Contenido

[**1.** **Introducción:** 5](#_Toc481177031)

[**2.** **Obtención del código:** 6](#_Toc481177032)

[**3.** **Instalación del repositorio:** 8](#_Toc481177033)

[**3.1.** **Generaciones ejecutables:** 8](#_Toc481177034)

[**3.1.1.** **Eclipse: Conversión Maven** 8](#_Toc481177035)

[**3.1.2.** **Eclipse: Incluir e Ejecutar** 11](#_Toc481177036)

[**3.2.** **Instalación de base de datos:** 16](#_Toc481177037)

[**3.2.1.** **Generación de estado inicial:** 17](#_Toc481177038)

[**3.3.** **Ejecución del repositorio:** 19](#_Toc481177039)

[**4.** **Instalación del servicio web:** 20](#_Toc481177040)

[**4.1.** **Instalación de Node.js:** 20](#_Toc481177041)

[**4.2.** **Arranque servidor web:** 21](#_Toc481177042)

[**5.** **Arranque rápido en Ubuntu server:** 22](#_Toc481177043)

[**5.1.** **Instrucciones de arranque:** 22](#_Toc481177044)

[**5.2.** **Listado de programas:** 22](#_Toc481177045)

[**6.** **Manual de creación de proyectos:** 23](#_Toc481177046)

[**7.** **Manual de utilización del repositorio:** 23](#_Toc481177047)

[**7.1.** **Iteración habitual:** 23](#_Toc481177048)

[**7.2.** **Protocolo de comunicación** 24](#_Toc481177049)

[**7.2.1.** **Autenticación:** 24](#_Toc481177050)

[**7.2.2.** **Grupos:** 25](#_Toc481177051)

[**7.2.2.1.** **Grupos acciones básicas:** 25](#_Toc481177052)

[**7.2.2.2.** **Grupos miembros:** 26](#_Toc481177053)

[**7.2.2.3.** **Grupos subgrupos:** 26](#_Toc481177054)

[**7.2.2.4.** **Grupos proyectos:** 27](#_Toc481177055)

[**7.2.3.** **Proyectos:** 28](#_Toc481177056)

[**7.2.4.** **Ejecuciones:** 28](#_Toc481177057)

[**7.2.5.** **Administración:** 29](#_Toc481177058)

[**7.2.6.** **Navegación:** 30](#_Toc481177059)

[**7.3.** **UniApi (JS):** 31](#_Toc481177060)

[**8.** **Manual de utilización de la aplicación web:** 32](#_Toc481177061)

[**8.1.1.** **Autenticarse y crear una cuenta:** 32](#_Toc481177062)

[**8.1.2.** **Navegar por los grupos:** 34](#_Toc481177063)

[**8.1.3.** **Crear un grupo:** 35](#_Toc481177064)

[**8.1.4.** **Crear un proyecto:** 37](#_Toc481177065)

[**8.1.5.** **Crear una ejecución:** 39](#_Toc481177066)

[**8.1.6.** **Gestionar un proyecto propio:** 40](#_Toc481177067)

[**8.1.7.** **Gestionar un grupo propio:** 42](#_Toc481177068)

[**8.1.8.** **Información básica:** 43](#_Toc481177069)

[**8.1.9.** **Miembros:** 44](#_Toc481177070)

[**8.1.10.** **Subgrupos:** 44](#_Toc481177071)

[**8.1.11.** **Permisos:** 44](#_Toc481177072)

[**8.1.12.** **Gestionar una ejecución propia:** 44](#_Toc481177073)

[Ilustración 1. Página del repositorio 5](#_Toc480479757)

[Ilustración 2. Botón de descarga GIT 6](#_Toc480479758)

[Ilustración 3. Seleccion import eclipse 8](#_Toc480479759)

[Ilustración 4. Multiples opciones de import 8](#_Toc480479760)

[Ilustración 5. Import projects final 9](#_Toc480479761)

[Ilustración 6. Convert project maven 10](#_Toc480479762)

[Ilustración 7. Maven run as... 11](#_Toc480479763)

[Ilustración 8. Opciones maven 11](#_Toc480479764)

[Ilustración 9. Carpeta target 12](#_Toc480479765)

[Ilustración 10. Estado final externalJar del proyecto REST 13](#_Toc480479766)

[Ilustración 11. Comprobación JDK 14](#_Toc480479767)

[Ilustración 12. Comprobación JavaSE 14](#_Toc480479768)

[Ilustración 13. Ventana neo4j 15](#_Toc480479769)

[Ilustración 14. Ventana neo4j Iniciada 16](#_Toc480479770)

[Ilustración 15. Jerarquía de uniapi (prompt) 20](#_Toc480479771)

[Ilustración 16. Ejecución del servidor web 20](#_Toc480479772)

[Ilustración 17 Ventana Loggin 31](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479773)

[Ilustración 18 Pagina Principal 32](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479774)

[Ilustración 19 Botón sign up 32](#_Toc480479775)

[Ilustración 20 Ventana nuevo usuario 33](#_Toc480479776)

[Ilustración 21 Menú Navegación 33](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479777)

[Ilustración 22 Nuevo grupo 35](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479778)

[Ilustración 23 Nuevo grupo: Tipo grupo 35](#_Toc480479779)

[Ilustración 24 Nuevo grupo: Permisos 36](#_Toc480479780)

[Ilustración 25 Nuevo proyecto: Integración o creación 36](#_Toc480479781)

[Ilustración 26 Nuevo proyecto: principal 37](#_Toc480479782)

[Ilustración 27 Nuevo grupo: tipo de proyecto 38](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479783)

[Ilustración 28 Nuevo grupo: Entradas por defecto 38](#_Toc480479784)

[Ilustración 29 Nueva ejecución 39](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479785)

[Ilustración 30 Proyectos 40](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479786)

[Ilustración 31 Gestión proyecto 40](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479787)

[Ilustración 32 Mis Grupos 41](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479788)

[Ilustración 33 Gestión grupo 42](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479789)

[Ilustración 34 Gestión de ejecución: En ejecución 44](file:///C:\Users\rulo-\Desktop\UniApi\documentación\individual\ManualDelUsuario.docx#_Toc480479790)

[Ilustración 35 Gestión de ejecución: Finalizado 45](#_Toc480479791)

[Tabla 1. Protocolo repositorio: Autenticación 24](#_Toc480479792)

[Tabla 2.Protocolo repositorio: Grupo acciones básicas 25](#_Toc480479793)

[Tabla 3.Protocolo repositorio: Grupo miembros 25](#_Toc480479794)

[Tabla 4.Protocolo repositorio: Grupo subgrupos 26](#_Toc480479795)

[Tabla 5.Protocolo repositorio: Grupo proyectos 27](#_Toc480479796)

[Tabla 6 .Protocolo repositorio: Proyectos 28](#_Toc480479797)

[Tabla 7.Protocolo repositorio: Ejecuciones 28](#_Toc480479798)

[Tabla 8.Protocolo repositorio: Administración 29](#_Toc480479799)

[Tabla 9.Protocolo repositorio: Navegación 30](#_Toc480479800)

[Tabla 10 UniApi.js: Nombre de funciones 31](#_Toc480479801)

1. **Introducción:**

Este documento denominado “Manual del usuario”, es un documento de un conjunto de documentos cuya finalidad es el definir y diseñar una “Aplicación web que gestione un repositorio”.

Para poder describir mejor el problema deberemos desarrollar el significado de lo que se va a definir, es decir, “Aplicación web que gestione un repositorio”. Un repositorio es según la Wikipedia: Un repositorio de software es un lugar de almacenamiento del cual pueden ser recuperados e instalados los paquetes de software en un ordenador.

Nuestro repositorio mantendrá un lugar de almacenamiento dentro del ordenador que lo instale, con el fin de contener diversos proyectos que albergarán programas de diversos lenguajes que podrán ser ejecutados. Nuestro repositorio devolverá no es programa almacenado, sino, la información de interactuar con él.

Nuestra aplicación web, gestionara la comunicación con nuestro repositorio. Pidiéndole información y mostrándosela al usuario de una manera sencilla para su entendimiento. Se insta que si se quiere saber más sobre el objetivo de la aplicación se lea el documento general de la memoria.

El objetivo de este documento es el realizar un manual del usuario, para que todos los lectores e revisores de este articulo puedan utilizar el software a desarrollar. Para ello se listará un conjunto de instrucciones y descripciones del sistema que creemos que serán de utilidad para poder utilizar el sistema.

1. **Obtención del código:**

Para poder obtener los códigos de los diversos programas usuraremos la plataforma de repositorios GIT conocida como **Github**. Github provee un repositorio permanente para nuestro programa con un seguidor de modificaciones muy notorio que facilitara a cualquier persona el descargarse versiones modificadas o mejoradas.

<https://github.com/RaulGF92/UniApi>

La dirección previamente mostrada se referencia al espacio donde el repositorio se alberga, ahí se podrá ver todas las modificaciones, paso por paso, que el desarrollador a usado. Cualquier persona con conocimientos básicos sobre GIT puede descargárselo. No obstante, mostraremos las dos formas de realizarlo:

* Descargando a través de la página web: En este caso el usuario deberá introducirse en la página indicada por el documento. Se le mostrara una página web como la indicada en la figura de a continuación. Ahí deberá pulsar en el botón “*Clone or download*”.

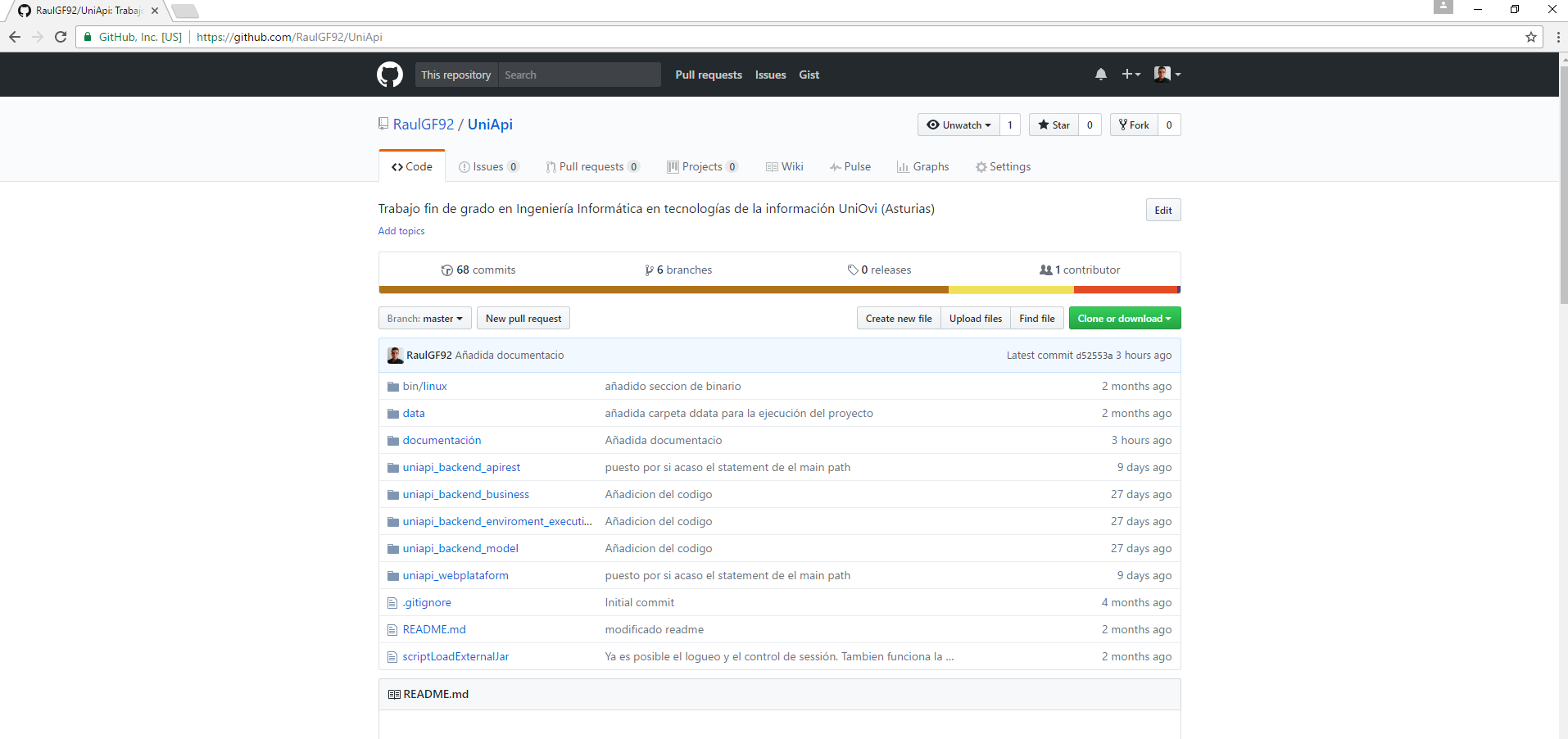


Ilustración . Página del repositorio

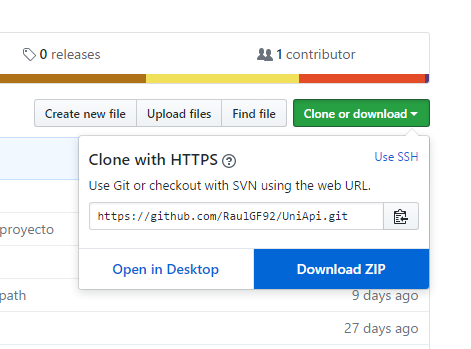


Ilustración . Botón de descarga GIT

Tras pulsar el botón de download Zip. Se descargará un archivo Zip donde se contendrá el proyecto UniApi. Es necesario descomprimirlo y depositarlo en el sistema de archivos del servidor que albergara el servicio.

