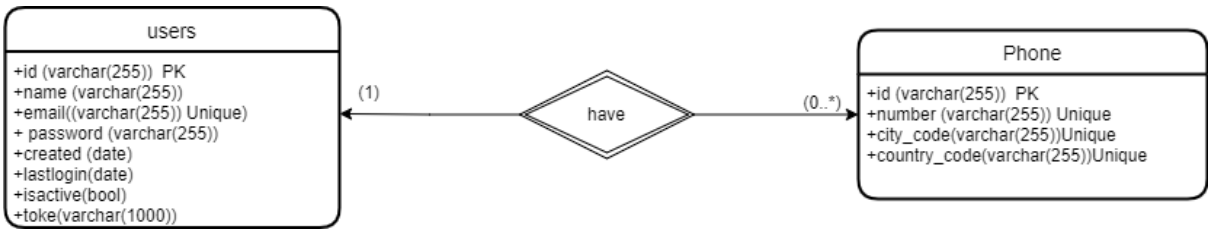
Documentación técnica

1.Base de datos

Se utilizó la base de datos que viene integrada H2.

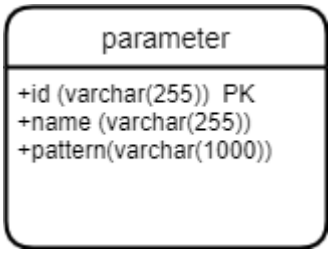
1.1 Tablas principales

La relación entre las tablas: Un usuario puede o no tener muchos teléfonos, un teléfono debe tener un usuario



1.2 Tabla secundaria

Esta tabla almacena los parámetros que serán configurables desde la aplicación.(Para esta versión solo se almacenarán parámetros para validación RegExr)



2. Funcionalidades Principales para la primera versión

2.1 Registro de un usuario y sus teléfonos

Para acceder a esta funcionalidad tendremos que ejecutar el proyecto si estamos en un ambiente local la url será esta:

es un servicio post **http://localhost:8080/user**

ejemplo de curl

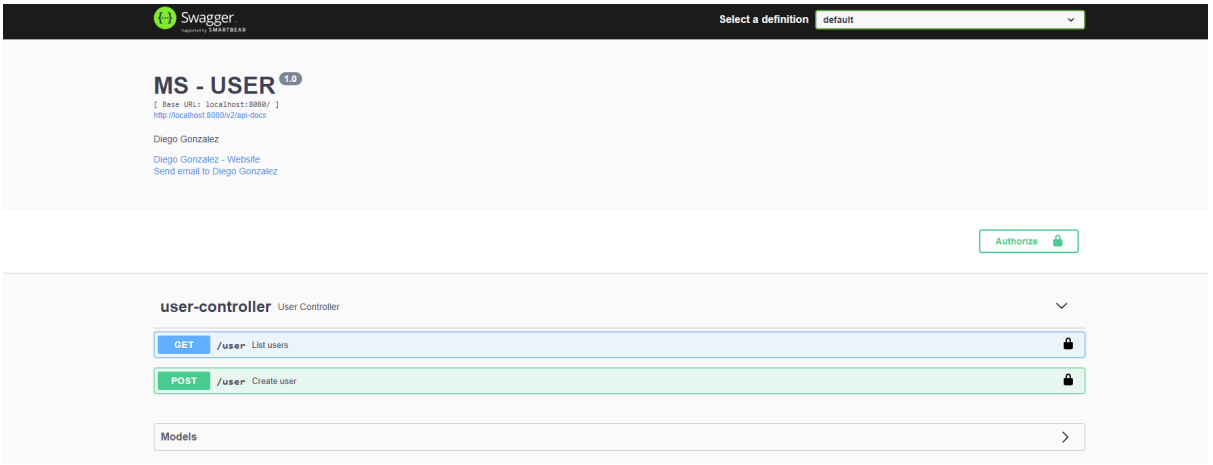
curl -X POST "http://localhost:8080/user" -H "accept: */*" -H "Content-Type: application/json" -d "{ \"email\": \"diegoaliriogm@gmail.com\", \"name\": \"string\", \"password\": \"A!$12#\", \"phones\": [{ \"cityCode\": \"string\", \"contryCode\": \"string\", \"number\": \"string\" }]}"

http://localhost:8080/swagger-ui/index.html#/

Si estamos en un ambiente desplegado en el servidor

http://msusers-env.eba-eqkm3rfj.us-east-1.elasticbeanstalk.com/swagger-ui/index.html#/

