

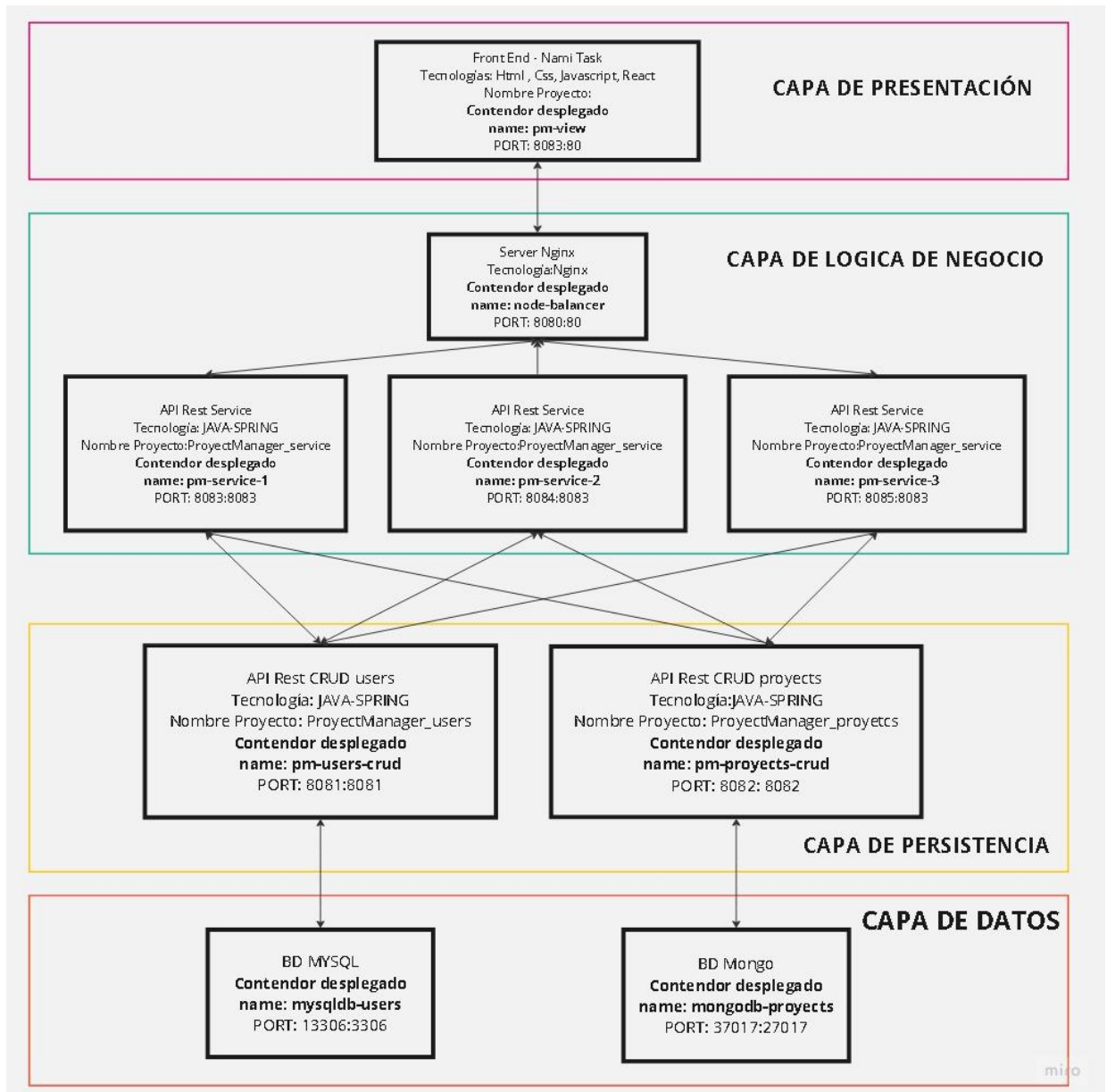
**UNIVERSIDAD CENTRAL DEL ECUADOR**  
**FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS**  
**CARRERA DE SISTEMAS DE INFORMACIÓN**  
**ARQUITECTURA DE SOFTWARE**

**MANUAL DE IMPLEMENTACION SISTEMA**

## ARQUITECTURA

### Arquitecturas presentes en el Sistema:

- API REST ful
- En capas
- DTO y DAO
- Node Balancer.
- Abstrac Factory



### ESTRUCTURA DE LAS BASES DE DATOS

- **MySQL:** Base de datos para Usuario

Utilizada para almacenar la información de usuarios, se usó SQL para su búsqueda rápida y estructurada

Tabla: users

users	
id	
email	
first_name	
last_name	
nick_name	
password	

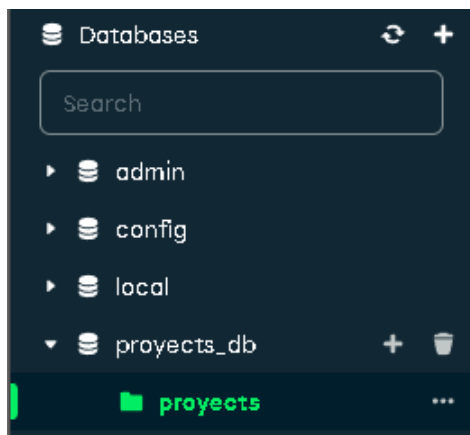
### Formato de registro:

	id	email	first_name	last_name	nick_name	password
1	1	cmorales@gmail.com	Cristian	Morales	cmorales	\$2a\$10\$uxRmAbLIRjYBIVASoyCGPO6GzW4KP19mmOOqilLusyWkT/7S7omvjy
2	2	gacaro@gmail.com	Gisela	Acaro	gacaro	\$2a\$10\$032hFdcyaIFPg4KZyiiZZuarSF9LKm16ZtLkRK6do91LkOXpqAC
3	3	lorbe@gmail.com	Leonardo	Orbe	lorbe	\$2a\$10\$Y3EVVlpgtzyONpdjKpotuWgmWKJ6zKKNZhyfHx8Dgp9EYJA2TQK
4	4	jtuibice@gmail.com	Jeremy	Tuitice	jtuibice	\$2a\$10\$q918fZmtTZaAXgx0vh33SO8oikgRxY5NHBPkgY9FLUYEU/lo/kNu
5	5	jrozero@gmail.com	Jenny	Rosero	jrozero	\$2a\$10\$mLphlYZ9e.evD2pCcL4JB.kYxNAB9/bl6fA2BSyyStS1Ze8ZtNyaa
6	6	pllaguno@gmail.com	Paulo	Llaguno	pllaguno	\$2a\$10\$BhtS8LpkBq4rk3.tSi1DPeM4erlqx1em7K47ttQoyXq8vul5zhUG
7	7	smorales@gmail.com	Sarahi	Morales	smorales	\$2a\$10\$G3UN36njSjB/gf4tCbV.dOg0HijgZ5nO/mx:Aaon.kNk9SIIHOvVa
8	8	kmorales@gmail.com	Karina	Morales	kmorales	\$2a\$10\$9X.9hjYw17WAZLtnqfAs.wGO3FkH7FWvKMZHC5.xu5VgVOg5fCzm

- MongoDB: Base de datos para proyectos

Utilizada por la complejidad de los datos almacenados y evitar construir muchas relaciones

### Colección: projects



### Formato de registro

```
{
  "_id": {
    "$numberLong": "1"
  },
  "name": "Proyecto Arquitectura de Software",
  "idOwner": {
    "$numberLong": "1"
  },
  "codeInvitation": "hXIN1f4zeIN",
  "users": [
    {
      "_id": {},
      "role": "ADMIN"
    },
    {
      "_id": {},
      "role": "USER"
    }
  ],
  "tasks": [
    {
      "_id": {},
      "name": "Establecer Bases de Datos",
      "description": "Encontrar la Base de Datos mas \u00f3ptima para el sistema",
      "dateStart": {},
      "dateFinish": {}
    },
    {
      "_id": {},
      "name": "Prueba Tarea",
      "description": "Prueba Tarea",
      "dateStart": {},
      "dateFinish": {}
    }
  ],
  "_class": "com.packet.projects.model.Project"
}
```

## Archivos necesarios para levantar el sistema:

Los archivos necesarios estarán todos implementados en la carpeta:

### Proyecto\_Manager\_Projects

En su interior están las 4 secciones importantes:

- ✓ 1\_RespaldosBDD
- ✓ 2\_APIs\_BackEnd
- ✓ 3\_FrontEnd
- ✓ 4\_Nginx

Y se desplegarán los componentes del sistema en función del número en la carpeta, es decir:

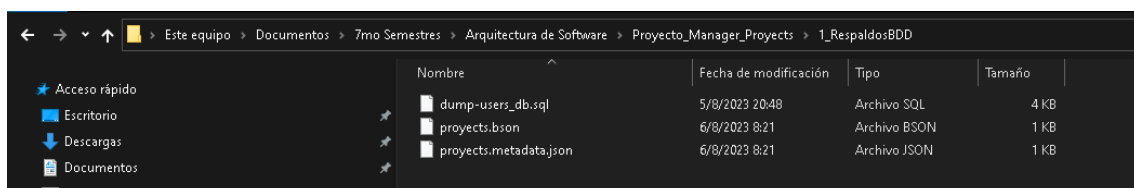
- Primero las Bases de Datos -> RespaldosBDD (Capa de Datos)
- Segundo las APIs pertenecientes al Back End - > APIs\_BackEnd (Capa de persistencia y lógica de negocio)
- Tercero el servidor Nginx -> Nginx(Balanceador de Carga)
- Cuarto el Front\_End -> FrontEnd(Capa de presentación)

## PASOS PARA EL LEVANTAMIENTO DE LAS BASES DE DATOS

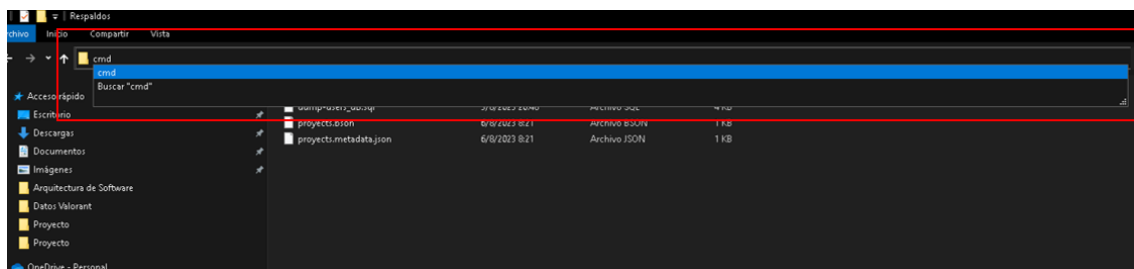
**\*Es indispensable el levantamiento de las bases de datos en primer lugar, para evitar conflictos con las APIs CRUD. \***

