)
    at org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:687)
    at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:78)
    at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:360)
    at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:180)
    at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:85)
    at org.apache.coyote.AbstractProtocolConnectionHandler.process(AbstractProtocol.java:893)
    at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1780)
    at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
    at org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191)
    at org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659)
    at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
    at java.base/java.lang.Thread.run(Thread.java:834)
Caused by: org.apache.tomcat.util.http.fileupload.impl.SizeLimitExceededException: the request was rejected because its size (72771942) exceeds the configured maximum (52428800)
    at org.apache.tomcat.util.http.fileupload.impl.FileItemIteratorImpl.init(FileItemIteratorImpl.java:161)
    at org.apache.tomcat.util.http.fileupload.impl.FileItemIteratorImpl.getInputStream(FileItemIteratorImpl.java:205)
    at org.apache.tomcat.util.http.fileupload.impl.FileItemIteratorImpl.findNextItem(FileItemIteratorImpl.java:124)
    at org.apache.tomcat.util.http.fileupload.impl.FileItemIteratorImpl.<init>(FileItemIteratorImpl.java:142)
    at org.apache.tomcat.util.http.fileupload.FileUploadBase.getItemIterator(FileUploadBase.java:253)
    at org.apache.tomcat.util.http.fileupload.FileUploadBase.parseRequest(FileUploadBase.java:276)
    at org.apache.catalina.connector.Request.parseParts(Request.java:2032)
    ... 31 more
08-Nov-2022 19:58:02.486 INFO [http-nio-8080-exec-4] org.apache.catalina.core.ApplicationContext.log HTMLManager: List: Listing contents for virtual host 'localhost'
```

Figura D.15: Logs proporcionados por Tomcat

Para solucionar este problema se accede al archivo *apache-tomcat-9.0.68-webapps-manager-WEB-INF* y se modifican las siguientes líneas D.16 aumentando el número que se indica.



```
web.xml: Bloc de notas
Archivo Edición Formato Ver Ayuda
<request-character-encoding>UTF-8</request-character-encoding>

<servlet>
  <servlet-name>Managers</servlet-name>
  <servlet-class>org.apache.catalina.manager.ManagerServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>2</param-value>
  </init-param>
</servlet>
<servlet>
  <servlet-name>HTMLManager</servlet-name>
  <servlet-class>org.apache.catalina.manager.HTMLManagerServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>2</param-value>
  </init-param>
  <!-- Uncomment this to show proxy sessions from the Backup manager or a
  StoreManager in the sessions list for an application
  <init-param>
    <param-name>showProxySessions</param-name>
    <param-value>true</param-value>
  </init-param>
  -->
  <multipart-config>
    <!-- 50MB max -->
    <max-file-size>524288000</max-file-size>
    <max-request-size>524288000</max-request-size>
    <file-size-threshold>8</file-size-threshold>
  </multipart-config>
</servlet>
<servlet>
  <servlet-name>Status</servlet-name>
  <servlet-class>org.apache.catalina.manager.StatusManagerServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>8</param-value>
  </init-param>
</servlet>
<servlet>
  <servlet-name>JSPProxy</servlet-name>
```

Figura D.16: Logs proporcionados por Tomcat

Cuando quise ejecutarlo mediante Eclipse no me dejaba añadir el proyecto al servidor de *tomcat*, indicando que las versiones no eran compatibles. Por ello se ha entrado en las propiedades del proyecto y se ha cambiado la versión del parametro *Dynamic Web Module* a la 3.1 en el apartado *Project Facets* como se aprecia en D.17 .

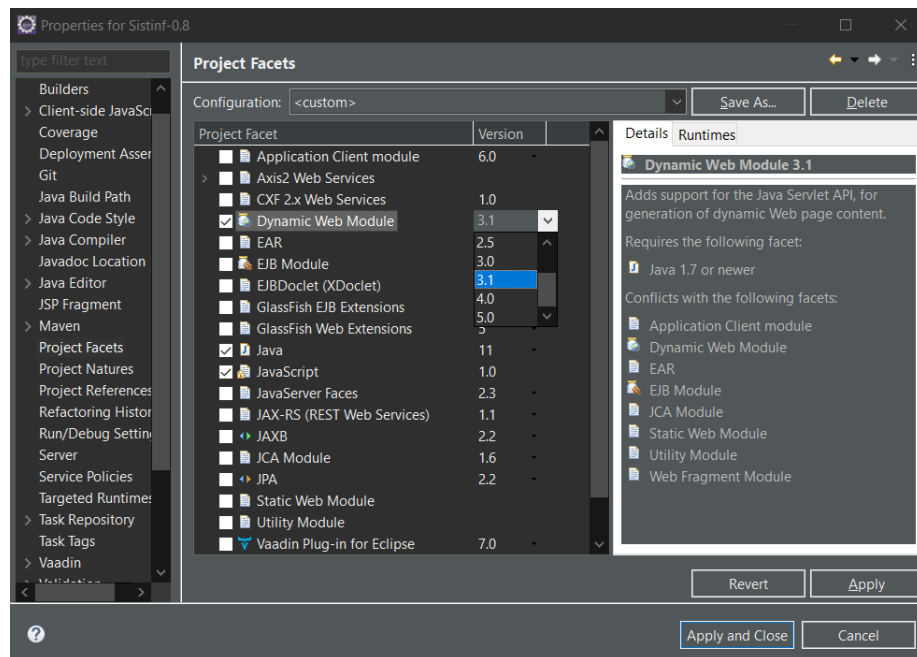


Figura D.17: Cambio de versión Dynamic Web Module

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

- E.1. Introducción
- E.2. Requisitos de usuarios
- E.3. Instalación
- E.4. Manual del usuario

Bibliografía
