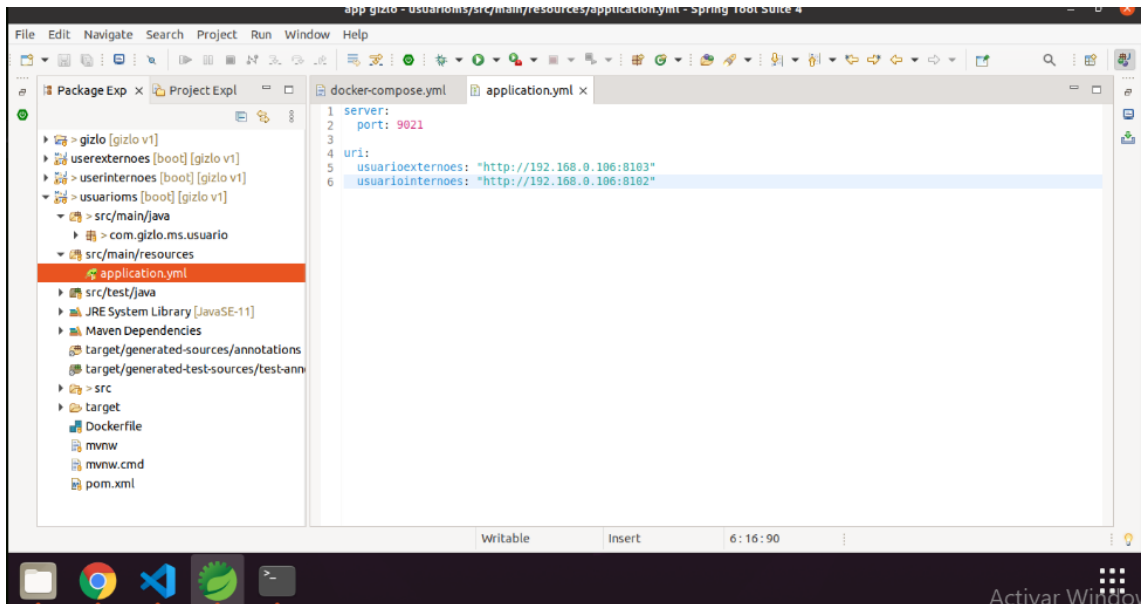


Manual de Instalación

1. Clonar repositorio de GitHub: <https://github.com/Jesus24DarkSaider/gizlo.git>
2. Abrir Servicios de SpringBoot.
3. Modificar el Application.yml y especificar la ip de la interfaz de red del host donde se va a levantar el microservicio.



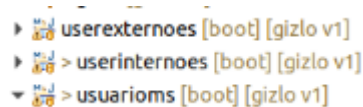
4. Ejecutar comando ifconfig en caso de no tener la ip a la mano.

```
vethdd25ef8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::8c02:74ff:fe1c:7870 prefixlen 64 scopeid 0x20<link>
    ether 8e:02:74:1c:78:70 txqueuelen 0 (Ethernet)
    RX packets 18628 bytes 2454702 (2.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14151 bytes 12737313 (12.7 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.106 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::c78e:b4da:fcc5:e916 prefixlen 64 scopeid 0x20<link>
    ether 24:fd:52:54:43:45 txqueuelen 1000 (Ethernet)
    RX packets 970919 bytes 1221327082 (1.2 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 514512 bytes 83804518 (83.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

jesus@jesus-HP-1000-Notebook-PC:~/Escritorio/front/gizlo/appgizlo$
```