### Levantamiento de BDD MySql

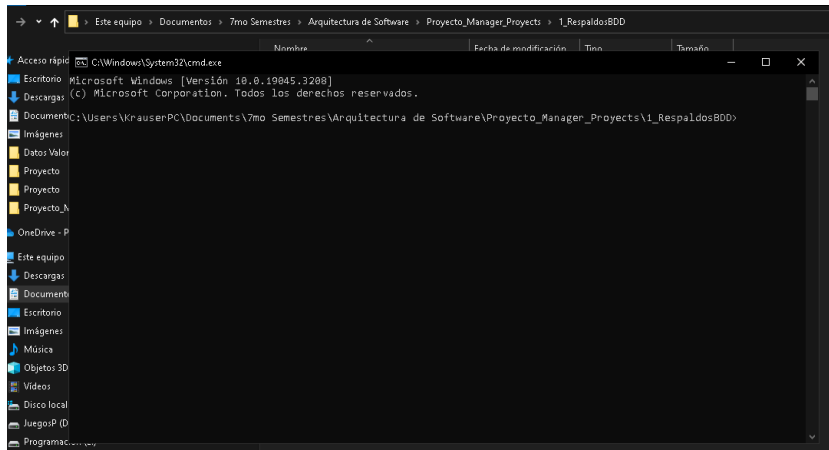
Entramos en la carpeta de 1\_RespaldosBDD



Y en la ruta del gestor de archivos colocamos la palabra cmd de la siguiente manera y presionamos ENTER:

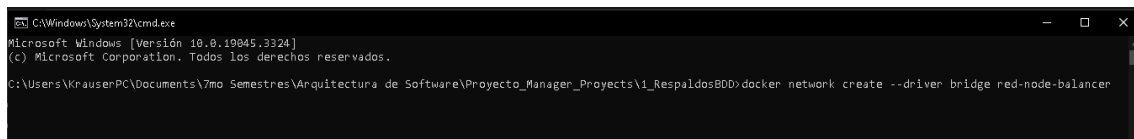


Así se abrirá una consola en dicha ruta (con el objetivo de ahorrar la búsqueda de la ruta al usuario):



**\*\*\*Ahora vamos a crear la red de contenedores donde funcionará nuestro sistema, con el siguiente comando: \*\*\***

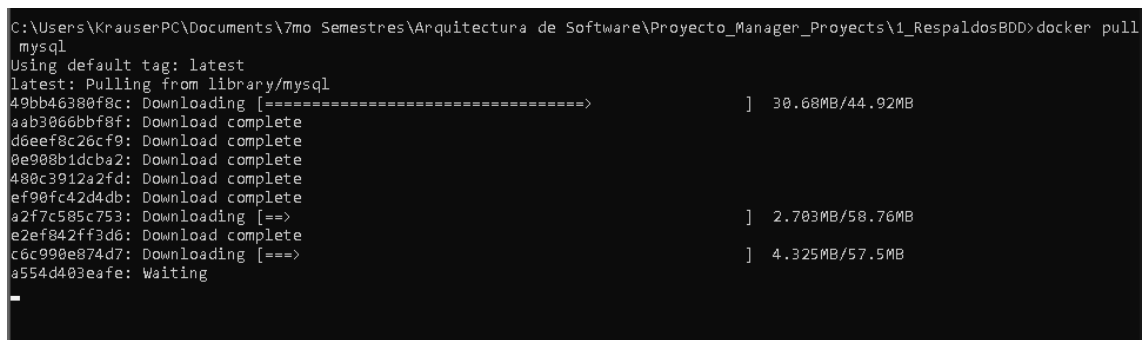
**docker network create --driver bridge red-node-balancer**



**Y continuamos con el levantamiento de la bdd MySQL**

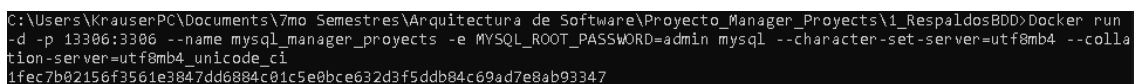
Ejecutamos los siguientes comandos en el CMD abierto anteriormente, para descargar una imagen de MySQL y después crear una nueva instancia del contenedor MySQL:

**docker pull mysql**



Ahora creamos una instancia de la base de datos con el comando:

**docker run -d -p 13306:3306 --network=red-node-balancer --name mysql\_manager\_projects -e MYSQL\_ROOT\_PASSWORD=admin mysql --character-set-server=utf8mb4 --collation-server=utf8mb4\_unicode\_ci**



Nota: estamos usando el usuario root y contraseña : admin con esta opción  
:MYSQL\_ROOT\_PASSWORD=admin

Ahora procedemos a crear la BDD donde volcaremos el archivo de respaldo con el siguiente comando:

**docker exec -it mysql\_manager\_proyectos mysql -uroot -padmin -e "CREATE DATABASE users\_db;"**

```
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Proyectos\1_RespalDOSBDD>docker exec -it mysql_manager_proyectos mysql -uroot -padmin -e "CREATE DATABASE users_db"
mysql: [Warning] Using a password on the command line interface can be insecure.
```

**\*Hasta este paso la base de datos ya estaría activa y lista para conectarse a la API CRUD (pero sin registros) y para volcar los datos hacemos lo siguientes pasos: \***

Copiamos el archivo con el respaldo de la BDD original en el contenedor, con el siguiente comando:

**docker cp dump-users\_db.sql mysql\_manager\_proyectos:/dump.sql**

```
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Proyectos\1_RespalDOSBDD>docker cp dump-users_db.sql mysql_manager_proyectos:/dump.sql
Successfully copied 5.63kB to mysql_manager_proyectos:/dump.sql
```

Finalmente hacemos el volcado de los datos en la nueva BDD con los siguientes comandos:

Entramos al contenedor con:

**docker exec -it mysql\_manager\_proyectos bash**

```
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Proyectos\1_RespalDOSBDD>docker exec -it mysql_manager_proyectos bash
bash-4.4#
```

y escribimos el siguiente comando para realizar el volcado:

**bash-4.4# mysql -uroot -padmin users\_db < /dump.sql**

```
bash-4.4# mysql -uroot -padmin users_db < /dump.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
bash-4.4#
```

Para salir del contenedor usamos el comando exit.

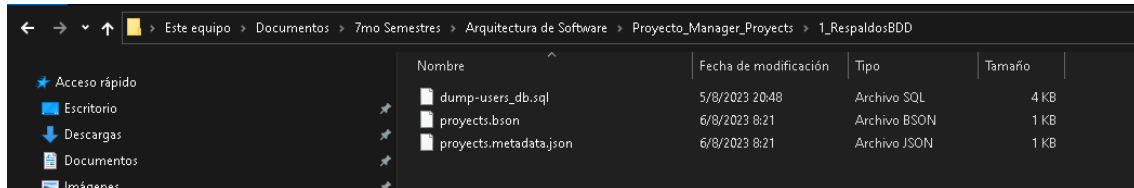
```
bash-4.4# exit
exit
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Proyectos\1_RespalDOSBDD>
```

**\*\*Listo, nuestra BDD MySql en un contenedor esta levantada, y lista para ser usada. \*\***

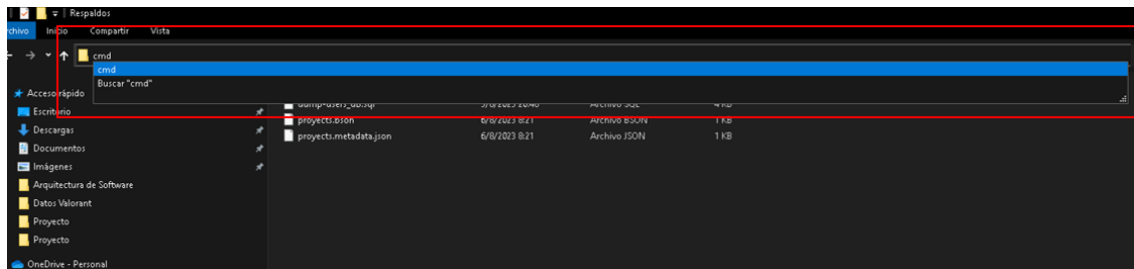
## **2.- Levantamiento de BDD Mongo**

Seguimos los mismos pasos iniciales de la instalación de MySQL

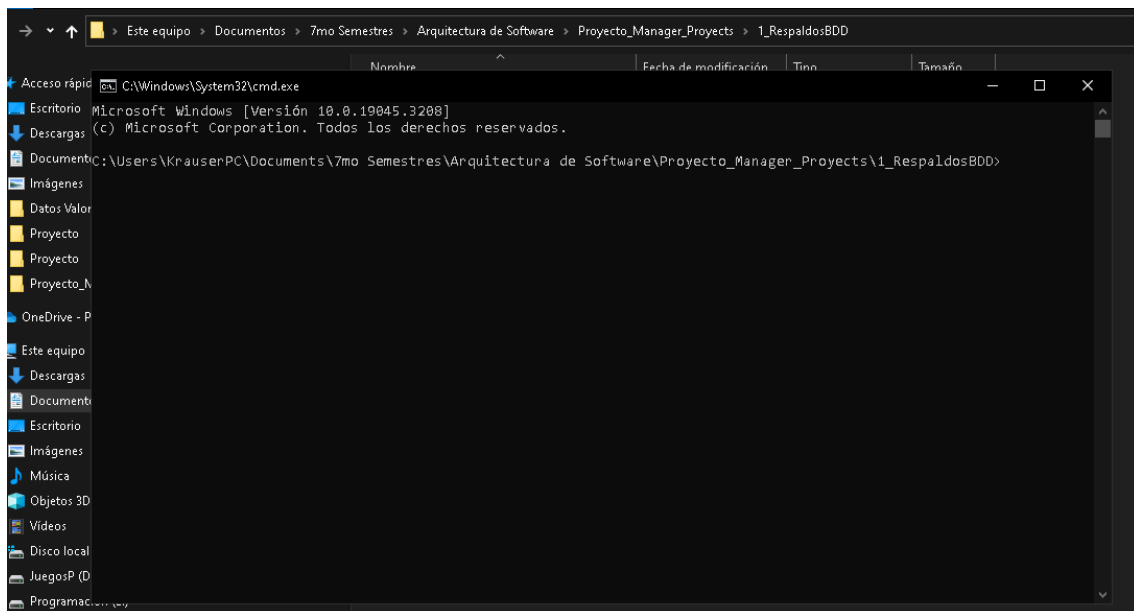
Entramos en la carpeta de 1\_RespaldoBDD



Y en la ruta del gestor de archivos colocamos la palabra cmd de la siguiente manera y presionamos ENTER:

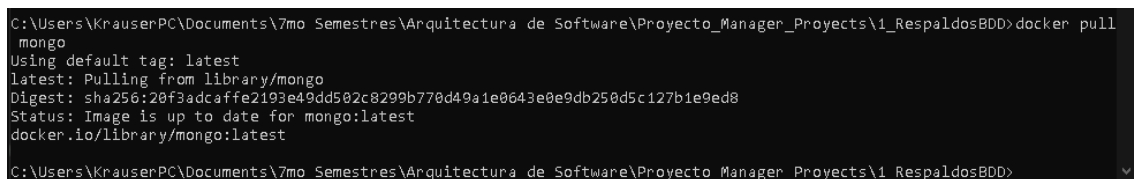


Así se abrirá una consola en dicha ruta (con el objetivo de ahorrar la búsqueda de la ruta al usuario):



Ahora, ejecutamos los siguientes comandos en el CMD abierto anteriormente, para descargar una imagen de Mongo.