* Mediante consola de comandos GIT: Para ello debemos conocer la dinámica del protocolo GIT y sus comportamientos. Para ello realizaremos un clonado del repositorio usando la URL que se nos indica posteriormente. Tras eso el protocolo se ocupará automáticamente de la descarga.

1. **Instalación del repositorio:**

Para instalar el repositorio necesitaremos los datos del repositorio además de un IDE denominado Eclipse. Usaremos este IDE por que gestiona Maven un gestor de paquetes Java de forma transparente para el desarrollador. Aunque es posible el uso de maven usando el prompt del sistema operativo.

* 1. **Generaciones ejecutables:**

En esta parte realizaremos un ejecutable que se utilizara para ejecutar el repositorio a partir de todos los datos de la jerarquía uniapi, descargada o clonada por el usuario en el repositorio indicado a inicio de este documento.

* + 1. **Eclipse: Conversión Maven**

Para empezar con la conversión inicializamos el eclipse. Nos preguntara donde albergar el workspace. Es recomendable indicarle que debe de ser la jerarquía uniapi donde se alberga los datos. El eclipse se empezará a cargar y nos proporcionará un espacio de trabajo vacío.

Entramos en las opciones File y elegimos la opción importar. Ahí nos desplega un sinfín de opciones elegimos la opción “**Existing projects in the workspace**”.

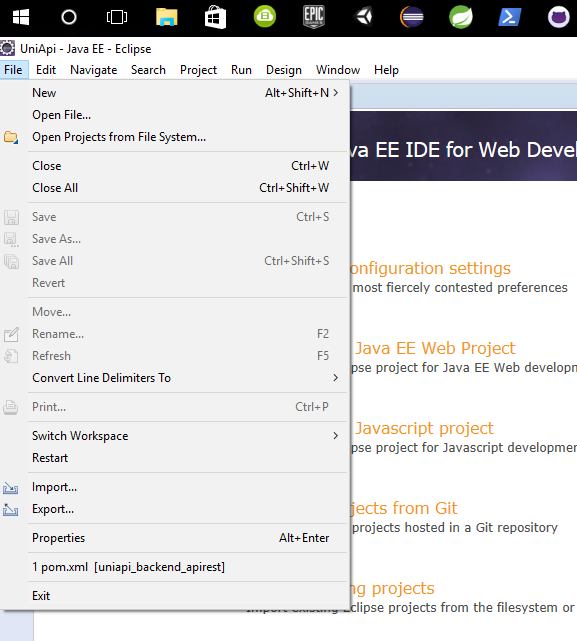


Ilustración . Seleccion import eclipse

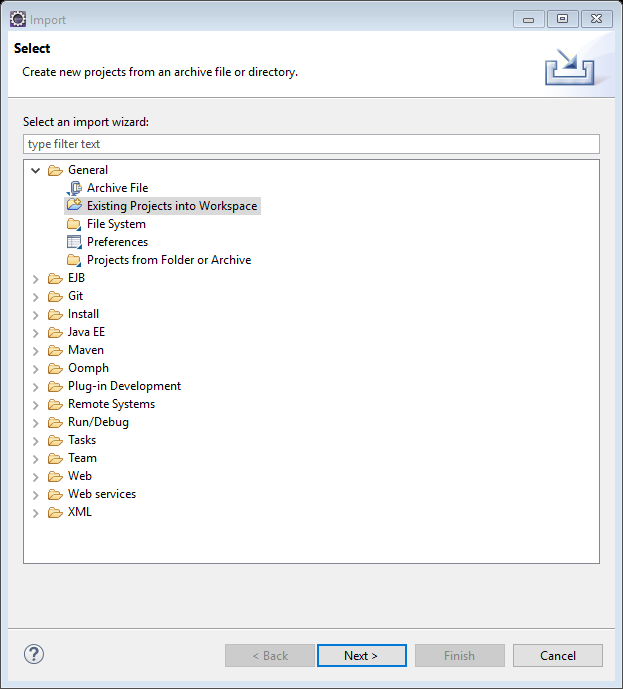


Ilustración . Multiples opciones de import

Tras seleccionar la opción se nos mostrara otra ventana donde deberemos seleccionar el check box “**root directive**”. Siguiendo una ventana que nos ayuda a navegar por la jerarquía seleccionamos la carpeta uniapi. Se nos indicara los tres proyectos de uniapi, seleccionaremos los tres y por ultimo daremos a finish.

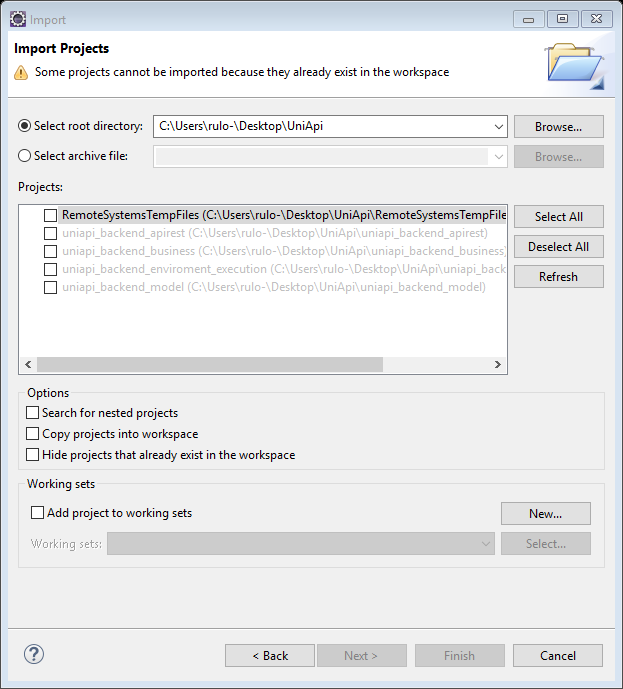


Ilustración . Import projects final

Tras la importación tendremos cuatro proyectos java. Para terminar la conversión deberemos ir uno por uno pulsando el botón derecho. Llendo a la pestaña configure y por ultimo pulsando convert in maven Project.

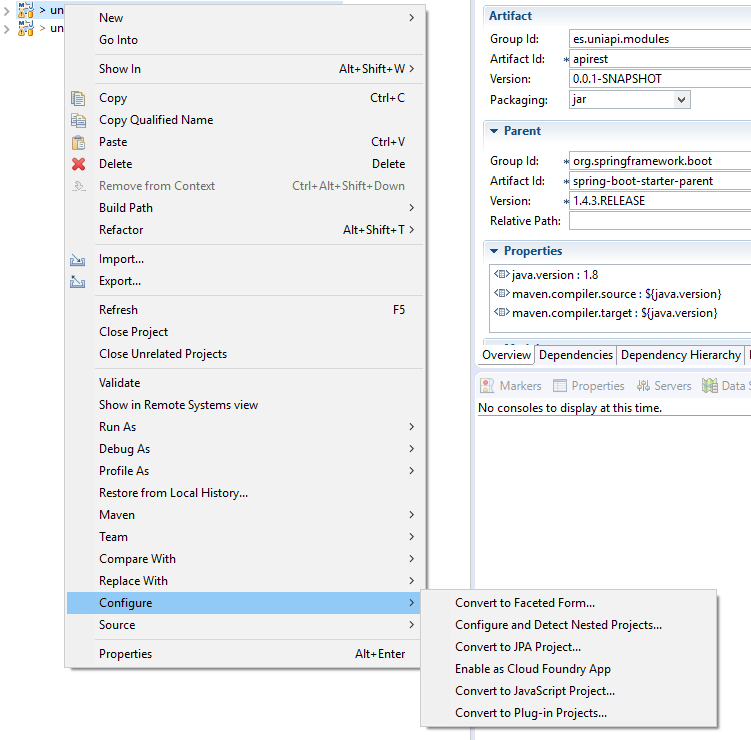


Ilustración . Convert project maven

Con esto todos los proyectos de nuestro workspace serán maven. Por ultimo selecciona cada proyecto de nuevo. Pulsa botón derecho, clicka la opción maven y clicka update. Esto actualizara los proyectos. Esta acción generara muchos errores, pero luego serán arreglados.

* + 1. **Eclipse: Incluir e Ejecutar**

Tras contener todos los proyectos maven instalados e actualizados en el workspace vamos a compilarlos y ejecutarlos. La aplicación del repositorio es una aplicación en pila, es decir, las capas superiores utilizan datos de la capa inferior. Es necesario compilar los datos para que las capas puedan utilizarlos. Para realizar esta operación iremos de la capa más inferior a la más superior construyendo sus proyectos, compilándolos y entregándolos a las capas superiores.

1. **Modelo de datos**: nos colocamos en el proyecto “**uniapi\_backend\_model**” y lo seleccionamos, pulsamos el botón derecho y nos dirigimos a run, donde pincharemos maven build…

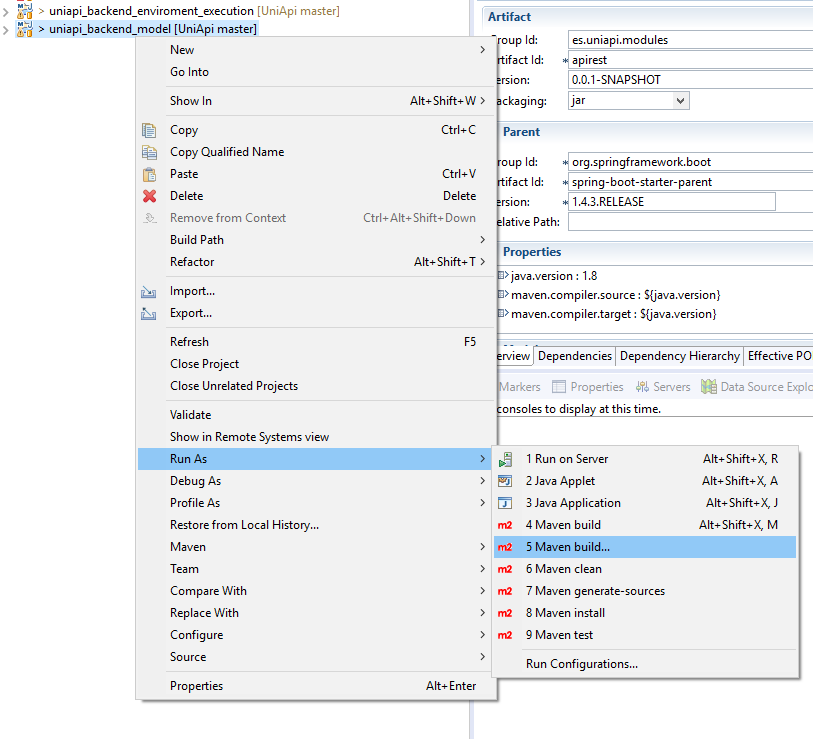
****

Ilustración . Maven run as...