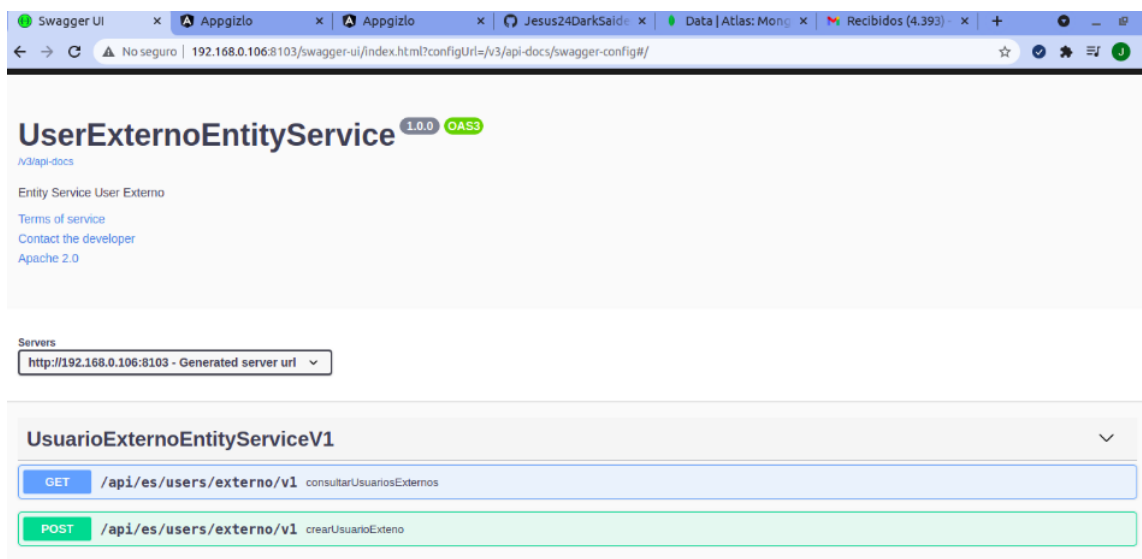
5. Después de especificar la interfaz de red en el Application.yml se debe guardar los cambios, luego dar click derecho sobre el proyecto y realizar un maven install.
Nota: Realizar el maven install de los demás servicios especificados en la siguiente imagen.



6. Ejecutar el archivo `docker-compose.yml` de la solución backend que está ubicada en la carpeta `gizlo` (directorio de los servicios).

userexteriores	ambiente contenerizado	yesterday
userinteriores	ambiente contenerizado	yesterday
usuarioms	se añade habilita el cors para el consumo desde cualquier host	6 hours ago
.gitignore	Initial commit	2 days ago
.project	Iniciando repositorio	2 days ago
README.md	Create README.md	2 days ago
<u>docker-compose.yml</u>	ambiente contenerizado	yesterday

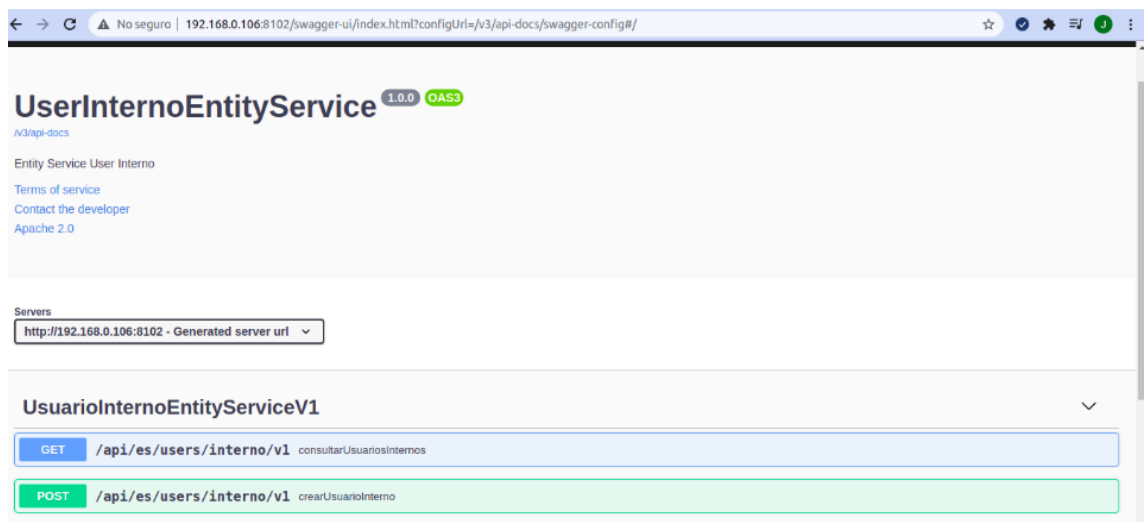
7. Una vez ejecutado el archivo `docker-compose.yml` esperar un minuto hasta que carguen los contenedores, una vez que estén disponibles puede acceder al contrato de los servicios mediante la ip de la interfaz de red junto al puerto especificado al contenedor.