**docker pull mongo**



Ejecutamos el siguiente comando para crear una instancia de Bdd mongo, para el Sistema:

**docker run -d --name mongodb\_manager\_projects -p 37017:27017 --network=red-node-balancer mongo**

```
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\1_RespalDOSBDD>docker run -d --name mongodb_manager_projects -p 37017:27017 mongo
0a3320225579a39275e14561604ef91f7e4a571ba35efc09ac0
```

Ahora copiamos los archivos con los respaldos de la BDD original al contenedor con el siguiente comando:

**docker cp projects.bson mongodb\_manager\_projects:/projects.bson**

```
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\1_RespalDOSBDD>docker cp projects.bson mongodb_manager_projects:/projects.bson
Successfully copied 2.50kB to mongodb_manager_projects:/projects.bson
```

Nota: es importante estar en la ruta de los respaldos con el CMD que se mostró al inicio debido a que, si no es así, el usuario debería colocar la ruta del archivo .SQL de forma manual.

El siguiente paso es ingresar al contenedor con el siguiente comando:

**docker exec -it mongodb\_manager\_projects bash**

```
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\1_RespalDOSBDD>docker exec -it mongodb_manager_projects bash
root@0a3320225579a:/#
```

Finalmente subimos el respaldo a la BDD escribiendo el siguiente comando:

**mongorestore --db projects\_db /projects.bson**

```
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\1_RespalDOSBDD>docker exec -it mongodb_manager_projects bash
root@0a3320225579a:/# mongorestore --db projects_db /projects.bson
2023-08-06T16:36:46.358+0000 checking for collection data in /projects.bson
2023-08-06T16:36:46.381+0000 restoring projects_db.projects from /projects.bson
2023-08-06T16:36:46.428+0000 finished restoring projects_db.projects (4 documents, 0 failures)
2023-08-06T16:36:46.428+0000 4 document(s) restored successfully. 0 document(s) failed to restore.
```

salimos del contenedor con el comando **exit**:

```
root@0a3320225579a:/# exit
exit
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\1_RespalDOSBDD>
```

**\*\*Listo, nuestra BDD Mongo en un contenedor esta levantada, y lista para ser usada. \*\***

## LEVANTAMIENTO DE LAS APIS

### REQUISITOS INDISPENSABLES:

Se requiere el compilador de java JDK 11

Link JDK 11: <https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>

Es necesario usar el JDK 11 debido a que las APIs están construidas para ser compiladas con dicha versión, otra versión superior generaría problemas, mientras que una versión menor como JDK 8 podría funcionar.



Se requiere tener instalado algún IDE que soporte Java, se sugiere NETBEANS versión 17 de preferencia

Link NetBeans: <https://netbeans.apache.org/download/nb17/>

Tutorial para instalar el JDK e IDE (video de 3 minutos):

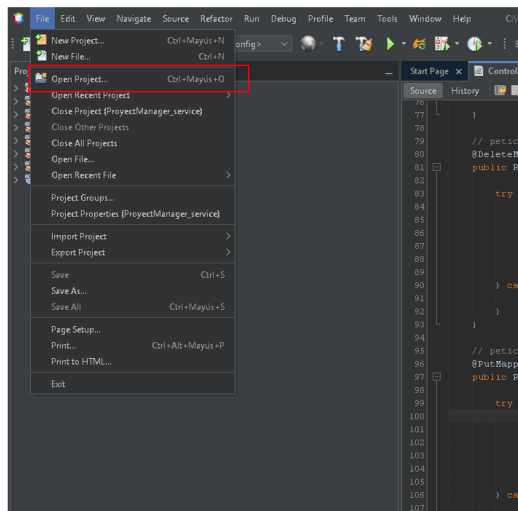
Se debe cambiar las versiones del tutorial por las sugeridas en este manual

Link: <https://www.youtube.com/watch?v=njLXdLjVJfw>

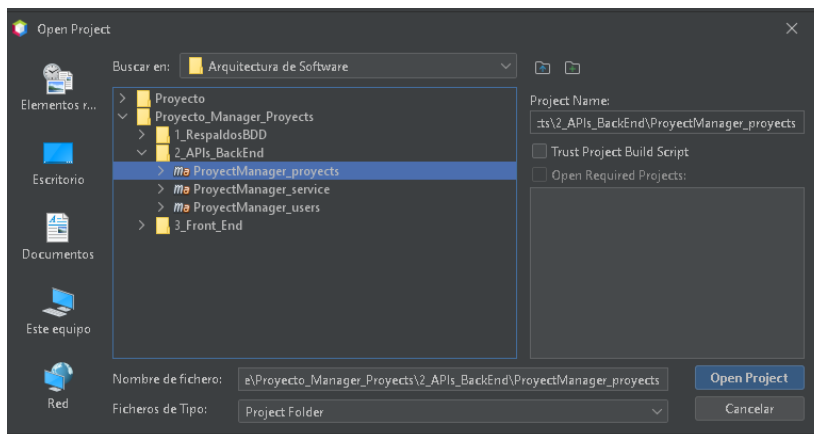
**UNA VEZ CUMPLIDOS LOS REQUISITOS:**

En el IDE ejecutándose cargamos cada uno de los proyectos de la siguiente manera:

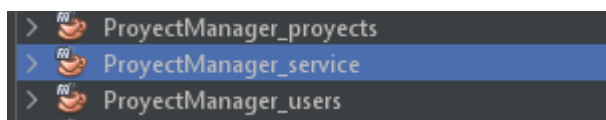
Damos clic en la opción File y Open Project:



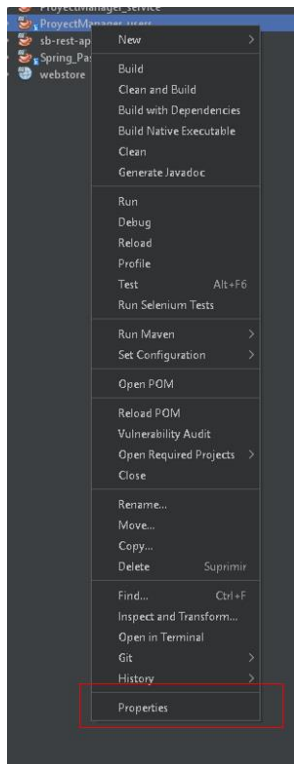
Y buscamos los archivos de las API en la carpeta 2\_APIs\_Back\_end



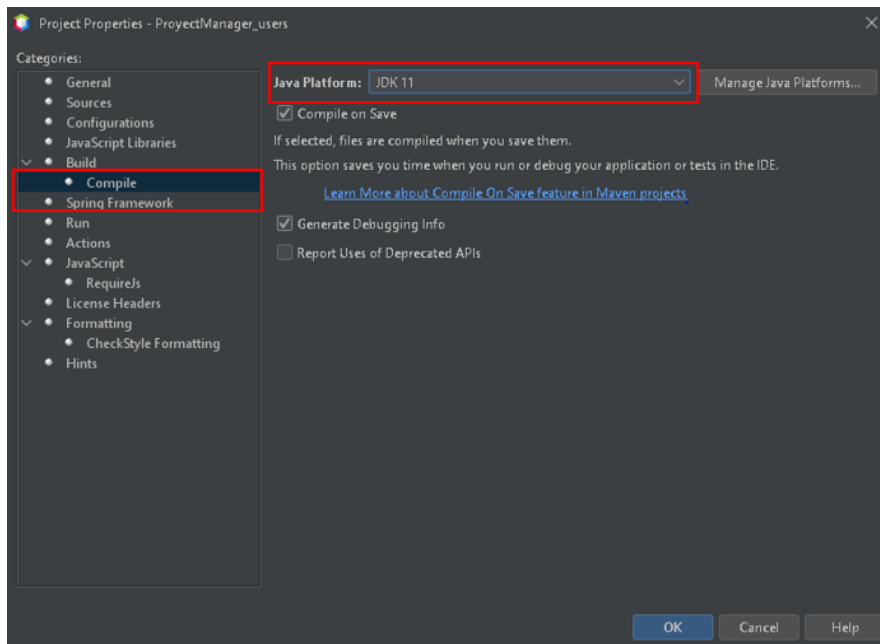
Y abrimos cada uno de ellos con la opción Open Project



Es necesario que se revise que los proyectos sean compilados con el JDK 11 de la siguiente manera :

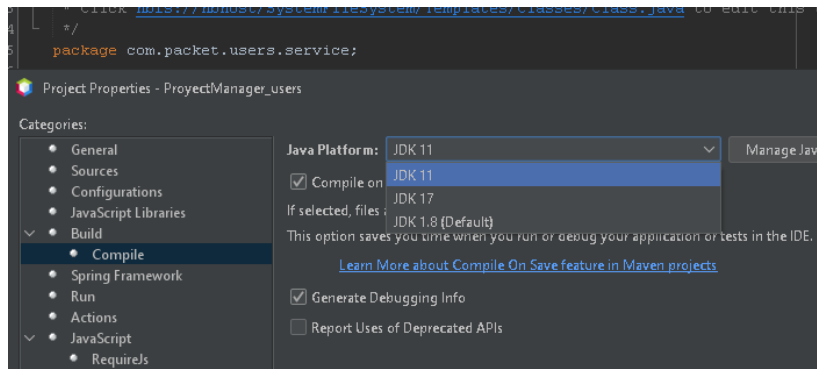


El siguiente Paso es revisar en la sección de Build, y la subsección Compile, para comprobar que Java Platform JDK 11



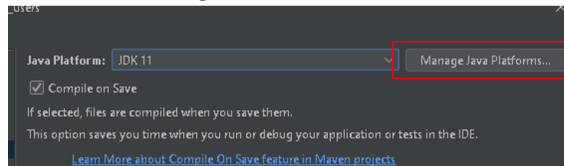
#### EN CASO DE NO TENERLO DISPONIBLE

Primero Verificar si existe en las opciones(en la misma ventana)

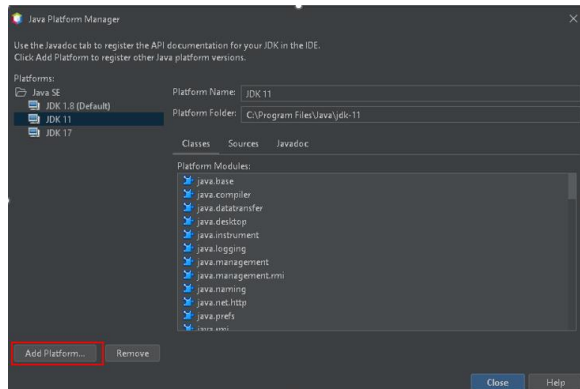


Si no existe seguir los siguientes pasos :