Tras pinchar en la opción nos aparecerá una ventana de opción de maven en la pestaña “Goals” elegimos el siguiente código:

clean build package

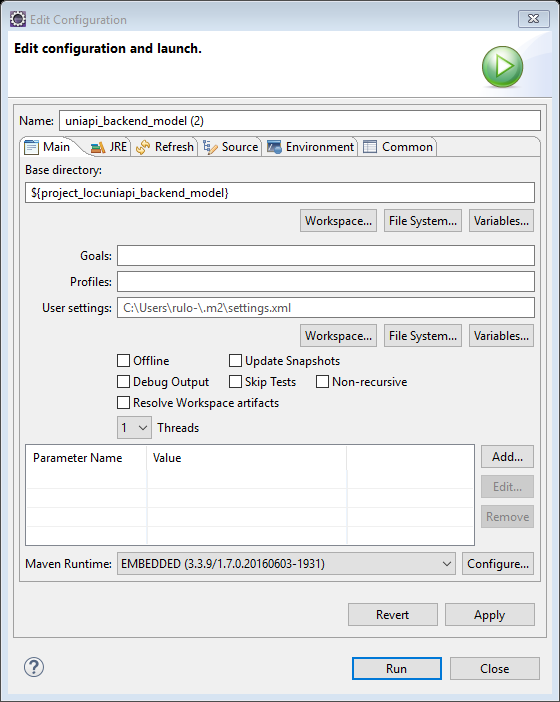


Ilustración . Opciones maven

Tras realizar esto tendremos un ejecutable dentro de la carpeta target del proyecto justo tal que en la figura siguiente:

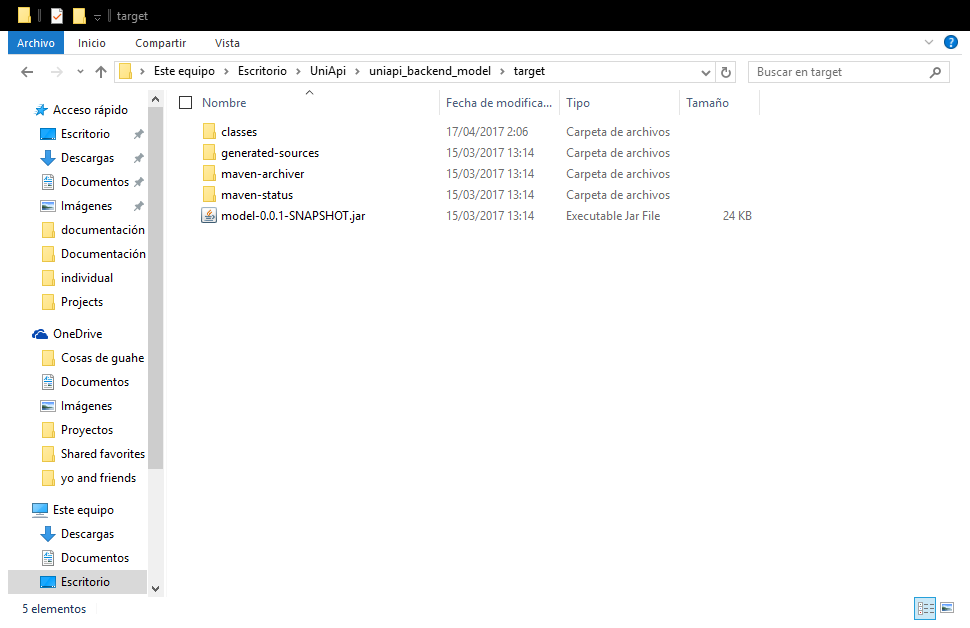


Ilustración 9. Carpeta target

Deberemos crear en la carpeta de todos los proyectos, menos este mismo, una capeta denominada “externalJar”. La carpeta debe ser denomina igual que lo descrito ya que si no, el proceso no funcionara. Tras la creación de la carpeta introducimos el “**model-0.0.1-SNAPSHOT.jar**” en todas las externalJar creadas de los proyectos: “**uniapi\_backend\_enviroment\_execution**”, “**uniapi\_backend\_business**”, “**uniapi\_backend\_apirest**”.

2. **Capa de ejecución**: Para esta capa realizaremos el mismo procedimiento en el proyecto “**uniapi\_backend\_enviroment\_execution**”. Utilizaremos maven con la instrucción descripta. Conseguiremos un denominado “**enviroment\_execution-0.0.1-SNAPSHOT.jar**” tras eso es necesario el trasladarlo a las capas superiores (“**uniapi\_backend\_business**”, “**uniapi\_backend\_apirest**”) a sus carpetas externalJar.

3. **Capa de negocio**: Realizaremos las mismas instrucciones en el proyecto “**uniapi\_backend\_business**” conseguiremos un archivo denominado “**business-0.0.1-SNAPSHOT.jar”** en la carpeta target correspondiente a su proyecto, este archivo será necesario trasladarlo al externalJar del proyecto “**uniapi\_backend\_apirest**”.

4. **Capa REST**: Esta capa es la última. Para que todo funcione correctamente deberá albergar todos los ejecutables como sigue la siguiente figura:

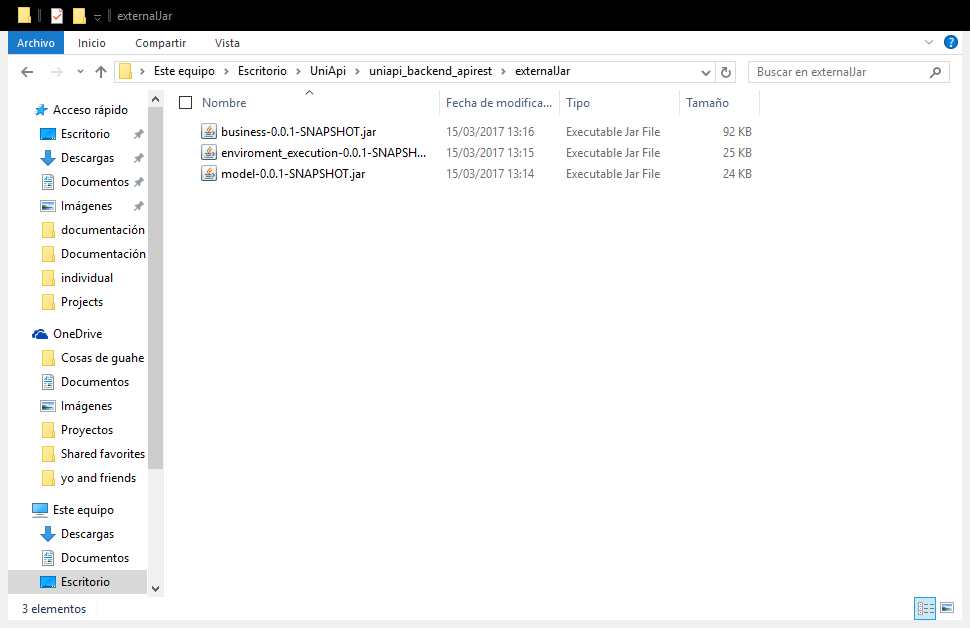


Ilustración . Estado final externalJar del proyecto REST

No es necesario aplicar la compilación a esta capa ya que no se necesita su ejecutable como dependencia. No obstante, puede hacerse para asegurarse que no hay errores si la compilación es correcta. Existirá en su target un archivo denominado “**apirest-0.0.1-SNAPSHOT.jar**”.

**Posibles errores**:

1. **Fallo en las dependencias**: Asegúrate que los archivos que pasas a las carpetas externalJar, contienen los mismos nombres que los descritos en este documento.
2. **JRE Than JDK**: Este fallo se da en los proyectos maven que no tienen un JRE igual que el del compilador que enlaza eclipse. Para eso deberás entran en las propiedades del proyecto (botón derecho en el proyecto). Buscar la pestaña Java y la subpestaña Java Compiler. También deberías observar el Java Buid Path, y comprobar que es la librería JavaSE es el mismo que el JDK.

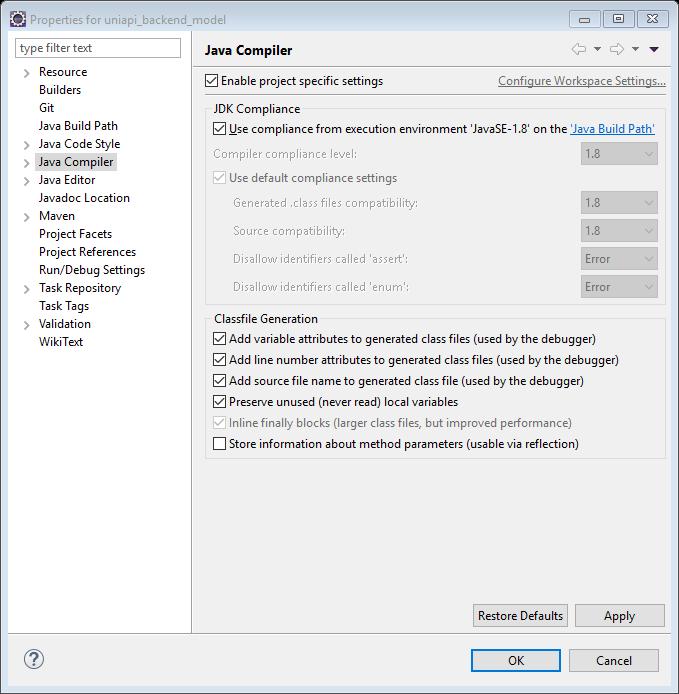


Ilustración . Comprobación JDK

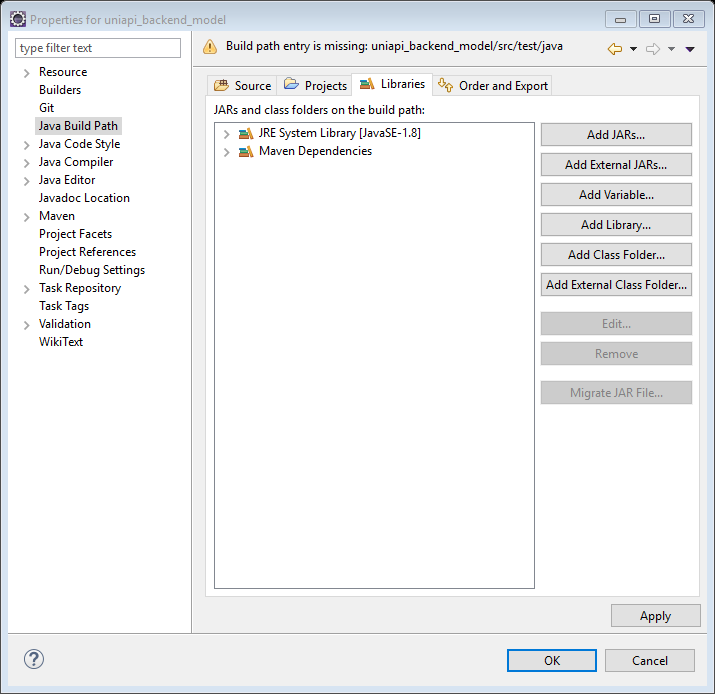


Ilustración . Comprobación JavaSE

* 1. **Instalación de base de datos:**

Para realizar este repositorio se ha utilizado como base de persistencia la base de datos Neo4j. Para la realización de esta instalación Neo4j difiere del sistema operativo en el que opere. Se describirán las dos instalaciones en función de su núcleo Windows o Linux:

* **Windows**: Para instalar la base de datos en Windows vamos a la siguiente dirección url.

<https://neo4j.com/download/community-edition/>

en él se nos descargara un .exe. Instalaremos ese ejecutable y se nos dispondrá un programa denominado neo4j Community editon.

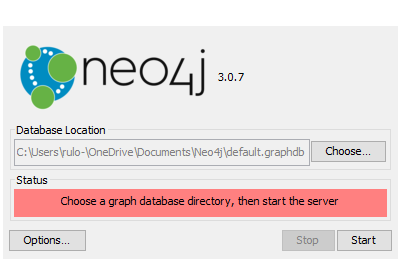


Ilustración . Ventana neo4j

Podemos instalar la base de datos donde queramos, no obstante, la carpeta uniapi tiene una carpeta denominada data. Esta carpeta tiene como objetivo albergar toda la información de la aplicación. Cuando Obtengamos la dirección que queramos solo pulsamos el botón Start, y se nos proporcionara en Status una dirección donde podremos acceder para hacer una gestión de consola.

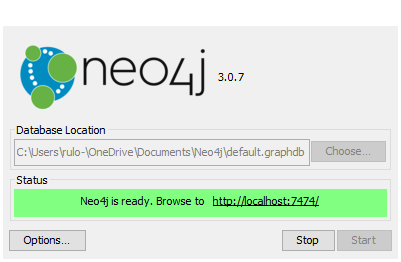


Ilustración . Ventana neo4j Iniciada

* **Linux**: Para instalar la base de datos en Linux iremos a la URL listada a continuación y descargamos la opción de Linux.

<https://neo4j.com/download/community-edition/>

Tras eso obtendremos un archivo comprimo donde se encontrará una carpeta denominada con neo4j-versionDeSoftware. La recomendación del desarrollador es que se descomprima la carpeta albergándola en la carpeta uniapi. Para iniciar la base de datos en linux, será necesario introducirse en la carpeta descomprimida. Buscar la carpeta bin y ejecutar el comando.

./neo4j-community-3.1.1/bin/neo4j console

* + 1. **Generación de estado inicial:**

Para poder realizar una correcta ejecución del repositorio es necesario que la base de datos este en un estado exacto o correcto. Para ello el repositorio necesita unas entidades mínimas para garantizar un correcto funcionamiento. Actualmente el desarrollador no tiene automatizado la generación del caso inicial. Aun así, cuando se realice estas instrucciones si se encuentran ya creadas las entidades es que se ha solventado este problema.