Se nos mostrara la pagina de swagger



debemos ir a post Create user que nos pide el siguiente request:

```
{
  "email": "diegoaliriogm@gmail.com",
  "name": "string",
  "password": "A!qweqeq$12#",
  "phones": [
    {
      "cityCode": "string",
      "contryCode": "string",
      "number": "string"
    }
  ]
}
```

existen algunas **validaciones** principales:

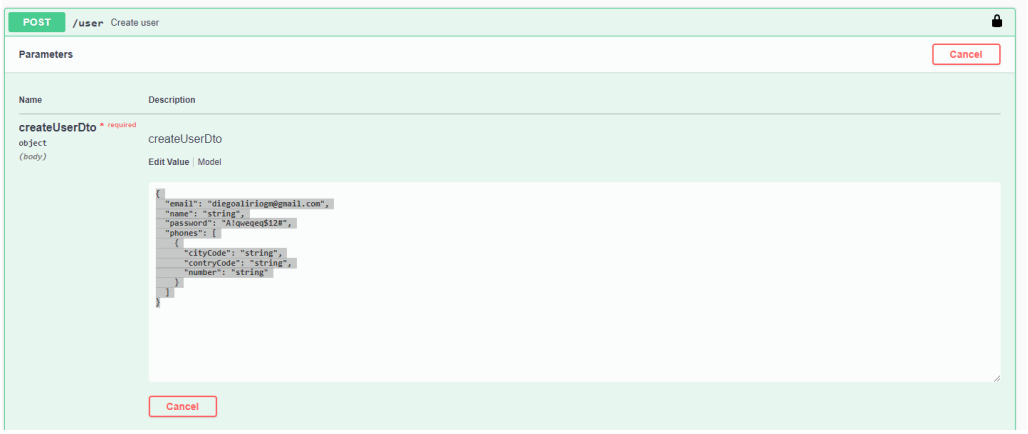
- La del email que tiene que ser el formato específico de un correo electrónico:

```
@Pattern(regexp = REGEDEXEMAIL,  message = "No es un formato valido para un email")
private String email;
```

- La de la contraseña que utiliza el siguiente RegExr
  `^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#&()–[{}]:;',?/*~$^+=<>]).{8,20}$`
  que traduce :
  La contraseña debe contener al menos un carácter en minúscula, un carácter en mayúscula, un dígito, un carácter especial y una longitud entre 8 y 20.
- Para que no existan teléfonos repetidos se creó un constraint para evitar eso.

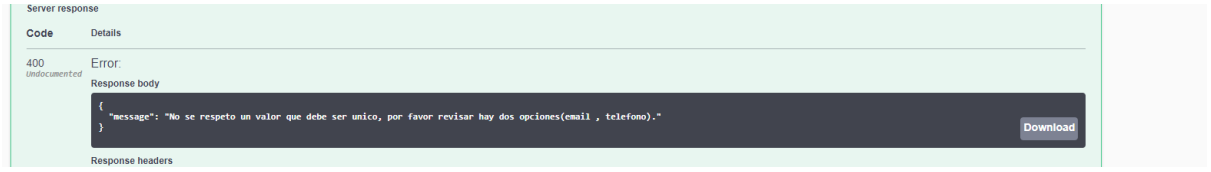
```
@Entity
@Table(name = "phone" ,uniqueConstraints=
@UniqueConstraint(columnNames={"number", "city_code","country_code"}))
```

Ejemplo de este endpoint:

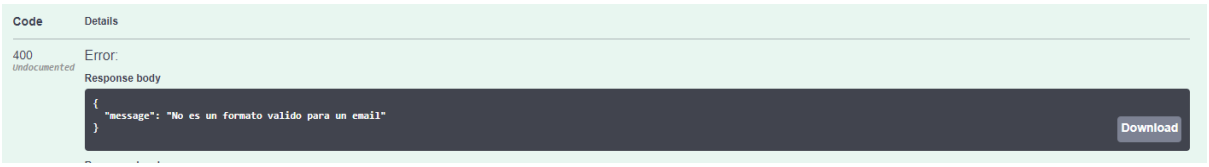


Al ejecutarlo nos traerá este resultado

Al ejecutarlo y si no pasa la validación de columnas únicas nos traerá este resultado



Al ejecutarlo y si no cumple la validación del email:



y si no cumple el valor de la contraseña:

Nota: para poder crear un usuario no debemos estar logueados.

2.2 Listado de los usuarios y sus datos:

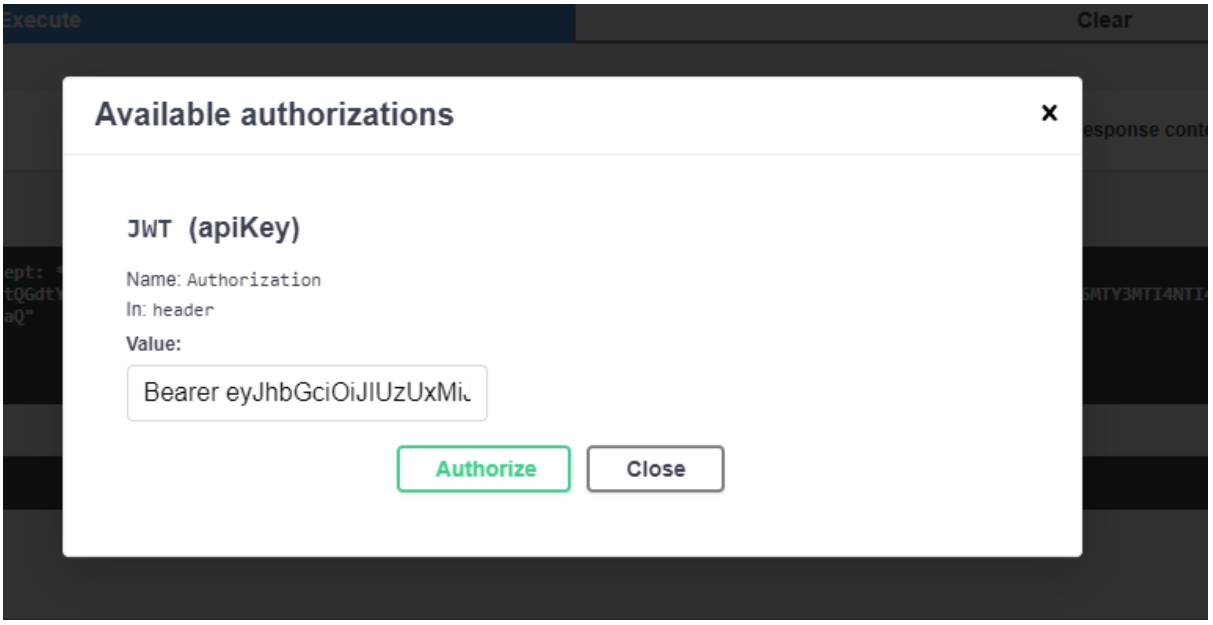
Esto nos permitirá saber que usuarios están registrados en el sistema, debemos ir a ejecutar:

http://localhost:8080/swagger-ui/index.html#/

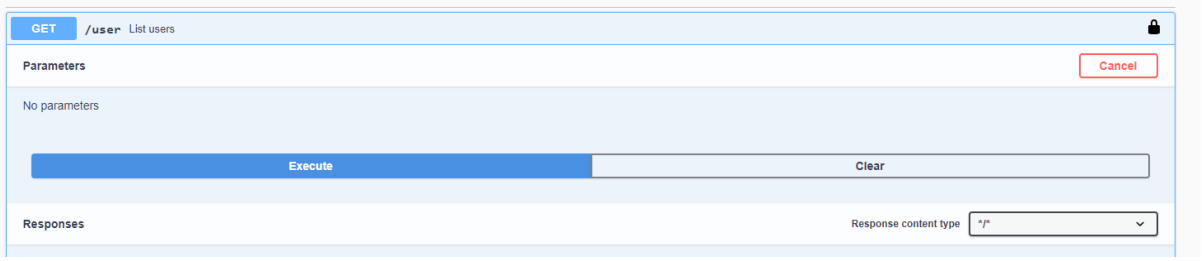
Este es un ejemplo de curl
curl -X GET "http://localhost:8080/user" -H "accept: */*" -H "Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJqdGkiOiJkaWVnb2FsaXJpb2dtQGdtYWlsLmNvbSIsInN1

YiI6InN0cmluZyIsImF1dGhvcml0aWVzIjpbIlJPTEVfVVNFUiJdLCJpYXQiOjE2NzEyODQ2ODMsImV4cCI6MTY3MTI4NTI4M30.0hfPuG0EexnZBrz04rUZZ1dhpBmZqsWaS3_3kQEp6uEOf0UmFrvsyYs0m5Je_pxApzUmdycOG6b9-AcIvlNOaQ"

Para ejecutar tendremos que enviar el token que nos da como resultado al registrar un usuario



le damos Authorize.



le damos Execute y nos trae el siguiente resultado.

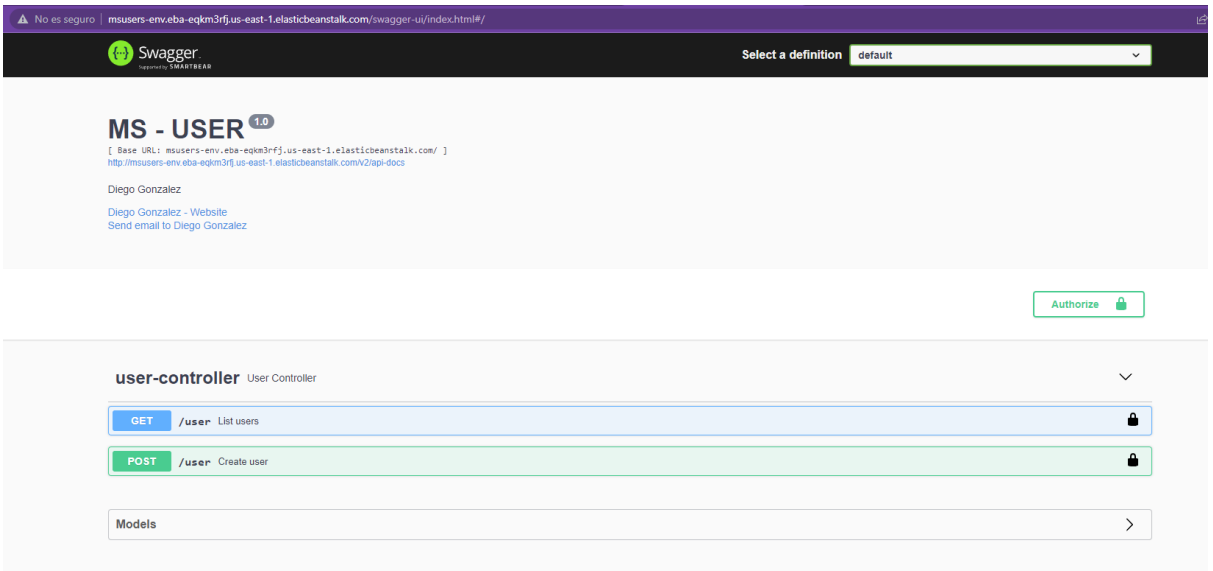

```
[
  {
    "id": "b6186628-bb3b-43bf-a8d7-6c00bb32ef72",
    "name": "string",
    "email": "diegoaliriogm@gmail.com",
    "password":
"$2a$10$pQJwfVFfwQ38hirobERNk.mBzbAT0EIc6d5imrnMPlWEnrbvFYsLu",
    "phones": [
      {
        "number": "string",
        "cityCode": "string",
        "contryCode": "string"
      }
    ],
    "created": [
```

```
      2022,
      12,
      17
    ],
    "lastLogin": [
      2022,
      12,
      17
    ],
    "token": "Bearer
eyJhbGciOiJIUzUxMiJ9.eyJqdGkiOiJkaWVnb2FsaXJpb2dtQGdtYWlsLmNvbSIsInN1YiI6InN
0cmluZyIsImF1dGhvcml0aWVzIjpbIlJPTEVfVVNFUiJdLCJpYXQiOjE2NzEyODk5MzMsImV4
cCI6MTY3MTI5MDUzM30.pr-BHezB2eq4nhMqsPms1et3OGQ9i2hSLXG551RDbzOrZTWM
BAnv4SkMgGTN2qxC58iHFaMLvGM2-7NICetwtg",
    "active": true
  }
]
```