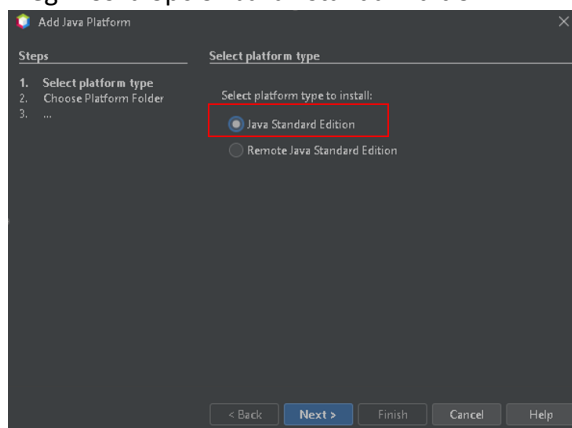
### Presionar Manage Java Platforms



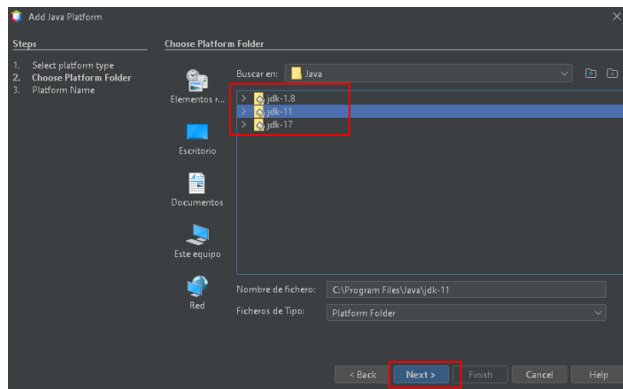
### Seguido de la opción Add Platform



### Elegimos la opción Java Estándar Edition:



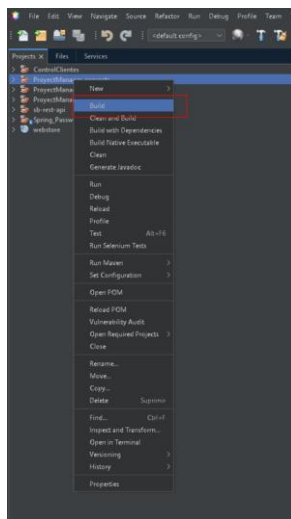
Y buscamos la ruta donde instalamos el JDK 11 al principio y presionamos Next, seguido de Finalizar:



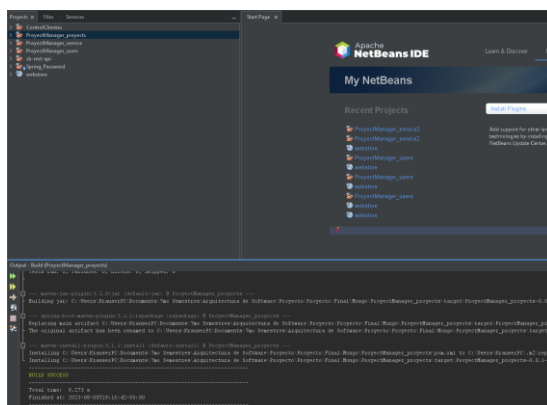
Una vez cargados los proyectos, realizamos el siguiente proceso para cada una de la APIs

### ProjectManager\_projects

Desde el Netbeans, presionamos la opción Build sobre el proyecto ProjectManager\_projects para construir el archivo JAR, con el cual podremos dockerizar la aplicación.

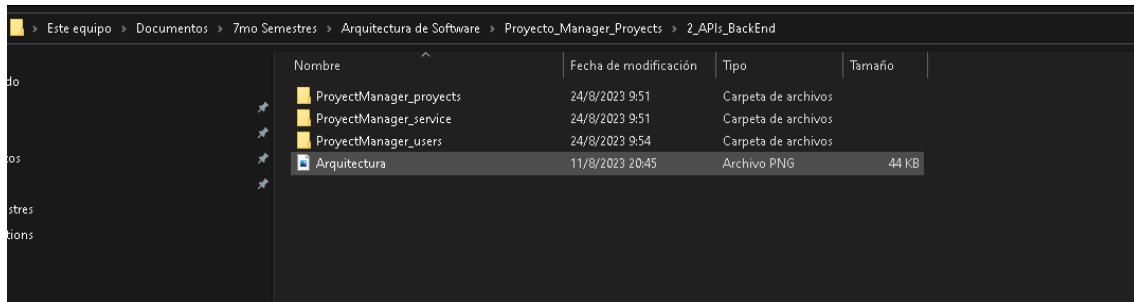


Aparecerá esto en el output, lo que significa que el proceso se hizo correctamente

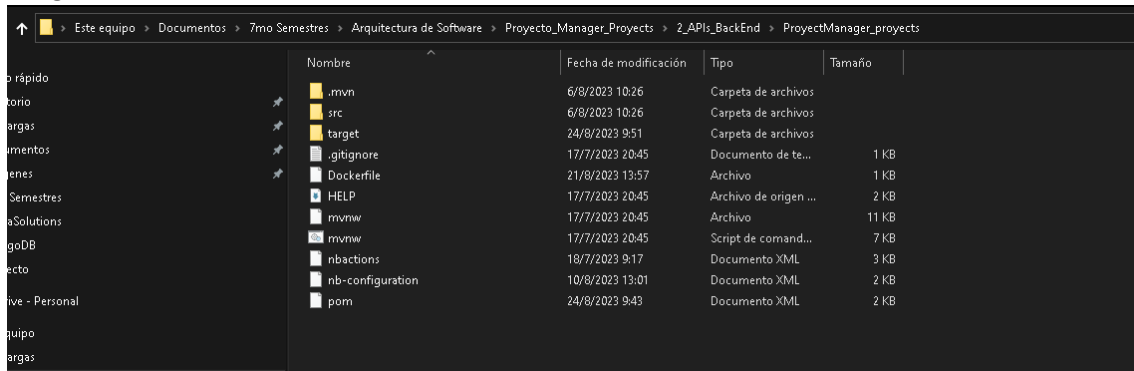


Ahora procedemos a crear una imagen de nuestra aplicación:

Nos dirigimos a la carpeta del proyecto ProjectManager\_projects, que se ubica en la carpeta raíz del proyecto, dentro de la carpeta 2\_APIs\_BackEnd:

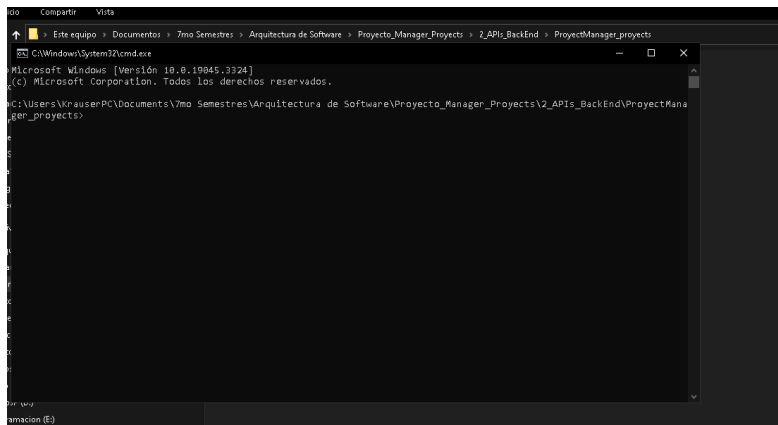


Entonces ingresamos a su interior:



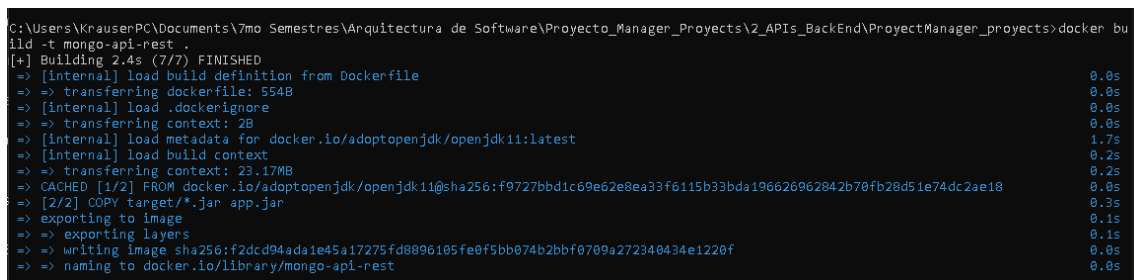
- Dockerfile: este es el archivo el cual va a generar una imagen de nuestra aplicación que posteriormente será ejecutado en una instancia.

Abrimos un CMD en la ruta actual, reemplazando la ruta por la palabra cmd:



Ahora colocamos el siguiente comando para crear la imagen a partir del archivo Dockerfile:

**docker build -t mongo-api-rest .**



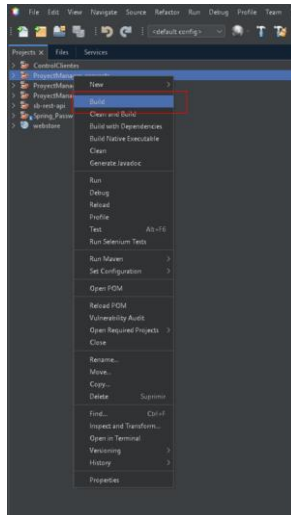
Ahora creamos una instancia de esta imagen con el siguiente comando:

**docker run -d --name pm-projects-crud -p 8082:8082 --network=red-node-balancer mongo-api-rest**

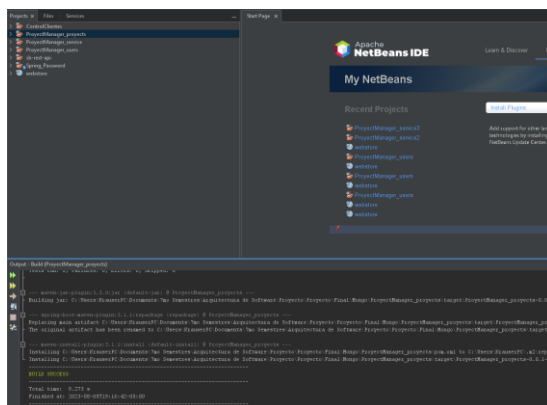
```
=> => naming to docker.io/library/mongo-api-rest 0.0s
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\2_APIs_BackEnd\ProjectManager_projects> docker run
n -d --name pm-projects-crud -p 8082:8082 --network=red-node-balancer mongo-api-rest
84b935a6eb01599af037305664d989d9c2fef82213d4cb1c3809bd4bf354986d
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\2_APIs_BackEnd\ProjectManager_projects>
```

## ManagerProjects\_service

Desde el Netbeans, presionamos la opción Build sobre el proyecto ProjectManager\_service para construir el archivo JAR, con el cual podremos dockerizar la aplicación.