Para empezar, necesitaremos verificar si en la carpeta uniapi existe la carpeta **data**/ la cual deberá contener tres hijos **conf/**, **projects/**, **tmpExecution/**. Si no es así se requiere la creación de estas carpetas.

Después de eso, inicializaremos la base de datos siguiendo las instrucciones del anterior punto. Arrancaremos el navegador web e introduciremos la siguiente dirección URL:

<http://localhost:7474/>

Esto nos introducirá en la base de datos. Nos solicitara que cambiemos la contraseña, lo ejecutaremos e iremos al archivo de la carpeta **data/conf/uniapiConfig.properties** ahí cambiaremos las credenciales a las que instalamos para que el repositorio pueda conectarse a la base de datos.

Después de esto generaremos dos entidades necesarias para el funcionamiento de la aplicación. La entidad administradora original y a entidad del grupo público. Para ello escribimos las siguientes Querys en el lenguaje chyper.

1. CREATE (a:UserLogin {user:”admin@admin.com”,pass:admin,creationTime:0,rol:ADMIN,hashcode:””})
2. CREATE(g:Group{name:”public”,creationDate:0,type:”MAIN\_GROUP”,hashcode:”publicHASHCODE”,sharingGroupPermissions:”[YES,YES,YES]”,projectPropertiesPermissions:”[YES,YES,YES]”,memberGestionPermissions: ”[YES,YES,YES]”,groupCreationPermissions:”[YES,YES,YES]”,description:”PUBLIC GROUP”})
3. MATCH (u:UserLogin),(g:Group) WHERE u.user=”admin@admin.com” AND g.hashcode=”publicHASHCODE” CREATE (u)-[k:KNOWS {dateFrom:0}]->(g);

Después de esto, se podrá inicializar el repositorio sin fallo alguno.

* 1. **Ejecución del repositorio:**

Si se ha realizado correctamente los pasos tendremos por una parte todos los proyectos compilados e inyectados en los diversos proyectos, hasta llegar al proyecto final donde se tendrá todos los proyectos. Aquí tenemos dos tres opciones de ejecución del repositorio:

* **Ejecución mediante eclipse**: Seleccionamos el proyecto de “**uniapi\_backend\_apirest**. Aplicamos la misma metodologia que el punto 3.1 para los demás proyectos, solo que el comando a introducir en el espacio de Goals de la ventana de “maven build…” es diferente. Introduciremos entonces los siguientes comandos para generar la ejecución:  
  clean build package

spring-boot:run

* **Ejecución mediante consola y Maven**: para ello deberemos ir por la jerarquía de consola hasta el espacio de uniapi. Luego deberemos de entrar en el proyecto “**uniapi\_backend\_apirest**”. Después introducimos los siguientes comandos para generar la ejecución:

maven clean build package

maven spring-boot:run

* **Ejecución mediante .jar**: Si utilizamos el primer comando que introducimos en las otras dos opciones generamos un .jar, en la carpeta target. Este .jar puede ser ejecutado generando el siguiente comando:

java -jar apirest-0.0.1-SNAPSHOT.jar

1. **Instalación del servicio web:**
   1. **Instalación de Node.js:**

Para poder iniciar el servicio web del proyecto necesitaremos instalar el motor Node.js de JavaScript. Este motor puede ser instalado desde su página web principal donde puede ser instalado para diversos sistemas operativos. A continuación, se proporciona unas instrucciones básicas para su instalación en diversos sistemas operativos:

* **Windows**: Debe instalarse siguiendo esta dirección web. En él se descargará un instalador. Seguir las instrucciones, e instalarlo. Para poder manejarlo suele venir (en casos de W10) con un acceso directo a un prompt que viene desplegado la variable de entorno de Node.js. En caso contrario mirar en google como desplegar una variable de entorno en el sistema.

<https://nodejs.org/es/>

* **Linux**: Para Linux la cosa es más sencilla solo deberán ejecutar la siguiente instrucción y se instalara automáticamente. (la instrucción es para Ubuntu, para demás instrucciones usen el gestor de paquetes oportuno)

sudo apt-get install node

* 1. **Arranque servidor web:**

Para poder realizar el arranque del servidor web deberemos introducirnos con el prompt del sistema operativo que utilicemos en la jerarquía de la carpeta de UniApi.

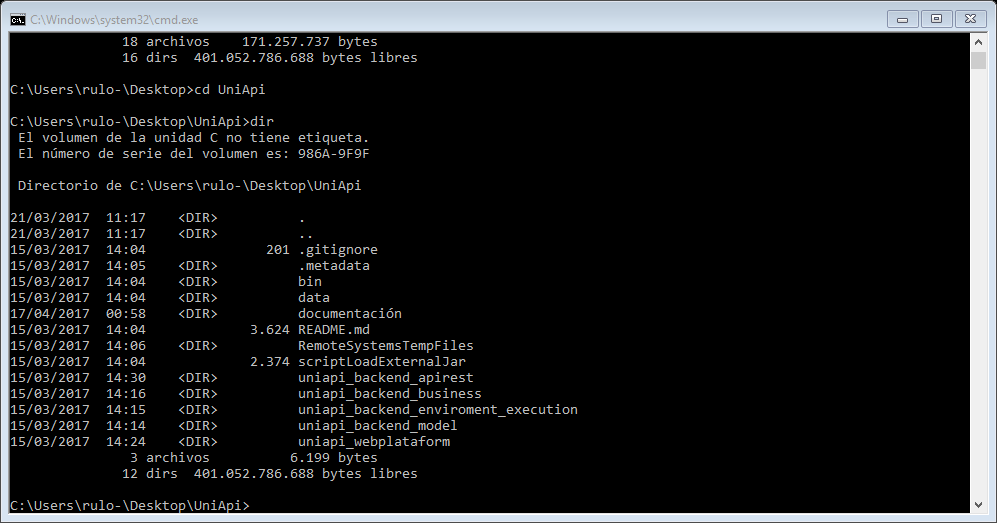


Ilustración . Jerarquía de uniapi (prompt)

Tras acceder a la jerarquía de UniApi deberemos introducirnos en la carpeta “**uniapi\_webplataform**” y tras eso ordenar a Node.js que ejecute el programa.

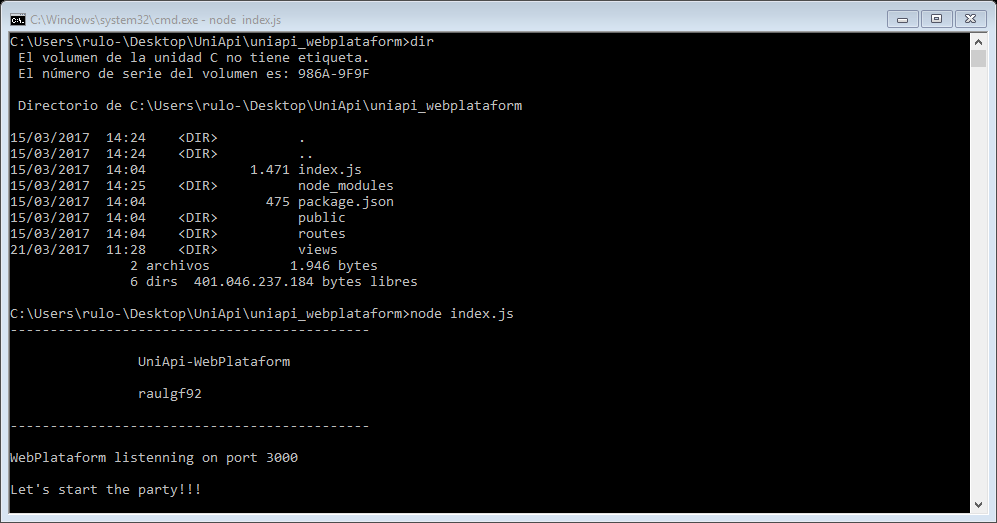


Ilustración . Ejecución del servidor web

1. **Arranque rápido en Ubuntu server:**

Para facilitar el desarrollo, el desarrollador ha producido un arranque rápido del sistema. Este arranque automatiza todas las tareas que están descritas en este documento. Cabe destacar que para que el arranque funcione, el sistema deberá estar en un estado de programas que serán descritos a continuación.

* 1. **Instrucciones de arranque:**

Para arrancar de manera rápida deberemos ubicarnos en la carpeta de la jerarquía principal (carpeta uniapi) mediante el Bash de Ubuntu (conocido como terminal). Tras estar en la ubicación necesaria realizamos la ejecución de las siguientes instrucciones:

cd bin/Linux

./uniapi

* 1. **Listado de programas:**

Los siguientes programas deberán estar instalados correctamente para que funcione el arranque y no se cree que incluso así, este funcione correctamente.

* Maven
* Node.js
* Neo4j (ubicado en la carpeta uniapi)

1. **Manual de creación de proyectos:**

Para poder ejecutar un proyecto en UniApi son necesarios ciertos requisitos para la realización de los proyectos. Hay que decir que actualmente el desarrollador no ha tenido en cuenta el análisis y monitorización del diseño de los proyectos. Es por eso que será parte del usuario el saber realizarlo.

* Main: Todos los programas tienen un main. Es un archivo donde se inicializa todos los proyectos. Es necesario que este proyecto esté en la carpeta superior del proyecto. En el caso que no se pueda introducir, hay que poner un path en el nombre main cuando se crea el proyecto.
* Argumentos: Los argumentos que se entregarán al programa serán por consola. Es decir que se necesitara controlar en el proyecto el conocimiento de conseguir los argumentos de la variable de entorno. Es recomendable que se trate la gestión de variables de entorno ya que el repositorio no lo hace. Hay que tener en cuenta que los argumentos siempre serán +1. Eso quiere decir que al final de los argumentos se añadirá un argumento final, con el path donde se quiere que se devuelva la respuesta.
* Respuesta: Es obligatorio para el proyecto que la respuesta se de en el path que se explicó en el punto anterior. Tendrá que ser de conocimiento del usuario el saber escribir un fichero en el lenguaje que maneja. El contenido de la respuesta debe estar escrito en JSON para que sea entendido por el repositorio. Existen muchas librerías que facilitan esto. Es obligación del usuario conocerlas.

1. **Manual de utilización del repositorio:**

El repositorio utiliza como método de comunicación el protocolo HTTP. Este protocolo contiene unas directrices especiales que hay que conocer. Pero si se quiere trabajar fácilmente con él, cualquier navegador o ciertas librerías de Python facilitan el uso de HTTP.

* 1. **Iteración habitual:**

Antes de empezar a utilizar el repositorio deberás autenticarte usando la parte del protocolo de comunicación 6.2.1. Si se autentica correctamente se le otorgara una ID de acceso. Después de eso cualquier acción que se ordene al sistema deberá contener el ID. Todas las acciones son sin estado, es decir, terminado la acción el sistema olvidara la acción realizada.

* 1. **Protocolo de comunicación**

El lenguaje que entiende el repositorio es el protocolo HTTP. Son ordenes realizadas mediante un protocolo que debe ser conocido para poder comunicarse a este nivel, No obstante, el navegador o diversas librerías de Python facilitan la comunicación. En este protocolo se entiende que TODAS las entidades (proyectos, grupos, usuarios) están representadas por un ID que será el que lo identifique. Cuando se quiera realizar una acción se deberá saber los ID’s de los afectados.   
http://DIRECCIONIP:8080/ACCION/QUE/SE/ORDENA

Para poder conectarnos al repositorio deberemos conocer la dirección IP o el DNS donde se alberga el servidor, tras eso se tendrá que conectar al puerto :8080 donde se albergara el repositorio. Luego para ordenar una acción se utiliza dominós de URL. Posteriormente se ha realizado una serie de tablas para facilitar la comprensión del repositorio.

http://127.0.0.1:8080/raulgf92@gmail.com/contrasena1

* + 1. **Autenticación:**

Tabla . Protocolo repositorio: Autenticación

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salidas |
| /{email}/{contraseña}  (HTTP method:GET) | Acción que permite la autenticación en el sistema. | Vacío | Mensaje con el ID del usuario. |
| /{idUser}/whoami  (HTTP method:GET) | Acción que devuelve la entidad del usuario que usa el id | Vacío | Devolverá un mensaje con los datos de autenticación y personales. |

* + 1. **Grupos:**

Todas estas acciones llevan por delante el comando **“/group/”** se deberá de concatenar por delante para que acción funcione **(dir+/group/+accion)**.