The screenshot shows the Swagger UI for the `UserExternoEntityService` API. The browser address bar indicates the URL `192.168.0.106:8103/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config#/`. The service is labeled with version `1.0.0` and `OAS3`. Below the service name, there are links for `v3/api-docs`, `Entity Service User Externo`, `Terms of service`, `Contact the developer`, and `Apache 2.0`. A `Servers` dropdown menu shows `http://192.168.0.106:8103 - Generated server url`. The `UsuarioExternoEntityServiceV1` section is expanded, showing two endpoints: a `GET` endpoint `/api/es/users/externo/v1` with description `consultarUsuariosExternos`, and a `POST` endpoint `/api/es/users/externo/v1` with description `crearUsuarioExterno`.



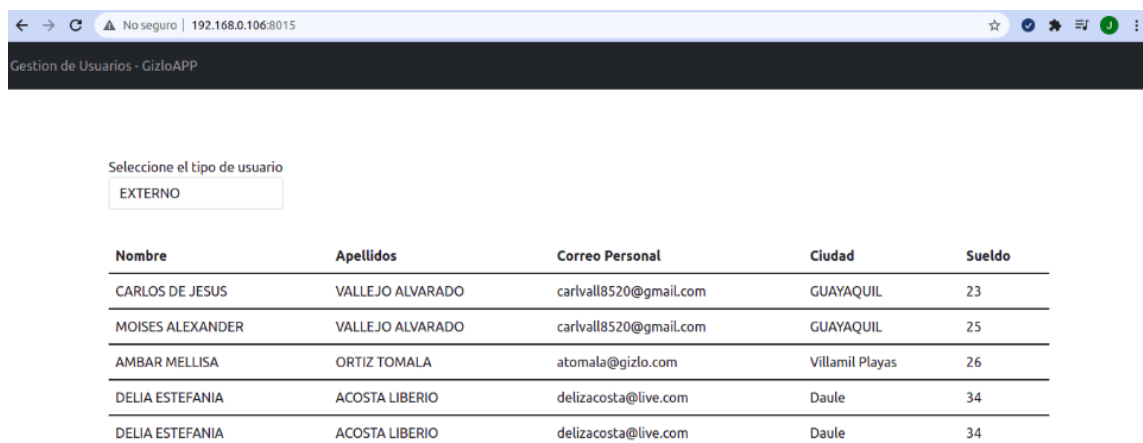
The screenshot shows the Swagger UI for the `Microservicio de Gestion de Usuarios` API. The browser address bar indicates the URL `192.168.0.106:9021/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config#/`. The service is labeled with version `1.0.0` and `OAS3`. Below the service name, there are links for `v3/api-docs`, `Microservicio Usuario`, `Terms of service`, `Contact the developer`, and `Apache 2.0`. A `Servers` dropdown menu shows `http://192.168.0.106:9021 - Generated server url`. The `UsuarioMSV1` section is expanded, showing two endpoints: a `POST` endpoint `/api/ms/users/interno/v1` with description `crearUsuarioInterno`, and a `POST` endpoint `/api/ms/users/externo/v1` with description `crearUsuarioExterno`.