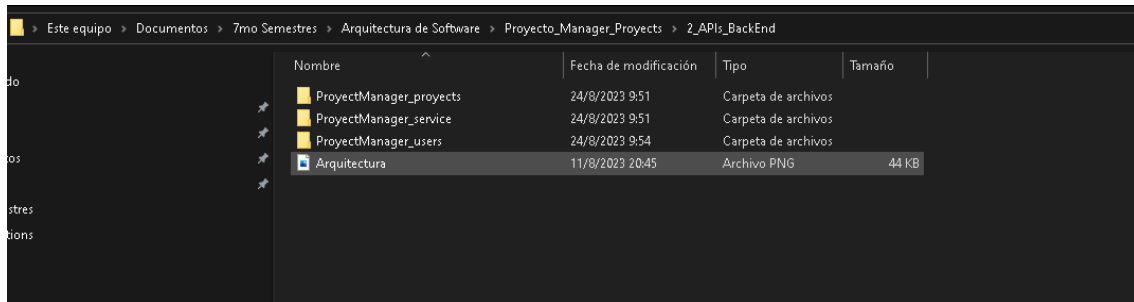


Aparecerá esto en el output, lo que significa que el proceso se hizo correctamente

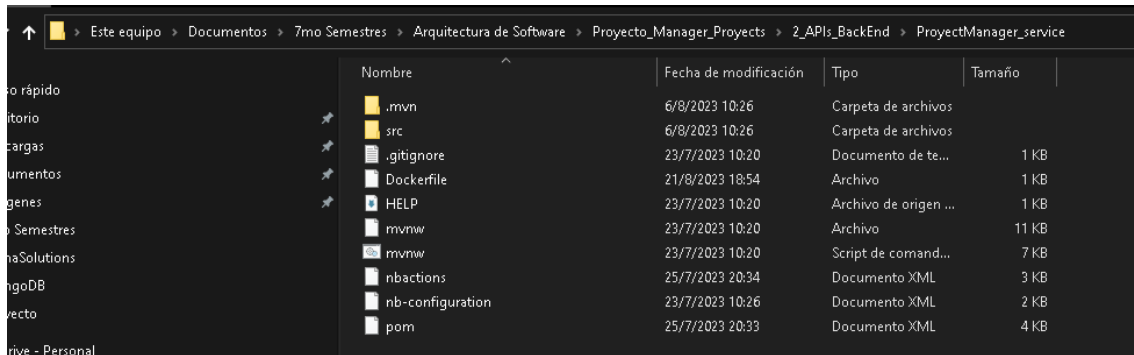


Ahora procedemos a crear una imagen de nuestra aplicación:

Nos dirigimos a la carpeta del proyecto ProjectManager\_service, que se ubica en la carpeta raíz del proyecto, dentro de la carpeta 2\_APIs\_BackEnd:

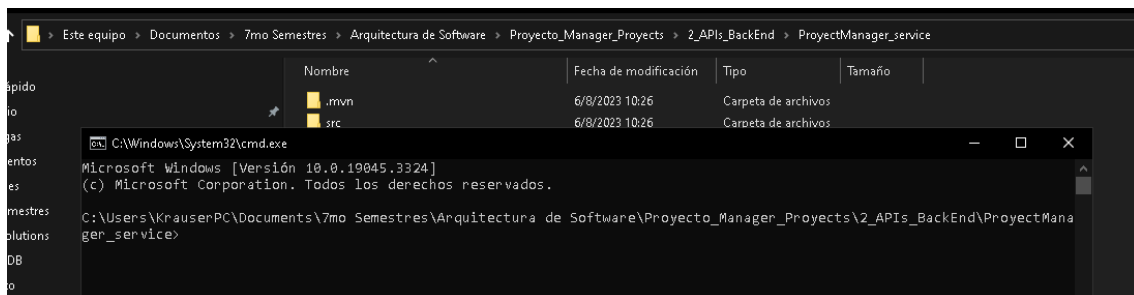


E ingresamos a su interior:



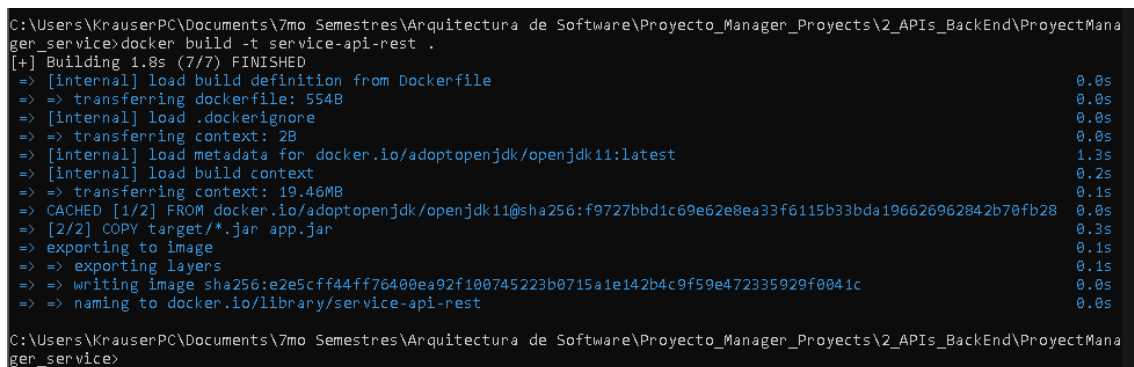
- Dockerfile: este es el archivo el cual va a generar una imagen de nuestra aplicación que posteriormente será ejecutado en una instancia.

Abrimos un CMD en la ruta actual, reemplazando la ruta por la palabra cmd:



Ahora colocamos el siguiente comando para crear la imagen a partir del archivo Dockerfile:

**docker build -t service-api-rest .**



Ahora creamos tres instancias de esta imagen con los siguientes comandos:

**docker run -d --name pm-service-1 -p 8083:8083 --network=red-node-balancer service-api-rest**

**docker run -d --name pm-service-2 -p 8084:8083 --network=red-node-balancer service-api-rest**

**docker run -d --name pm-service-3 -p 8085:8083 --network=red-node-balancer service-api-rest**

```
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\2_APIs_BackEnd\ProjectManager_service>docker run -d --name pm-service-1 -p 8083:8083 --network=red-node-balancer service-api-rest
62fc793d6c8c7a6883e867b5149cc85343e9aac980b04c447e564ed9226abed3

C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\2_APIs_BackEnd\ProjectManager_service>docker run -d --name pm-service-2 -p 8084:8083 --network=red-node-balancer service-api-rest
89fe2a56897ffc1a535ff548516ba0c24d585ae4e30af9a26a7a6d6f49ad21d7

C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\2_APIs_BackEnd\ProjectManager_service>docker run -d --name pm-service-3 -p 8085:8083 --network=red-node-balancer service-api-rest
870b877f345a8cb80aad971f9f3e8c48a8a8e723cfd7e7be13a43361e8359

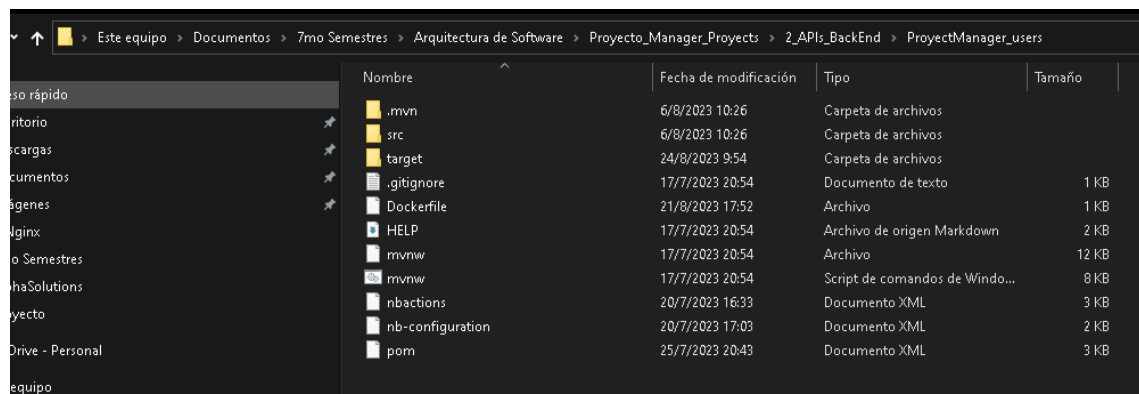
C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\2_APIs_BackEnd\ProjectManager_service>
```

Se crearon 3 instancia de la misma imagen(aplicación) con el objetivo de conectarlas un balanceador de carga que se encargara de gestionarlas.

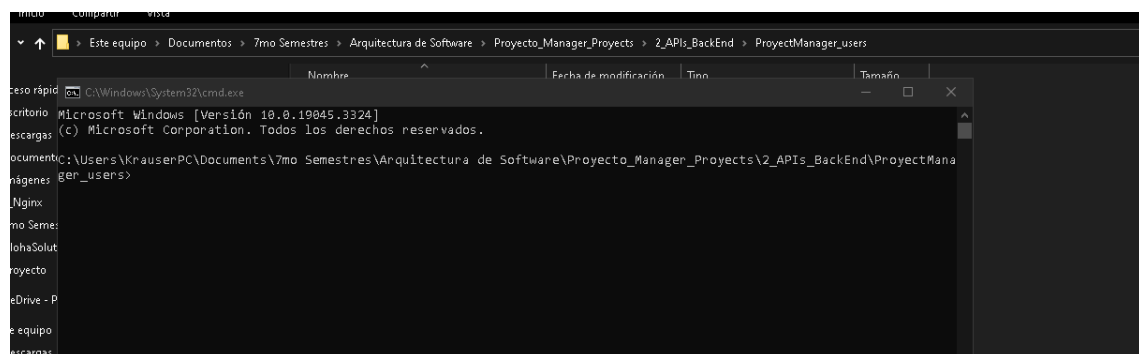
### ManagerProjects\_users

Para el proyecto ManagerProjects\_users, el caso es especial por lo que se debe hacer lo siguiente:

Primero entramos a la ruta donde tenemos el proyecto, en este caso dentro de la carpeta raíz del proyecto, seguido entramos a la carpeta 2\_APIs\_BackEnd y por último a la carpeta ProjectManager\_users:



Aquí ejecutamos en la ruta le comando CMD, para obtener una nueva consola en dicha ruta:



Y ejecutamos el siguiente comando:-> Para compilar el proyecto y obtener el archivo JAR.