* + - 1. **Grupos acciones básicas:**

Tabla .Protocolo repositorio: Grupo acciones básicas

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salida |
| /{idUser}/{groupID}  (HTTP method:GET) | Devuelve la entidad correspondiente al id del grupo entregado | Vacío | Devuelve un mensaje con la entidad de grupo solicitada. |
| /{idUser}/{groupID}  (HTTP method:DELETE) | Elimina la entidad correspondiente al id del grupo entregado | Vacío | Devuelve el resultado de la acción realizada. |
| /{idUser}/{groupID}  (HTTP method:PATH) | Modifica la entidad correspondiente al id del grupo entregado | Entidad grupo con los datos modificados que se va a modificar | Devuelve el resultado de la acción realizada |
| /{idUser}/all  (HTTP method:GET) | Devuelve todos los grupos creados por el usuario | Vacío | Devuelve un listado de ID’s de grupos que fueron creados por el usuario. |
| /{idUser}/create  (HTTP method:GET) | Facilita la creación de los grupos mediante un ejemplo | Vacío | Devuelve una entidad grupo con datos rellenados con ejemplos para poder crear el grupo. |
| /{idUser}/create  (HTTP method:POST) | Crea el grupo entregado por el usuario | Entidad grupo con los datos que se quieren crear | Devuelve el resultado de la acción realizada |

* + - 1. **Grupos miembros:**

Tabla .Protocolo repositorio: Grupo miembros

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salidas |
| /{idUser}/{groupID}/member  (HTTP method:GET) | Acción que permite visualizar todos los miembros de un grupo | Vacío | Devuelve un conjunto de ID’s de usuarios. |
| /{idUser}/{groupID}/member/create  (HTTP method:GET) | Acción que permite conseguir un ejemplo de método de entrada para crear miembros de grupos | Vacío | Devuelve una entidad mensaje que se podrá usar para comunicar con el sistema. |
| /{idUser}/{groupID}/member/create  (HTTP method:POST) | Acción que permite crear miembros para un grupo. | Mensaje de comunicación para generar un miembro | Devolverá el resultado sobre la ejecución de la acción. |
| /{idUser}/{groupID}/  member/{memberID}  (HTTP method:DELETE) | Acción que permite borrar un miembro de un grupo | Vacío | Devolverá el resultado sobre la ejecución de la acción |

* + - 1. **Grupos subgrupos:**

Tabla .Protocolo repositorio: Grupo subgrupos

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salidas |
| /{token}/{groupID}/subgroups  (HTTP method:GET) | Accion que devuelve todos los subgrupos de un grupo | Vacío | Devuelve un conjunto de ID’s de grupos |
| /{token}/{groupID}/subgroups/{subgroupID} (HTTP method:GET) | Acción que permite conseguir la información de que un grupo es subgrupo de otro | Vacío | Devuelve una entidad mensaje que contiene la información de la orden |
| /{token}/{groupID}/subgroups/{subgroupID}  (HTTP method:POST) | Acción que permite hacer que un grupo sea subgrupo de otro grupo | Vacío | Devolverá el resultado sobre la ejecución de la acción. |
| /{token}/{groupID}/subgroups/{subgroupID} (HTTP method:DELETE) | Acción que permite revertir que un grupo sea subgrupo de otro | Vacío | Devolverá el resultado sobre la ejecución de la acción |

* + - 1. **Grupos proyectos:**

Tabla .Protocolo repositorio: Grupo proyectos

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salidas |
| /{idUser}/{groupID}/contain/project  (HTTP method:GET) | Acción que permite ver todos los proyectos contenidos en un grupo. | Vacío | Devuelve un mensaje con un conjunto de ID’s de proyectos contenidos en un grupo |
| /{idUser}/{groupID}/contain/project/{projectID}  (HTTP method:GET) | Acción que permite visualizar la información de cuando un proyecto es contenido en un grupo | Vacío | Devuelve un mensaje con la información de la acción |
| /{idUser}/{groupID}/contain/project/{projectID}  (HTTP method:POST) | Acción que permite introducir proyectos en el grupo | Vacío | Devolverá el resultado sobre la ejecución de la acción. |
| /{idUser}/{groupID}/contain/project/{projectID}  (HTTP method:DELETE) | Acción que permite borrar un proyecto de un grupo | Vacío | Devolverá el resultado sobre la ejecución de la acción |

* + 1. **Proyectos:**

Todas estas acciones llevan por delante el comando **“/project/”** se deberá de concatenar por delante para que acción funcione **(dir+/project/+accion).**

Tabla .Protocolo repositorio: Proyectos

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salida |
| /{idUser}/{projectID}  (HTTP method:GET) | Devuelve la entidad correspondiente al id del proyecto entregado | Vacío | Devuelve un mensaje con la entidad de proyecto solicitada. |
| /{idUser}/{projectID}  (HTTP method:DELETE) | Elimina la entidad correspondiente al id del proyecto entregado | Vacío | Devuelve el resultado de la acción realizada. |
| /{idUser}/{projectID}  (HTTP method:PATH) | Modifica la entidad correspondiente al id del proyecto entregado | Entidad proyecto con los datos modificados que se va a modificar | Devuelve el resultado de la acción realizada |
| /{idUser}/all  (HTTP method:GET) | Devuelve todos los proyectos creados por el usuario | Vacío | Devuelve un listado de ID’s de proyecto que fueron creados por el usuario. |
| /{idUser}/create  (HTTP method:GET) | Facilita la creación de los proyectos mediante un ejemplo | Vacío | Devuelve una entidad proyectos con datos rellenados con ejemplos para poder crear el proyecto. |
| /{idUser}/create  (HTTP method:POST) | Crea el proyecto entregado por el usuario | Entidad proyecto con los datos que se quieren crear | Devuelve el resultado de la acción realizada |

* + 1. **Ejecuciones:**

Todas estas acciones llevan por delante el comando **“/execution/”** se deberá de concatenar por delante para que acción funcione **(dir+/execution/+accion)**.

Tabla .Protocolo repositorio: Ejecuciones

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salida |
| /{idUser}/{groupID}/{projectID}  (HTTP method:POST) | Crear una entidad ejecución y la ejecuta. Usando un proyecto con projectID albergado en el grupo de groupID | Archivo JSON, que será utilizado como la entrada de la ejecución del proyecto. | Devuelve el resultado de la acción realizada |
| /{idUser}/{executionID}  (HTTP method:GET) | Devuelve la entidad correspondiente al id de la ejecución entregada | Vacío | Devuelve un mensaje con la entidad de ejecución solicitada. |
| /{idUser}/{executionID}  (HTTP method:DELETE) | Elimina la entidad ejecución representada por executionID. En el caso de que este en ejecución se detendrá. | Vacío | Devuelve un mensaje con la entidad de ejecución solicitada. |
| /{idUser}/{executionID}/project  (HTTP method:GET) | Devuelve la información del proyecto que está ejecutando la entidad ejecución representada por el executionID | Vacío | Devuelve un mensaje con la entidad del proyecto de la ejecución solicitada. |
| /{idUser}/running  (HTTP method:GET) | Devuelve todas las ejecuciones del usuario que se están ejecutando. | Vacío | Devuelve un mensaje con el conjunto de las identidades de las ejecuciones en ejecución |
| /{idUser}/finish  (HTTP method:GET) | Devuelve todas las ejecuciones del usuario que hayan finalizado | Vacío | Devuelve un mensaje con el conjunto de las identidades de las ejecuciones finalizadas |

* + 1. **Administración:**

Todas estas acciones llevan por delante el comando **“/admin/”** se deberá de concatenar por delante para que acción funcione **(dir+/admin/+accion)**.

Tabla .Protocolo repositorio: Administración

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salida |
| /{idUser}/createAccount  (HTTP method:POST) | Crea una cuenta con el rol que elija el administrador | Entidad userLogin con los datos rellenos para su creación | Devuelve el resultado de la acción realizada |
| /{idUser }/allUsers  (HTTP method:GET) | Devuelve todos los usuarios del sistema | Vacío | Devuelve un mensaje con el conjunto de identidades de usuarios |
| /{idUser}/allGroups  (HTTP method:GET) | Devuelve todos los grupos del sistema | Vacío | Devuelve un mensaje con el conjunto de identidades de grupos |
| /{idUser}/allProjects  (HTTP method:GET) | Devuelve todos los proyectos del sistema | Vacío | Devuelve un mensaje con el conjunto de identidades de proyectos |
| /{idUser}/allExecution  (HTTP method:GET) | Devuelve todas las ejecuciones del sistema | Vacío | Devuelve un mensaje con el conjunto de identidades de ejecuciones |

* + 1. **Navegación:**

Todas estas acciones llevan por delante el comando **“/path/”** se deberá de concatenar por delante para que acción funcione **(dir+/path/+accion)**.

Tabla .Protocolo repositorio: Navegación

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Descripción | Entradas  (Body) | Salida |
| /{idUser}/main  (HTTP method:GET) | Devuelve los proyectos públicos y los grupos superiores del usuario | Vacío | Devuelve un mensaje con todas las identidades de proyectos y grupos que contiene la navegación actual |
| /{idUser}/enter/{groupID}  (HTTP method:GET) | Devuelve todos los proyectos y grupos que puede visualizar el usuario | Vacío | Devuelve un mensaje con todas las identidades de proyectos y grupos que contiene la navegación actual |
| /{idUser}/public  (HTTP method:GET) | Devuelve la información del grupo publico | Vacío | Devuelve un mensaje con la entidad del grupo publico |

* 1. **UniApi (JS):**

Para facilitar el desarrollo de aplicaciones web utilizando el repositorio, se ha diseñado una librería para angular.js extrapolable a cualquier tecnología que use JavaScript (utilizando JQuery). Esta API está hecha mediante funciones que utilizan el protocolo explicado en el punto 6.2. La API facilita a los desarrolladores de aplicaciones web que no quieren entender el protocolo de comunicación del repositorio el usarlo.

Para poder usarlo. Deberemos instalarlo en la página web incluyéndolo con el siguiente trozo de código HTML. Cabe destacar que deberá poder ser accedido por el servidor web que sustenta la aplicación web. Las funciones son:

Tabla UniApi.js: Nombre de funciones

|  |  |  |
| --- | --- | --- |
| Nombre de las funciones | | |
| Whoami() | **getMyProjects**() | **deleteSubgroup**() |
| changePass() | **getMyGroups**() | **getMainPathGroups**() |
| changeBio() | **getProject**() | **getPathGroups**() |
| createAccount() | **updateProject**() | **projectsInsideGroup**() |
| getAllUsers() | **deleteProject**() | **getProjectsInsideGroup**() |
| getAllProjects() | **createProject**() | **getExecutionsRunning**() |
| getAllGroups() | **createGroup**() | **getAllExecutionsRunning**() |
| getGroup() | **updateGroup**() | **getExecution**() |
| deleteGroup() | **inviteToGroup**() | **getProjectOfExecution**() |
| getAllGroupMembers() | **removeMemberOfTheGroup**() | **getExecutionFinish**() |
| getSubgroups() | **getPathProjects**() | **getAllExecutionsFinish**() |
| createSubgroup() | **putGroupProject**() | **executedProject**() |
| getMainPathProjects() | **deleteProjectInsideGroup**() |  |

1. **Manual de utilización de la aplicación web:**

La aplicación web utilizara para comunicarse con el usuario una serie de formularios y listados que facilitaran al usuario la interacción y gestión con el repositorio. A continuación, se desplegará unos tutoriales para realizar diferentes tareas.

* + 1. **Autenticarse y crear una cuenta:**

Cuando accedemos a la aplicación web se nos direccionara a un espacio donde tendremos que autenticarnos. En este espacio se nos dispondrá de un formulario para meter el nombre de usuario (email del usuario) y una contraseña.