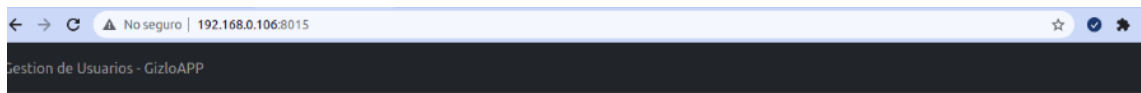


8. Para levantar el Aplicativo en angular solo se debe ejecutar el archivo docker-compose del aplicativo que se encuentra en la ruta appgizlo/src y esperar hasta que se compile la imagen del contenedor (puede tardar 5 min debido a que se instalan dependencias de node y retarda la construcción de la imagen).

src	mejora	6 hours ago
.browserslistrc	Iniciando repositorio	yesterday
.editorconfig	Iniciando repositorio	yesterday
.gitignore	Iniciando repositorio	yesterday
Dockerfile	dockerfile	6 hours ago
README.md	Iniciando repositorio	yesterday
angular.json	APP 80%	8 hours ago
docker-compose.yml	SE AGREGA CONFIGURACION CENTRALIZADA PARA LEVANTAR EL APLICATIVO	6 hours ago
karma.conf.js	Iniciando repositorio	yesterday
package-lock.json	APP 80%	8 hours ago
package.json	APP 80%	8 hours ago
tsconfig.app.json	Iniciando repositorio	yesterday
tsconfig.json	Iniciando repositorio	yesterday
tsconfig.spec.json	Iniciando repositorio	yesterday

9. Después del proceso anterior se espera un minuto para que se pueda acceder al sitio publicado.





Seleccione el tipo de usuario

INTERNO

Nombre	Apellidos	Correo Corporativo	Departamento	Sueldo
BYRON PAUL	VALLEJO ALVARADO	byro@mail.com	SISTEMAS	2000
JORGE ALBERTO	OVIEDO ALVARADO	joseoviedo@gizlo.com	MARKETING	450
NAOMI KARLA	CRUZ MOREIRA	nmoreira@gizlo.com	SISTEMAS	560

10. En caso de tener algún inconveniente con el aplicativo puede apuntar al ip del microservicio por default se está de la siguiente forma.

```

GIZLO
├── appgizlo
│   ├── node_modules
│   ├── src
│   │   ├── app
│   │   │   ├── model
│   │   │   └── service
│   │       ├── usuario.service.spec.ts
│   │       └── usuario.service.ts
│   ├── .dockerignore
│   ├── app-routing.module.ts
│   ├── app.component.css
│   ├── app.component.html
│   ├── app.component.spec.ts
│   ├── app.component.ts
│   ├── app.module.ts
│   ├── model.tipousuario.ts
│   ├── assets
│   │   ├── environments
│   │   ├── favicon.ico
│   │   ├── index.html
│   │   ├── main.ts
│   │   ├── polyfills.ts
│   │   └── styles.css
└── ...

appgizlo > src > app > service > ts usuario.service.ts > UsuarioService
1 import { Injectable } from '@angular/core';
2 import { TipoUsuario } from '../model.tipousuario';
3 import { UsuarioExternoModel } from '../model/UsuarioExternoModel';
4 import { UsuarioInternoModel } from '../model/UsuarioInternoModel';
5 import { HttpClient } from '@angular/common/http';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class UsuarioService {
11
12   private host = "http://localhost:9021";
13
14   private urlUsuarioInterno = "/api/ms/users/tipo/v1/USUARIO_INTERNO";
15   private urlUsuarioExterno = "/api/ms/users/tipo/v1/USUARIO_EXTERNO";
16
17   private tiposDeUsuario : TipoUsuario[] = [
18     { tipo : 'USUARIO_EXTERNO', descripcion : 'EXTERNO' },
19     { tipo : 'USUARIO_INTERNO', descripcion : 'INTERNO' }
20   ]
21
22   obtenerTipoUsuario(): TipoUsuario[] {
23     return this.tiposDeUsuario;
24   }
25 }

jesus@jesus-NP-1000-Notebook-PC:~/Escritorio/front/gizlo$ docker build
```