**mvnw clean package -DskipTests**



Ahora ejecutamos el siguiente comando para crear la imagen para levantar un contenedor con la aplicación API REST:

```

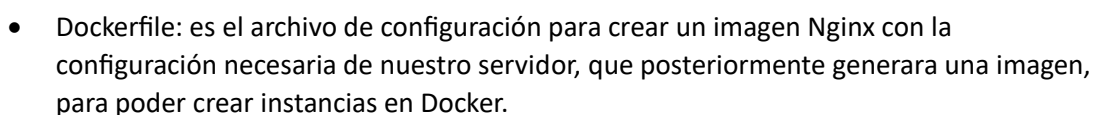
C:\Users\robert\Documents\IT\Semester\Ver\collectura de Software\Proiecte Manager_Proiects\2_APIs_BackEnd\ProjectManager_users\docker build -f mysql-apl-rest .
[+] Building 2.4s (777) FINISHED
[+] [Internal] load build definition from Dockerfile
[+] ==> transferring dockerfile: 554B
[+] [Internal] load .dockerignore
[+] ==> transferring context: 2B
[+] [Internal] load metadata for docker.io/dotopenjdk/openjdk:11-latest
[+] [Internal] load build context
[+] ==> transferring context: 42.37MB
[+] CACHED [1/2] FROM docker.io/dotopenjdk/openjdk:11ln@256:f9727bd0c9662e8ea33f15b33bda196626962842076fb8d5e7d7dc2ae18
[2/2] COPY target/*.jar app.jar
[+] ==> exporting to image
[+] ==> exporting layers
[+] ==> writing image sha256:274c241e6f60788604c18093f32acdd3f8125149fc0e4d627f4c84e191d9
[+] ==> naming to docker.io/library/mysql-apl-rest
[+] ==> image has been pushed

```

```
docker run -d --name pm-users-crud -p 8081:8081 --network=red-node-balancer mysql-api-rest
```

Y listo estará levantado el servicio en Dockers

Para crear una instancia de nuestro servidor Nginx para el balance de carga de nuestro sistema, primero accedemos a la carpeta 4 Nginx:



- Nginx.conf: es el archivo de configuración que contiene la información necesaria para el funcionamiento del servidor como: los servidores a los que enviara los request, el algoritmo de balanceo a utilizar.

Aquí hacemos el mismo proceso para abrir una consola del CMD, es decir cambiamos la ruta por cmd



Y ejecutamos el siguiente comando:

**docker build -t node-balancer-nginx .**

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3324]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\4_Nginx>docker build -t node-balancer-nginx .
[+] Building 3.9s (7/7) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 150B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/nginx:latest                  3.8s
=> [internal] load build context                                                0.0s
=> => transferring context: 712B                                                0.0s
=> CACHED [1/2] FROM docker.io/library/nginx:latest@sha256:104c7c5c54f2685f0f46f3be607ce60da7085da3eaa5ad22d3d9f 0.0s
=> [2/2] COPY nginx.conf.txt /etc/nginx/nginx.conf                             0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:fdeb01eeb9353c9c2d2f5c4b94f97c5a122a6bdf9d0521baeeb9872d5c864a40 0.0s
=> => naming to docker.io/library/node-balancer-nginx                          0.0s

C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\4_Nginx>

```

Ahora creamos una instancia del servidor con el comando:

**docker run -d --name node-balancer -p 8080:80 --network=red-node-balancer node-balancer-nginx**

```

=> => naming to docker.io/library/node-balancer-nginx                          0.0s

C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\4_Nginx>docker run -d --name node-balancer -p 8080:80 --network=red-node-balancer node-balancer-nginx
dda04c3c27341862d85f6d9f2556606673c8e459897be73f1e10c2221477f6bc

C:\Users\KrauserPC\Documents\7mo Semestres\Arquitectura de Software\Proyecto_Manager_Projects\4_Nginx>

```

Y listo nuestro servicio estará en funcionamiento.

Una vez levantados los servicios ya pueden ser consumidos, al ser independientes en caso de fallar alguno o caerse no afectara al funcionamiento de todo el sistema, gracias a su bajo acoplamiento

### 3.- LEVANTAMIENTO DEL FRONT END

#### REQUISITOS INDISPENSABLES:

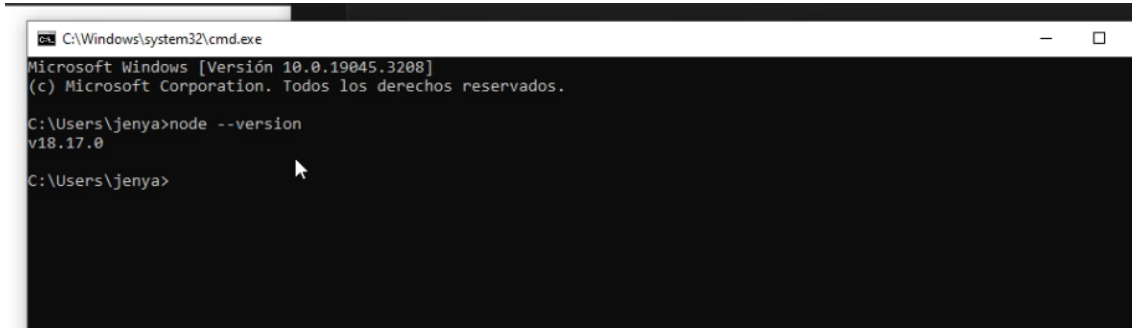
Se requiere tener **instalado Node.JS** (versión 18.17.0)

<https://nodejs.org/es>

Para su instalación ejecutar el archivo descargado y dar siguiente a todas las opciones hasta instalar.

### Comprobar que ya se instaló Node.js

Para comprobar que ya tengo instalado Node.js abro el cmd o terminal y coloco **node --version**



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3208]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jenya>node --version
v18.17.0

C:\Users\jenya>
```

Se comprueba su instalación con la salida de la versión en la consola, como en la imagen.

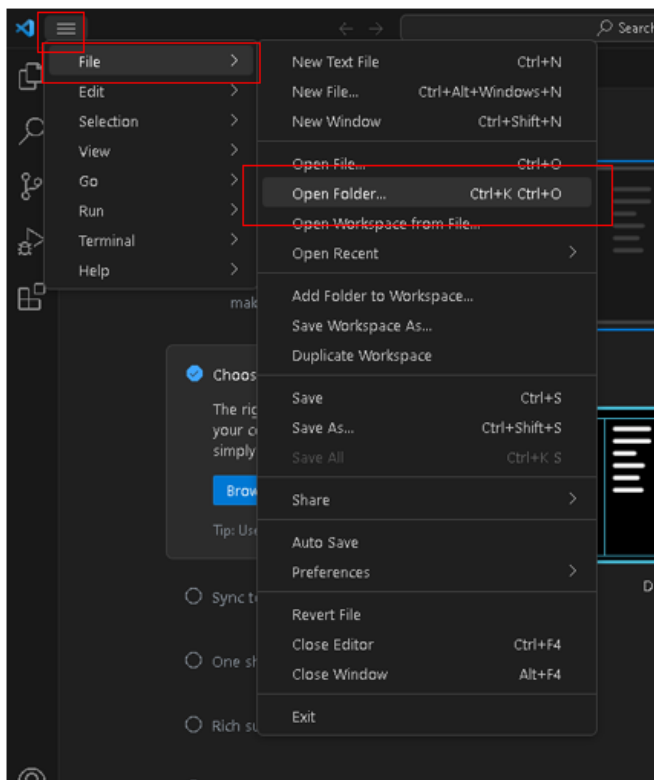
- **Instalar el editor de código Visual Studio Code**

<https://code.visualstudio.com/>

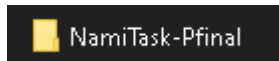
Para su instalación ejecutar el archivo descargado y dar siguiente a todas las opciones hasta instalar.

- **Abrir el programa**

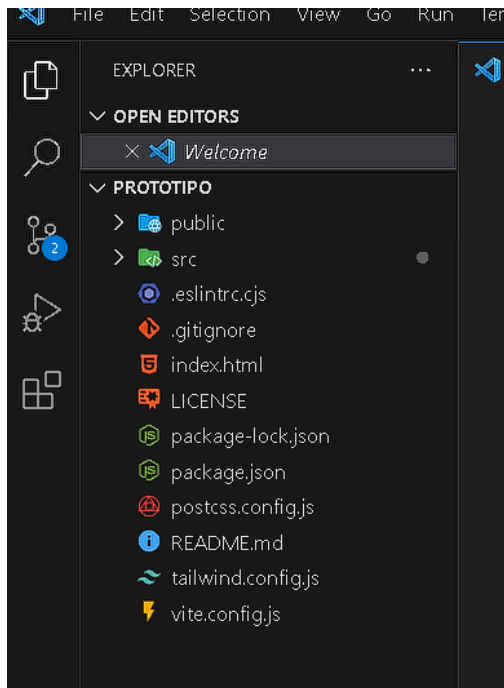
abrimos Visual Studio Code y en la parte superior presionamos las 3 líneas, seguido de la opción File, seguido de la opción Open Folder



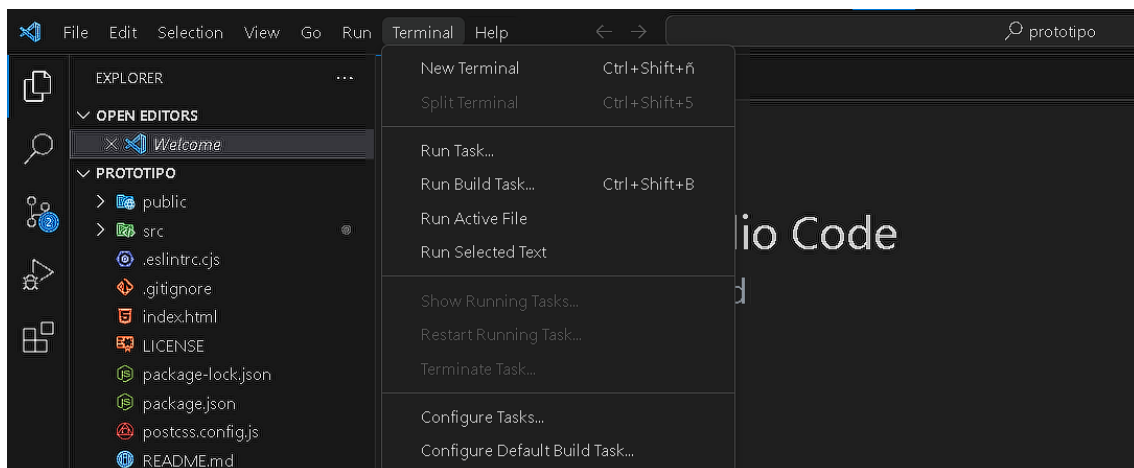
Seleccionamos la carpeta llamada NamiTask-Pfinal ubicada dentro de la carpeta 3\_Front\_End , que está al interior del proyecto



- Se va a abrir el proyecto de esta manera:



Vamos a la parte de terminal damos clic en new terminal:

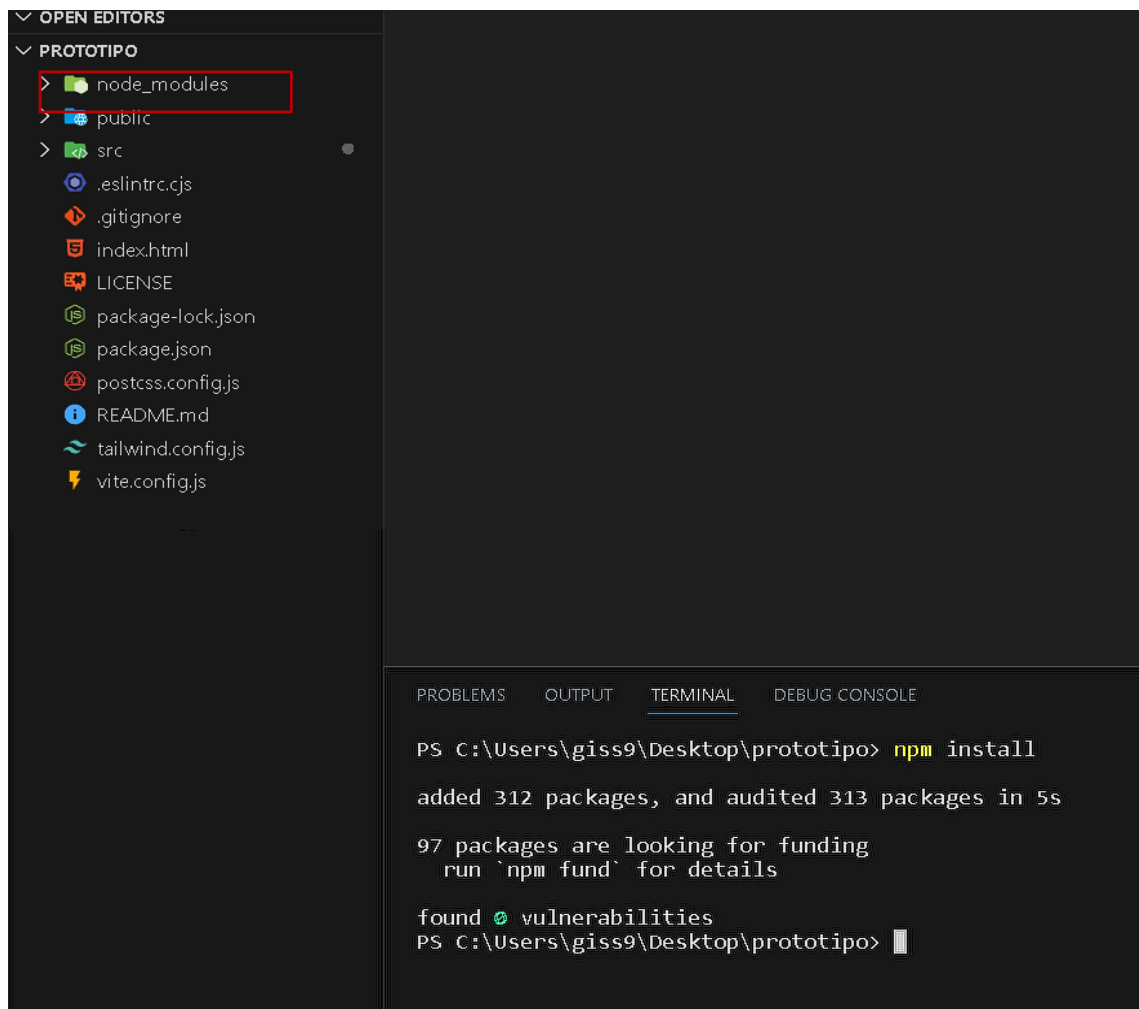


Ponemos el comando:

**npm install**

```
PS C:\Users\giss9\Desktop\prototipo> npm install
```

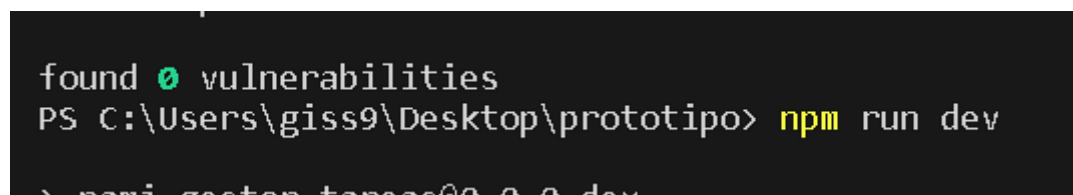
se va a crear una carpeta llamada node-modules:



Es necesario tener ese archivo instalado ya que ahí van a estar todas las dependencias que necesita el programa. Si existe el caso de que sale error al poner ese comando, se cierra el VisualStudioCode y se vuelven a abrir ya que en ocasiones no se actualiza que ya tenemos instalado el node.JS

Ahora, ingresamos el siguiente comando para ejecutar nuestro programa.:

**npm run dev**



Y el terminal nos indicara el siguiente mensaje

```
VITE v4.4.4 ready in 583 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h to show help
```

Damos Ctrl + click en enlace azul que nos muestra y finalmente nos abre la siguiente página:

Nami Task

**Iniciar sesión**

Nombre de usuario:

Contraseña:

Iniciar sesión

**Registrarse**

Nombre:

Apellido:

Nickname:

Correo electrónico:

Contraseña:

Registrarse

### COMO LUCE EL SISTEMA AL FINAL DEL PROCESO DE INSTALACIÓN:

Pues al realizar instancias de cada aplicación, se puede observar los contenedores desplegados desde la interfaz de Docker desktop:

<input type="checkbox"/>	Name	Image	Status	Port(s)	Last started	Actions
<input type="checkbox"/>	<a href="#">bdd-ares-pet</a> c0036a52c653	<a href="#">db_arespetshc</a>	Exited	3307:3306	3 hours ago	
<input type="checkbox"/>	<a href="#">mongodb_manager_proyectos</a> 0a320225579a	<a href="#">mongo</a>	Running	<a href="#">37017:27017</a>	2 hours ago	
<input type="checkbox"/>	<a href="#">mysql_manager_proyectos</a> 1fec7b02156f	<a href="#">mysql</a>	Running	<a href="#">13306:3306</a>	1 hour ago	
<input type="checkbox"/>	<a href="#">pm-users-crud</a> d14c322887b9	<a href="#">mysql-api-res1</a>	Running	<a href="#">8081:8081</a>	1 hour ago	
<input type="checkbox"/>	<a href="#">node-balancer</a> dda04c3c2734	<a href="#">node-balancer</a>	Running	<a href="#">8080:80</a>	1 hour ago	
<input type="checkbox"/>	<a href="#">pm-proyectos-crud</a> 84b935a6eb01	<a href="#">mongo-api-res</a>	Running	<a href="#">8082:8082</a>	2 minutes ago	
<input type="checkbox"/>	<a href="#">pm-service-1</a> 62fc793d6cbc	<a href="#">service-api-res</a>	Running	<a href="#">8083:8083</a>	2 minutes ago	
<input type="checkbox"/>	<a href="#">pm-service-2</a> 89fe2a56897f	<a href="#">service-api-res</a>	Running	<a href="#">8084:8083</a>	2 minutes ago	
<input type="checkbox"/>	<a href="#">pm-service-3</a> 87bb077f345a	<a href="#">service-api-res</a>	Running	<a href="#">8085:8083</a>	2 minutes ago	

## FUNCIONAMIENTO DEL SISTEMA:

Usuarios para pruebas:

Se puede entrar por nickname o email

User: pllaguno

Email: pllaguno@gmail.com

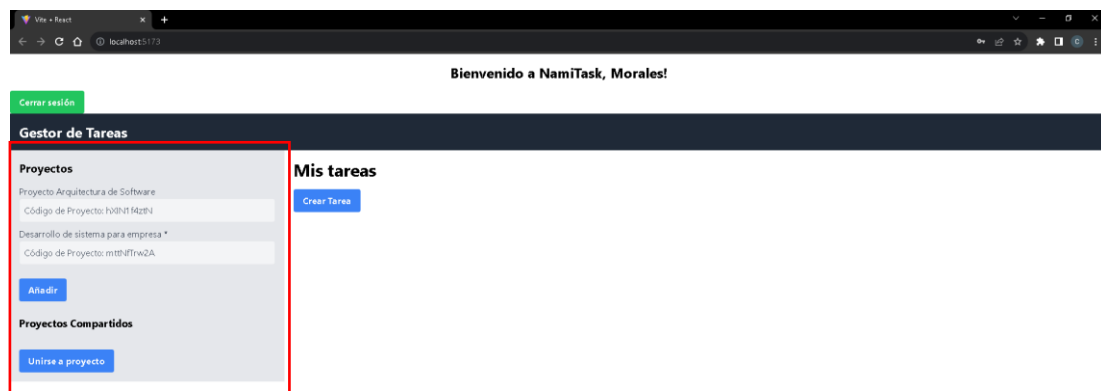
Contraseña: admin

User: cmorales

Email: cmorales@gmail.com

Contraseña: admin

- Con las credenciales accedemos al Sistema :



Al lado izquierdo podremos ver los Proyectos que tenemos.

- Si se desea crear un proyecto se presiona le botón añadir:

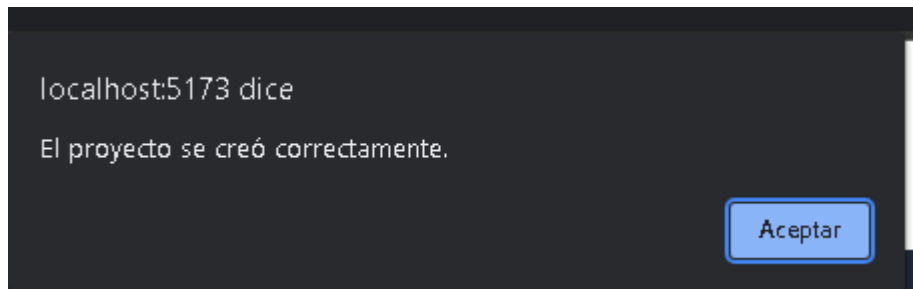
### Proyectos Compartidos

Unirse a proyecto

Nombre del proyecto

GuardarCancelar

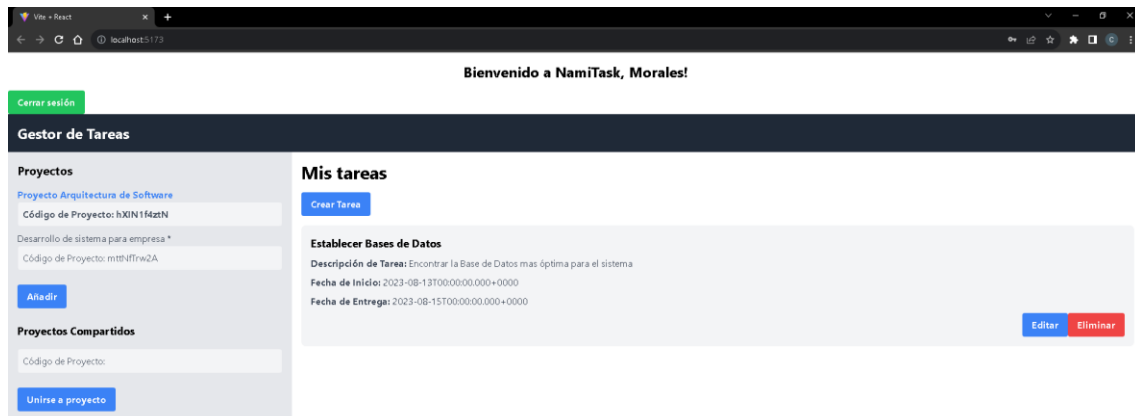
Es necesario que no tenga el mismo nombre de alguno que ya se ha creado, y nos lanzará el siguiente mensaje:



Esto en Caso de cumplir las validaciones, como que no se repita el nombre de alguno proyecto que el usuario en línea haya creado.