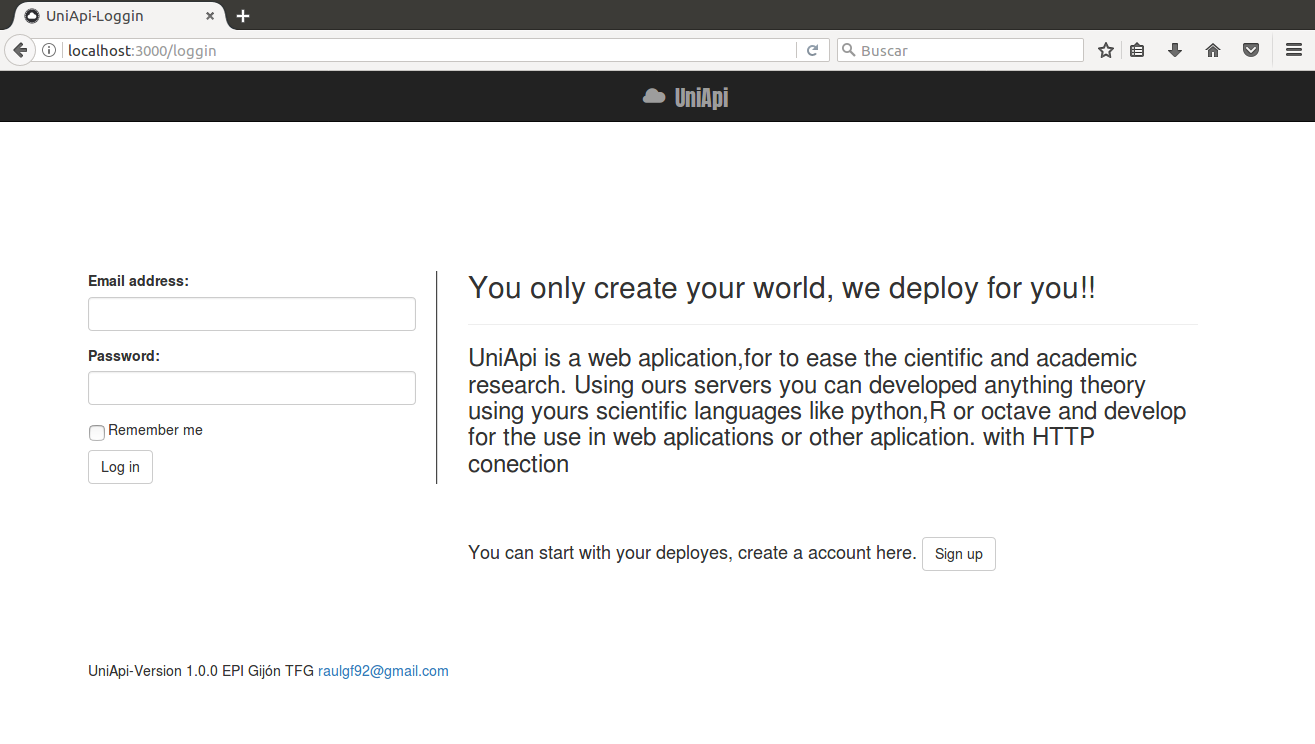


Ilustración Ventana Loggin



Ilustración Página Principal

Si se ha autenticado correctamente se le pasara a la página principal de la aplicación desde donde todos los demás tutoriales empiezan. En caso de que la contraseña se incorrecta se desplegara un cartel informativo declarando que la identidad es incorrecta. Y en el caso de que no dispongamos de identidad se pulsara el botón “Log in”.

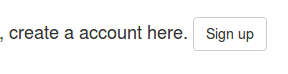


Ilustración Botón sign up

Automáticamente el sistema nos redirigirá a una página con un formulario para crearnos una nueva cuenta. En el tendremos que introducir nuestros datos, y posteriormente darle al botón “Create”

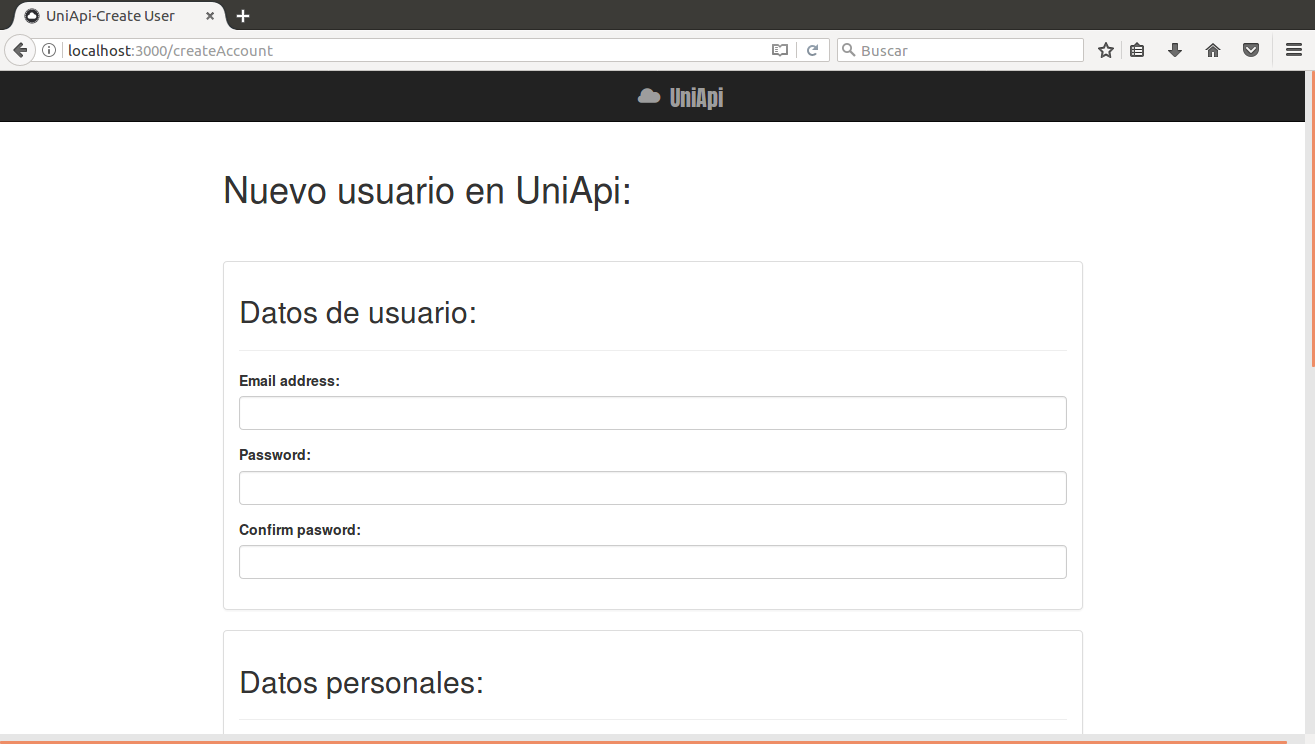


Ilustración Ventana nuevo usuario

Si no hay fallos de validación de los datos. Se le devolverá al usuario a la ventana de loggin donde podrá introducir el nombre de usuario y su contraseña.

* + 1. **Navegar por los grupos:**

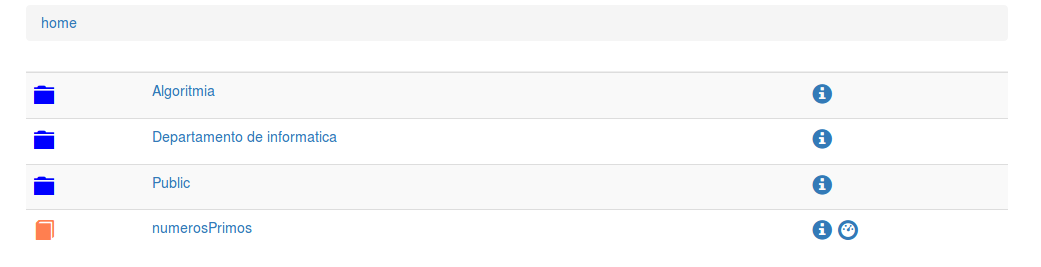
Cuando accedemos a la página principal de la aplicación se nos mostrara un menú lateral con todas las opciones de la aplicación. Dentro de este menú debemos fijarnos en los dos elementos del final del menú “Nuevo proyecto” y “Nuevo grupo”, ya que más tarde los usaremos en este tutorial. En el centro de la página principal se despliega un menú de navegación con todos los grupos y proyectos que podemos visualizar, algo como se muestra en la figura siguiente.

Ilustración Menú Navegación

En el menú de la navegación tenemos 3 elementos principales:

* Menú superior: El menú de navegación superior, es un menú que ira registrando la navegación de la jerarquía de los grupos. Si pulsamos en los nombres de los grupos nos redirigirá a ellos.
* Grupos: Son los elementos de la carpeta azul, contiene un elemento con la letra i. Esta nos llevara a la información del grupo. Si pulsamos en el susodicho grupo se activará la navegación hacia ese grupo, moviéndose el grupo en el que nos encontrábamos al menú superior de navegación.
* Proyectos: Son los elementos del libro naranja. Contienen dos elementos una i y un símbolo de un indicador. El símbolo de la i referencia a la información del proyecto, mientras que el símbolo del indicador nos lleva a realizar una ejecución sobre el proyecto.

Si en el proceso de navegación entre los grupos, pulsamos alguno de los dos elementos “Nuevo grupo” o “Nuevo proyecto” del menú lateral izquierdo como ha sido descrito al principio de este tutorial. La aplicación supondrá que se quiere crear uno de los dos elementos en el grupo que actualmente se está posicionado. El sistema comprobara los permisos del grupo y el rol del usuario para ver si es posible la realización de la creación.

* + 1. **Crear un grupo:**

Tras situarnos en un grupo que queramos crear un subgrupo y que estemos seguros que tenemos permisos, pulsamos el elemento del menú lateral izquierda denominado “Nuevo proyecto”. Automáticamente la aplicación se movilizará para disponerte un formulario para crear un grupo.

En el formulario se encontrará todos los datos que deberás introducir para crear un grupo. Hay que tener en cuenta que el nombre de un grupo es inmutable y no podrá cambiarse. La descripción sirve para introducir lo que quiera el usuario. Los tipos de grupo identifican la privacidad del grupo:

* Publico: Cualquiera que este navegando por un grupo que contenga este grupo como subgrupo podrá visualizar este grupo.
* Privado: Solo puede ser visualizado por los miembros que sean invitados por el dueño. El administrador también podrá acceder.

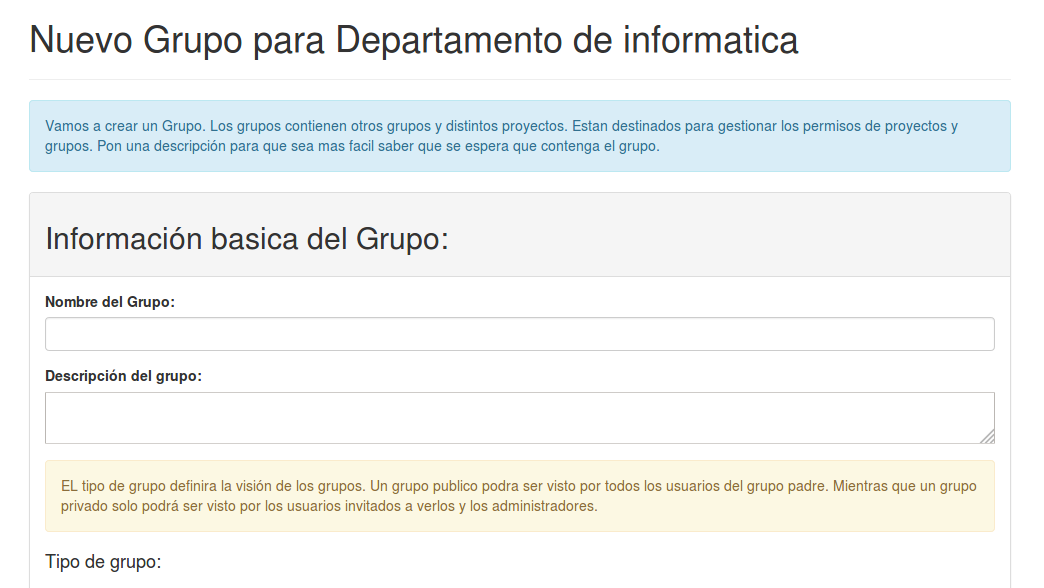


Ilustración Nuevo grupo

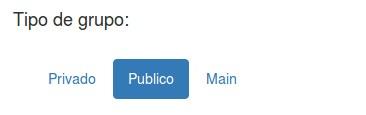


Ilustración Nuevo grupo: Tipo grupo

Tras crear los datos básicos del grupo como su nombre y tipo, ahora debemos configurar los permisos del grupo. Los permisos dan autorización a acciones sobre ejecuciones, proyectos, otros grupos y miembros del grupo. Estos grupos son heredados por el grupo que va a albergar al padre. Cabe destacar que los permisos son de SI o NO. Y si un permiso es negado por el padre del grupo a crear este no podrá poder modificar la negación.

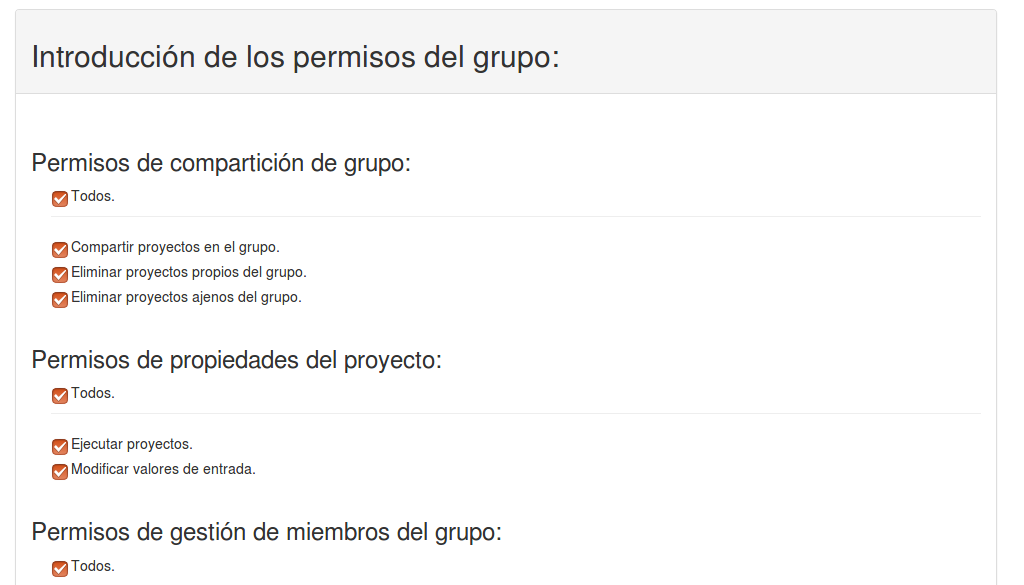


Ilustración Nuevo grupo: Permisos

Para terminar, solo tendremos que comprobar los datos y darle al botón crear. Saldrá un mensaje explicativo determinando si la creación se ha realizado.

* + 1. **Crear un proyecto:**

Para crear un proyecto deberemos navegar al grupo que deseemos después pulsaremos en el elemento del menú lateral izquierdo denominado “Nuevo proyecto”. Deberemos de tener en cuenta que tengamos los permisos para poder realizar la creación o la asociación. No obstante, si no tenemos los permisos la aplicación nos avisara de que no tenemos permisos.

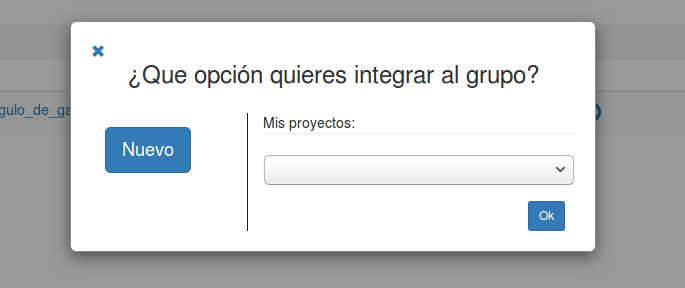


Ilustración Nuevo proyecto: Integración o creación

Cuando pulsemos el elemento del menú se desplegará una ventana como el de la figura 25. En él nos preguntara si queremos agregar un proyecto de los realizados por el usuario. Para ello el sistema nos dispondrá un listado de los proyectos existentes del usuario. El usuario solo tendrá que seleccionar uno y pulsar el botón OK.

La aplicación también permite el realizar la creación de un proyecto desde cero. La aplicación se movilizará para desplegar un formulario que proporcione los medios para introducir toda la información para crear un proyecto.



Ilustración Nuevo proyecto: principal

Cabe destacar que el nombre del proyecto es inmutable, así como su tipo, pero eso será explicado junto con los demás datos:

* Datos básicos: Son el nombre, descripción, descripción de la entrada y descripción de la salida. Salvo el nombre, estos datos son mutables y se espera una descripción extensa.
* Datos de repositorio: Estos datos sirven para conectar al repositorio GIT, cabe destacar que es necesario su introducción. Son la URL, el usuario y la contraseña.
* Tipo de proyecto: Los tipos de proyecto determinan en que lenguaje se encuentra el proyecto y cuál va a ser su ejecución.

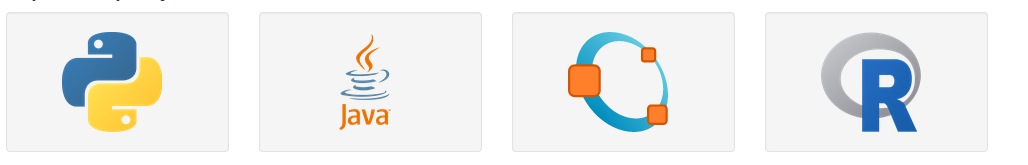


Ilustración Nuevo grupo: tipo de proyecto

* Datos de ejecución: Los datos de ejecución determinan toda la configuración de la ejecución, estos datos son el programa principal, el cual es la ubicación del fichero donde se va a ejecutar el programa. La respuesta es el nombre que se espera que se va a tener el archivo con la respuesta. Los inputs por defecto son las entradas que tendrá por defecto el proyecto están determinados mediante una columna de datos.

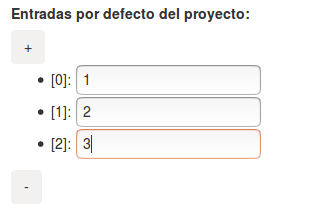


Ilustración Nuevo grupo: Entradas por defecto

Por ultimo solo se deberá revisar los datos y pulsar el botón crear grupo. La aplicación realizara la acción desplegando una ventana emergente informando del estado de la acción.

* + 1. **Crear una ejecución:**

Para crear una ejecución deberemos navegar primero por los grupos, buscando un proyecto que podamos ejecutar. Tras eso deberemos pulsar el icono del indicador. Tras eso la aplicación desplegara una ventana de ejecución. Esta ventana estará dividida en dos partes. Una parte la entrada y otra la salida (bloqueada evidentemente).

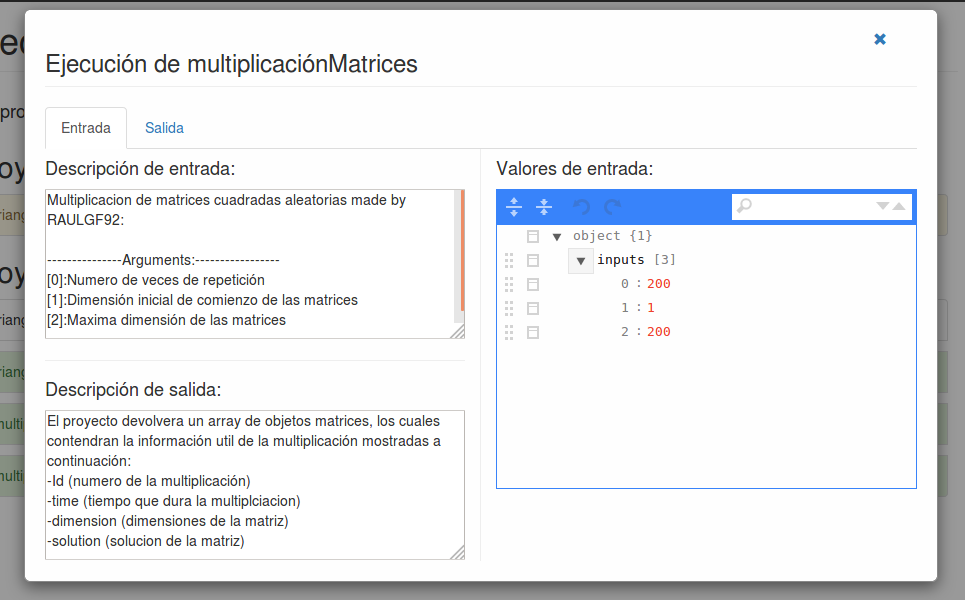
En la entrada se presentará la descripción de la entrada, es decir, los datos que se deberá poner en la sección de la derecha. Y debajo pone la descripción de la salida, es decir lo que se espera que salga. La sección de la derecha servirá para introducir datos de manera sencilla en un lenguaje máquina que entiende denominada JSON. Aunque se le puede meter cualquier tipo de dato nos interesa la introducción de lo que diga el creador del proyecto. Si le das al cuadro derecho se desplegará un menú. En él se deberá pulsar la opción “+Append”, he ir siguiendo con cada elemento hasta tener los que se desea. Tras eso se le dará al botón ejecutar y la ejecución se iniciará.

Ilustración Nueva ejecución

* + 1. **Gestionar un proyecto propio:**

Cuando queramos modificar alguna opción de un proyecto deberemos acceder al espacio del menú “Proyectos”. En él se encuentran todos los grupos creados por el usuario en cuestión. Solo deberemos pulsar en el grupo que deseemos saber la información de él. La aplicación se movilizará y desplegará una ventana modal con toda la información del proyecto.

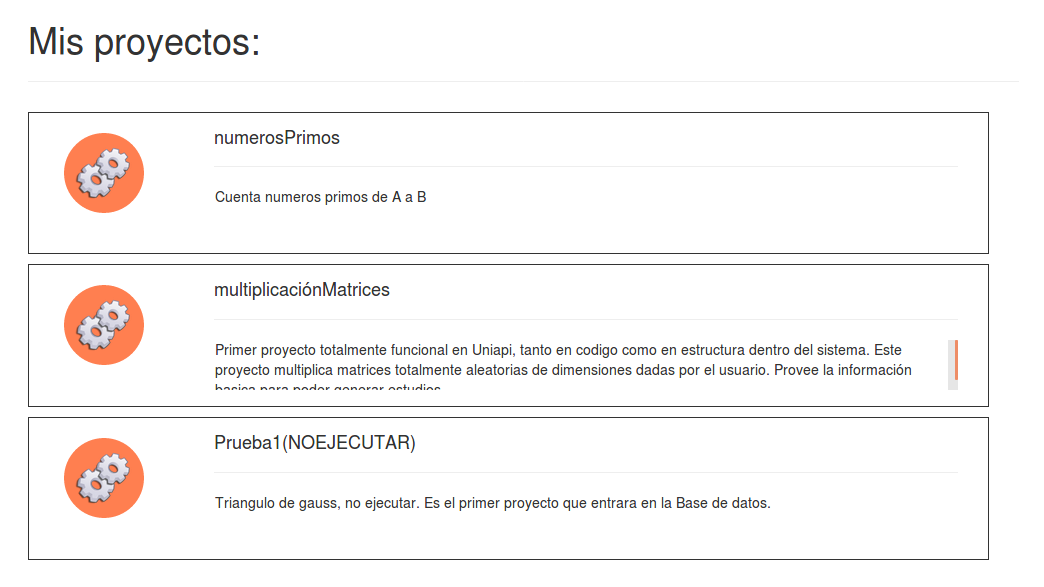
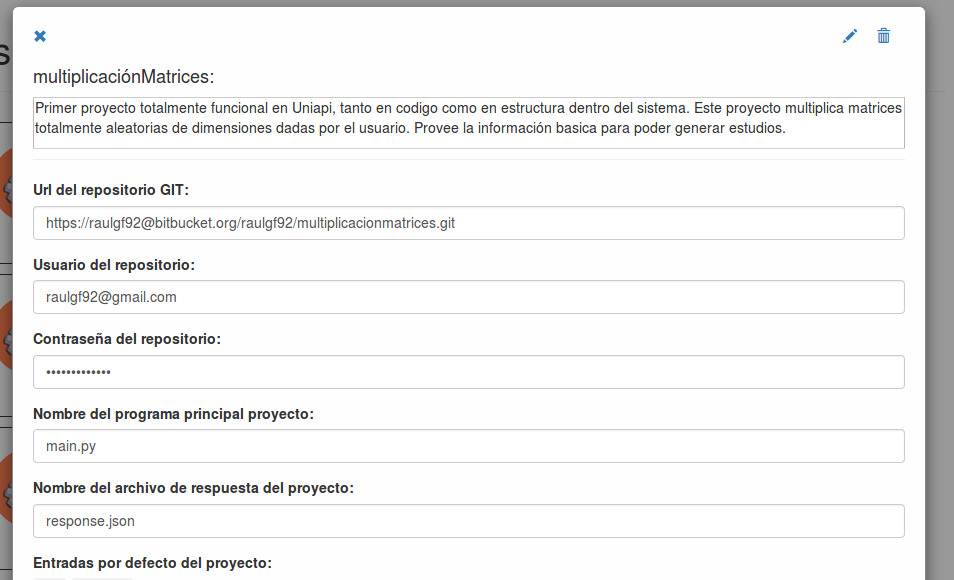


Ilustración Proyectos

Ilustración Gestión proyecto

Toda la información podrá ser modificada. Al final de la modificación solo tendrá que darle al botón del lápiz de arriba a la derecha para realizar una modificación del proyecto. Por el contrario, si quiere cancelar la modificación solo deberá cerrar la aplicación. La aplicación mandara un aviso emergente sobre el estado de la acción.

Si desea eliminar el proyecto deberá pulsar el botón de arriba a la derecha con la forma de un cubo de la basura. La aplicación preguntara si desea eliminar el proyecto. Tras la aceptación la aplicación mandara un aviso emergente con el estado de la acción.

* + 1. **Gestionar un grupo propio:**

Cuando se quiera gestionar o modificar un grupo, se deberá ir al espacio destinado a albergar el conjunto de los grupos creados por el usuario denominado “grupos”. En él se encuentra todos los grupos creados por el usuario. Para poder gestionar cualquier grupo solo se tendrá que pulsar en el grupo. Y la aplicación se movilizará para generar una ventana emergente con la información del grupo.