(Es necesario refrescar la página con el botón del navegador o el f5, ya que aún no se ha incorporado la opción de autorrefresco, si no, no se verán los cambios )

- Para revisar las tareas o actividades de un proyecto, se hace clic sobre cualquiera de los ya existentes:



Aquí podremos agregar tareas o actividades, presionando el botón crear Tarea:

A form titled "Mis tareas" for creating a new task. It includes a "Crear Tarea" button at the top. The form has four input fields: "Nombre de Tarea:" with the placeholder "Prueba Tarea", "Descripción de Tarea:" with the placeholder "Prueba Tarea", "Fecha de Inicio:" with the date "15/08/2023", and "Fecha de Entrega:" with the date "23/08/2023". At the bottom right, there are two buttons: "Crear Tarea" (blue) and "Cancelar" (red).



## Tarea Creada

**Nombre de Tarea:** Prueba Tarea

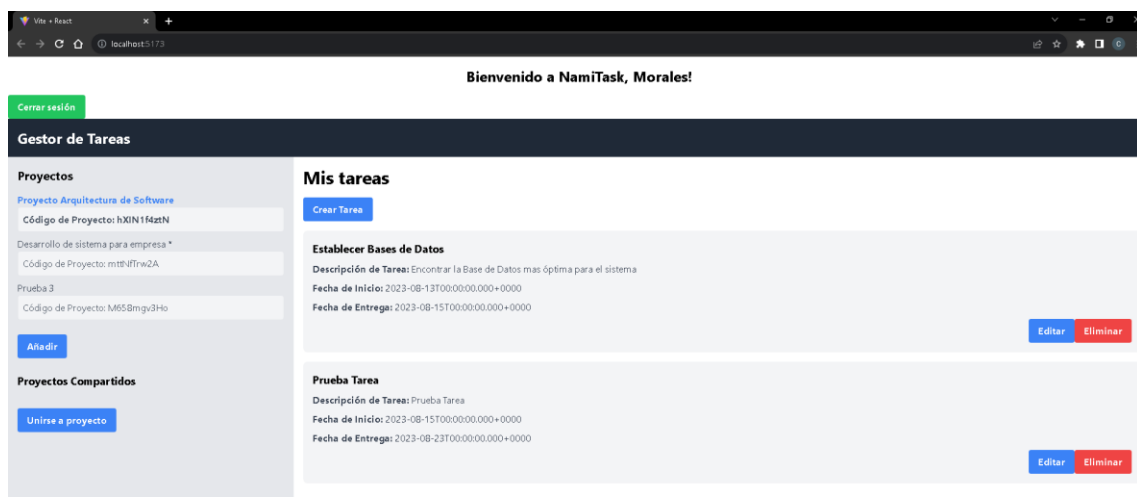
**Descripción de Tarea:** Prueba Tarea

**Fecha de Inicio:** 2023-08-15

**Fecha de Entrega:** 2023-08-23



Y presionamos crear tarea y Guardar, seguido de refrescar la página para notar el cambio.



De ese mismo modo podremos eliminarlas y editarlas en las opciones que parecen en cada una de ellas.

- Ahora podemos ver un código de Proyecto, desde el cual podremos invitar a otros usuarios a nuestro proyecto  
Dicho código deberá ser agregado en la opción Unirse a proyecto y colocarlo

### Proyectos Compartidos

Unirse a proyecto

## Proyectos Compartidos

Unirse

Cancelar

- Una vez agregado ese código ya seremos parte de un proyecto creado por otro usuario: ejemplo

Este proyecto está creado por el usuario cmorales

Cerrar sesión

Bienvenido a NamITask, Morales!

Gestor de Tareas

Proyectos

Proyecto Arquitectura de Software

Código de Proyecto: h0iNTMz9N

Desarrollo de sistema para empresa \*

Código de Proyecto: mttNTTrw2A

Prueba 3

Código de Proyecto: M658mgy2Ho

Añadir

Proyectos Compartidos

Unirse a proyecto

Mis tareas

Crear Tarea

Solicitar Requerimientos

Descripción de Tarea: Hacer reunión con el cliente para aclarar los requerimientos del Sistema

Fecha de Inicio: 2023-08-13T00:00:00.000+0000

Fecha de Entrega: 2023-08-27T00:00:00.000+0000

Editar

Eliminar

Reconocer las Tecnologías necesarias

Descripción de Tarea: Analizar las tecnologías requeridas para el Sistema después de obtener los requerimientos

Fecha de Inicio: 2023-08-27T00:00:00.000+0000

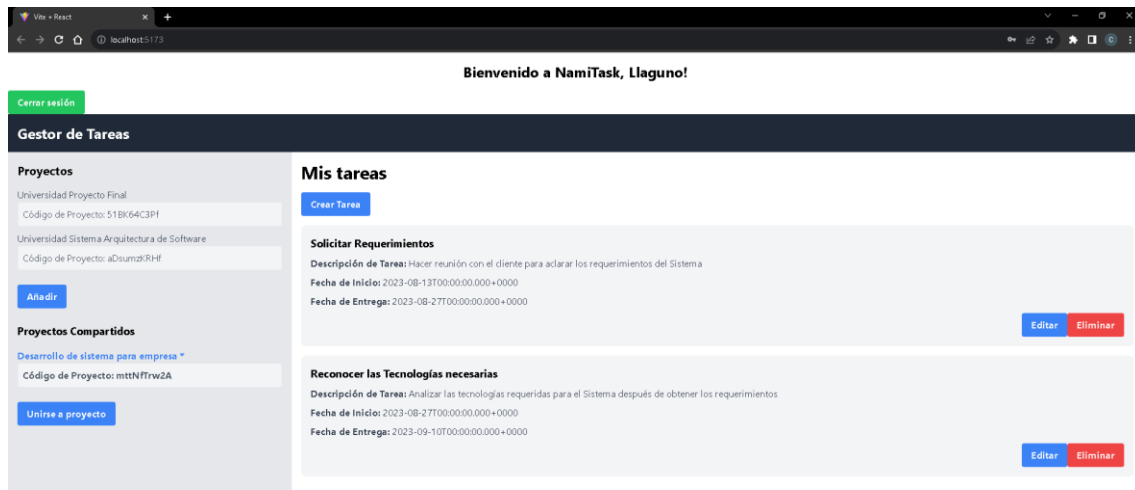
Fecha de Entrega: 2023-09-10T00:00:00.000+0000

Editar

Eliminar

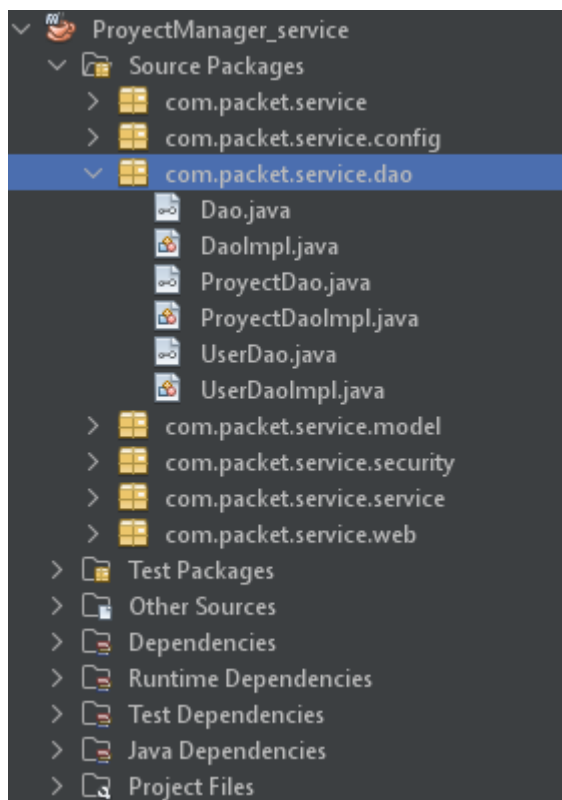
Y el usuario pllaguno, esta unido a este proyecto por el codigo, y este es desplegado en la sección de proyectos compartidos.

Teniendo las mismas funciones de crear, editar y eliminar tareas.



## Patrón Abstract Factory

Este patron se encuentra incorporado en la Capa DAO de la Capa de Servicio



Pues las herramientas para cada Base de Datos son ProjectDao(Mongo) usada para almacenar los registros de proyectos y UserDao(MySql) usada para almacenar los registros de usuarios

Su clase Factory es la siguiente:

```

//
package com.packet.service.dao;

/**
 *
 *
 */
// INTERFACE Dao es la factory(fabrica)de los objetos dao

public interface Dao {
    ProjectDao createProjectDao();
    UserDao createUserDao();
}

```

```

/**
 *
 * Clase DAO Factory
 * parte del patron abstract factory para generar objetos de tipo DAO
 */
@Component
public class DaoImpl implements Dao{

    //Inicializa un Objeto ProjectDao con Spring
    @Autowired
    ProjectDaoImpl projectDao;

    //Inicializa un Objeto UserDao con Spring
    @Autowired
    UserDaoImpl userDao;

    @Override
    public ProjectDaoImpl createProjectDao() {
        return projectDao;
    }

    @Override
    public UserDaoImpl createUserDao() {
        return userDao;
    }
}

```

Y es utilizada para la capa Service cada uno de las bases de Datos:

```

/**
 * Clase de servicio para los objetos DTO Project
 */
@Service
public class ProjectServiceImpl implements ProjectService {

    //conexion con la Capa Dao
    private final ProjectDao projectDao;

    @Autowired
    public ProjectServiceImpl(DaoImpl factory) {
        //metodo constructor que inicializa un objeto ProjectDao con su factory
        this.projectDao = factory.createProjectDao();
    }
}

```

```

/**
 *
 * Capa Service la cual contiene la lógica de negocio del sistema
 * Se comunica con la capa DAO
 */
@Service
public class UserServiceImpl implements UserService {

    // conexion a la CapaDao
    private final UserDao userDao;

    @Autowired
    public UserServiceImpl(DaoImpl factory) {
        //metodo constructor que inicializa un objeto UserDao con su factory
        this.userDao = factory.createUserDao();
    }

    // objeto de seguridad de Spring
    @Autowired
}

```