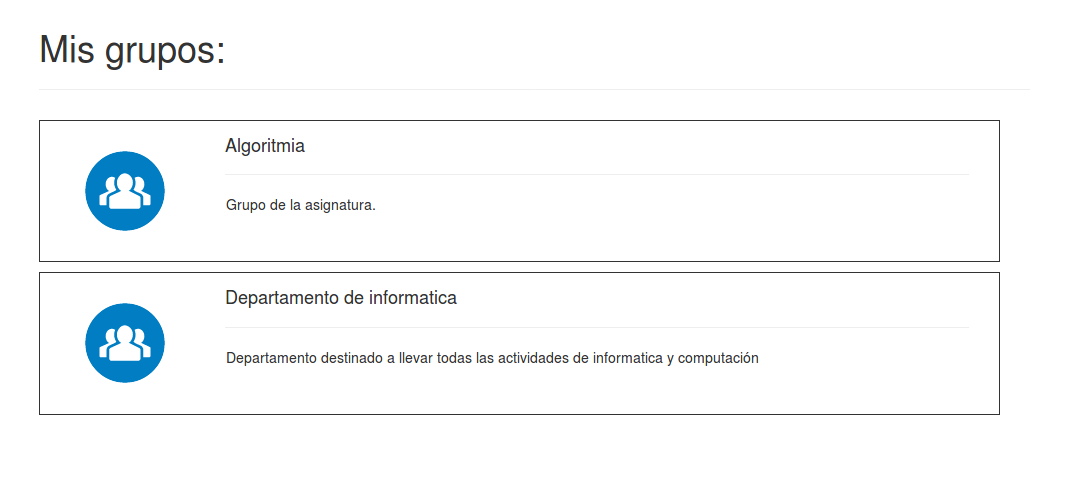


Ilustración Mis Grupos

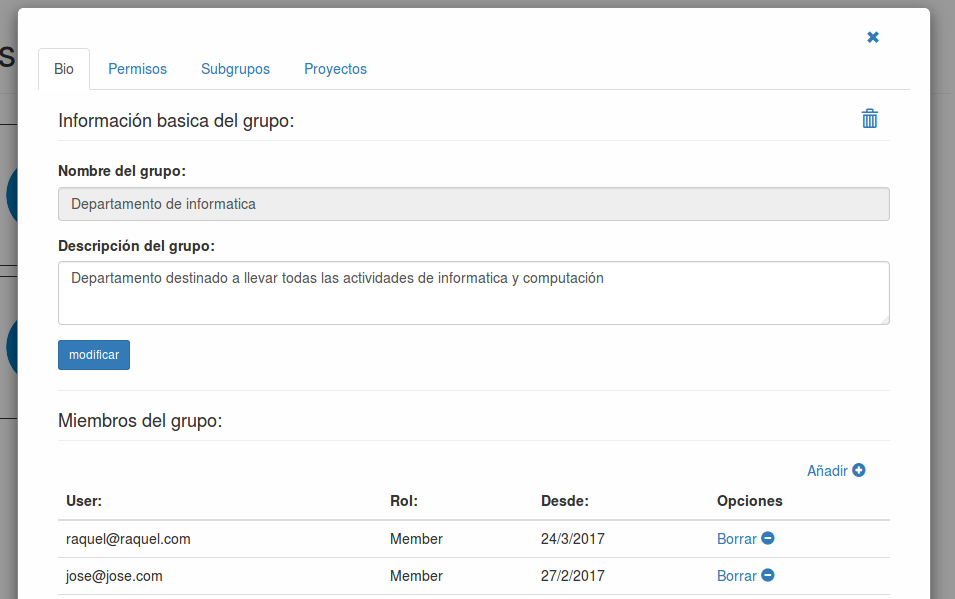
La ventana emergente esta ordenada en pestañas donde se gestionará cada diversos conjuntos de elementos. Cada uno será una lista mostrada como en la figura 33 que contendrá distinta información y dos acciones la de añadir o eliminar. Si se le da al botón de añadir se desplegará una opción donde podremos indicar lo que queremos añadir.

Ilustración Gestión grupo

* + 1. **Información básica:**

Para la información básica solo podremos modificar la descripción del grupo. Donde escribiremos lo que queramos y pulsaremos a modificar. Tras pulsar el botón la aplicación informara al usuario como ha salido la modificación. En la parte superior tenemos el botón de forma de cubo de basura, este botón realizara la eliminación del grupo. Una eliminación del grupo. Conlleva la eliminación de todos los subgrupos hijos de él y sus integraciones de proyectos (pero no los proyectos).

* + 1. **Miembros:**

Los miembros es una lista con el conjunto de todos los miembros de un grupo, en el podremos añadir miembros pulsando al botón de arriba “Añadir”. La aplicación desplegara un formulario donde podremos introducir el email de usuario del sistema que quieres invitar. Por el contrario, si se quiere eliminar un usuario, solo se tendrá que pulsar en el botón “Borrar” asociado a su fila. La eliminación consiste en la expulsión del usuario del grupo, pero no la eliminación de los elementos creados o integrados por él.

* + 1. **Subgrupos:**

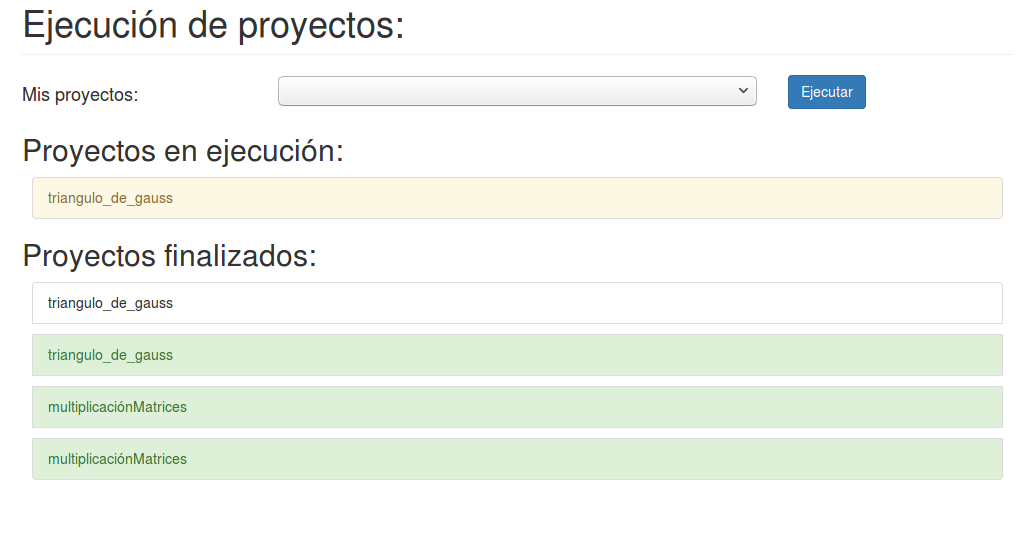
Subgrupos es una lista con todos los subgrupos de nivel 1 (hijos directos) que dependen del grupo. En la lista podremos eliminar cualquier grupo albergado en el grupo, sea público o privado. También podremos crear un grupo, lo que hará la aplicación será llevarnos al punto 7.1.3 para realizar una creación sobre este grupo. La eliminación de un subgrupo, es la eliminación de todos los subhijos del subgrupo y sus integraciones de proyectos.

* + 1. **Permisos:**

Los permisos son celdas iguales que cuando se crea los permisos, en ellos se pueden modificar los permisos y mandar a la aplicación modificarlos. Una modificación de los permisos genera una modificación en recursión, es decir, una modificación de todos los hijos y sus hijos…

* + 1. **Gestionar una ejecución propia:**

Para gestionar una ejecución propia, versión contenido, su estado… necesitamos entrar en el espacio de Ejecuciones, esta se puede acceder a través del menú lateral de la izquierda de la aplicación. La aplicación se moviliza generando un espacio con listas informativas de las ejecuciones. Estas se organizan en no finalizadas, la lista de arriba con elementos de color amarillo. Y la lista de finalizadas con elementos de color verde para éxito y rojo para fallo.



Si pulsamos en cualquier elemento se nos enviara a una ventana emergente con la gestión de la ejecución:

* En ejecución: Se nos enviará a la ventana de salidas, la respuesta estará vacía, la cual es una ventana igual que la descrita en la creación de nueva ejecución. Esta ventana traduce el lenguaje maquina a una interfaz que ayuda a la compresión del usuario. También nos indicara la cuenta de tiempo que lleva la ejecución.

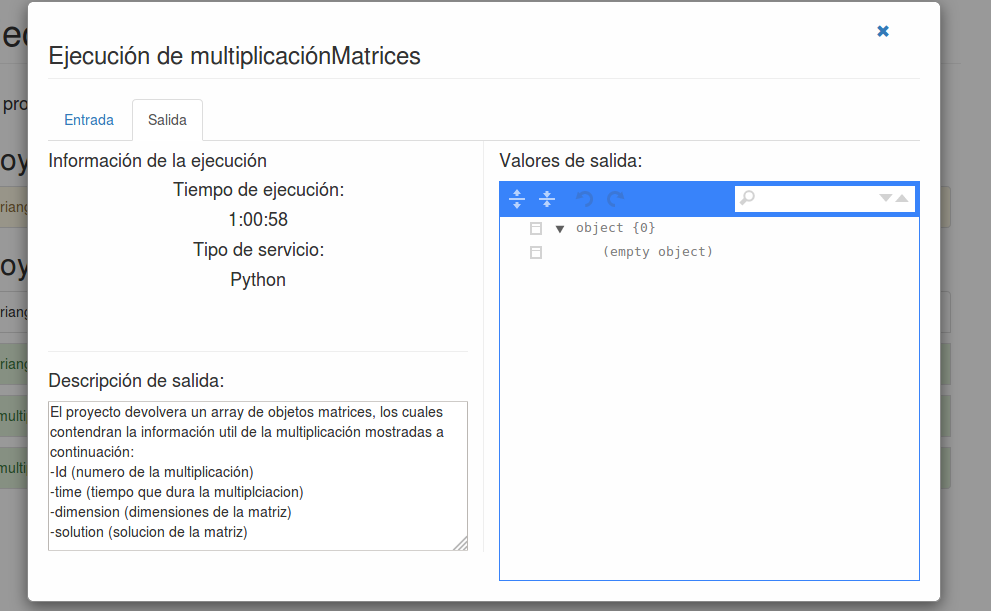


Ilustración Gestión de ejecución: En ejecución

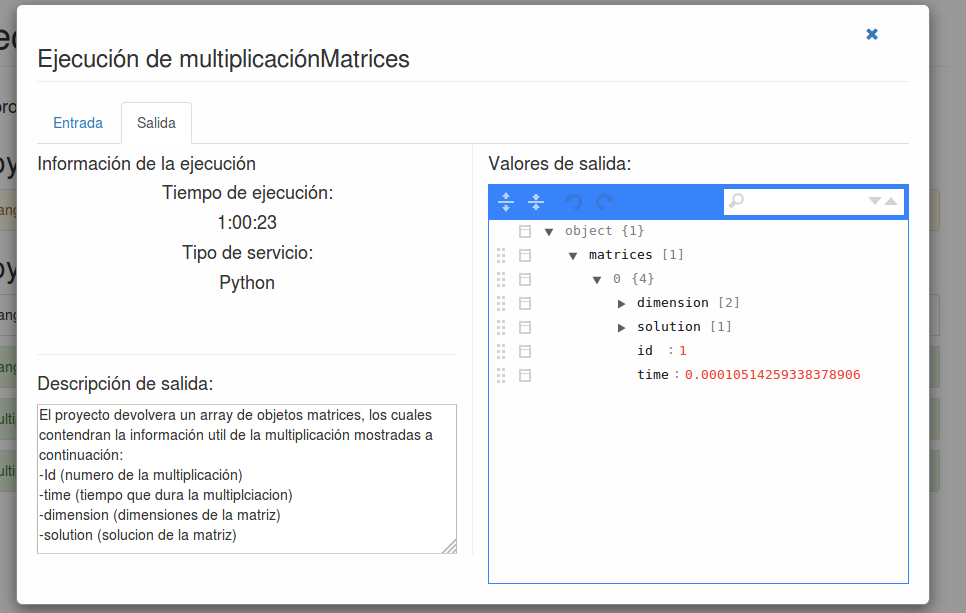
* Finalizado: Finalizado nos enviará a la ventana de salidas, la respuesta estará llena de los datos de respuesta de la ejecución del proyecto, salvo si ha finalizado con fallo que estará vacío. La ventana también nos dará el tiempo de la ejecución.

Ilustración Gestión de ejecución: Finalizado