Nota: para poder crear un usuario no debemos estar logueados.

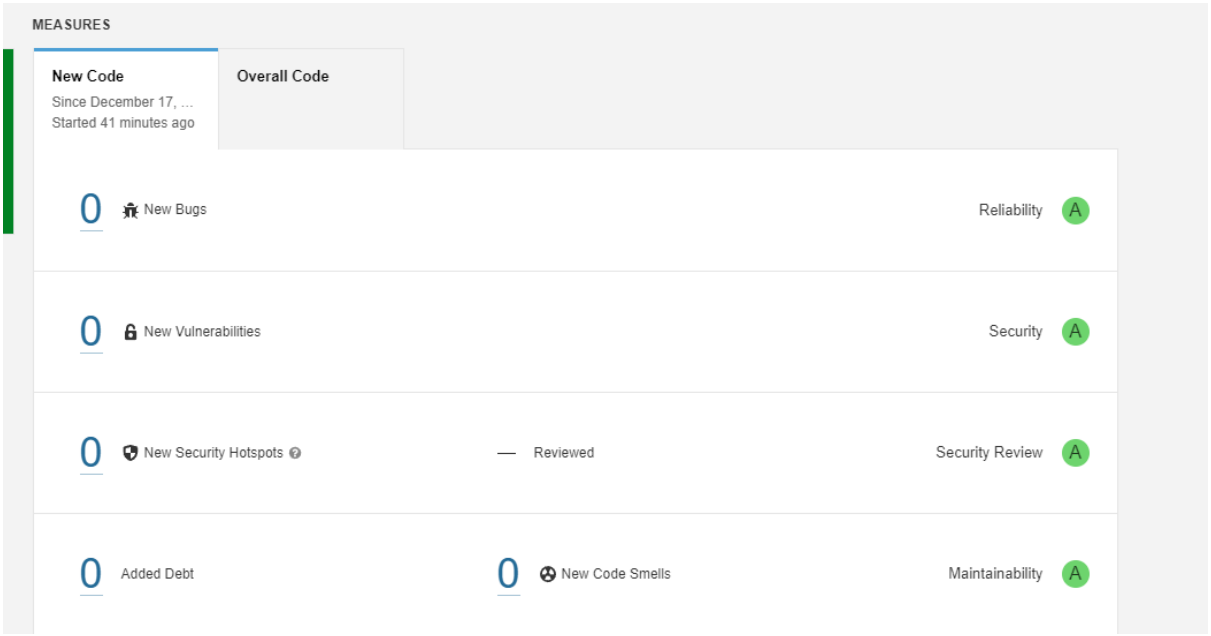
3 Despliegue en AWS

Este proyecto se desplegó en la nube de aws específicamente en el servicio de Elastic Beanstalk, la url del proyecto la podremos ver en : http://msusers-env.eba-eqkm3rfj.us-east-1.elasticbeanstalk.com/swagger-ui/index.html#/user-controller/getAllUsingGET



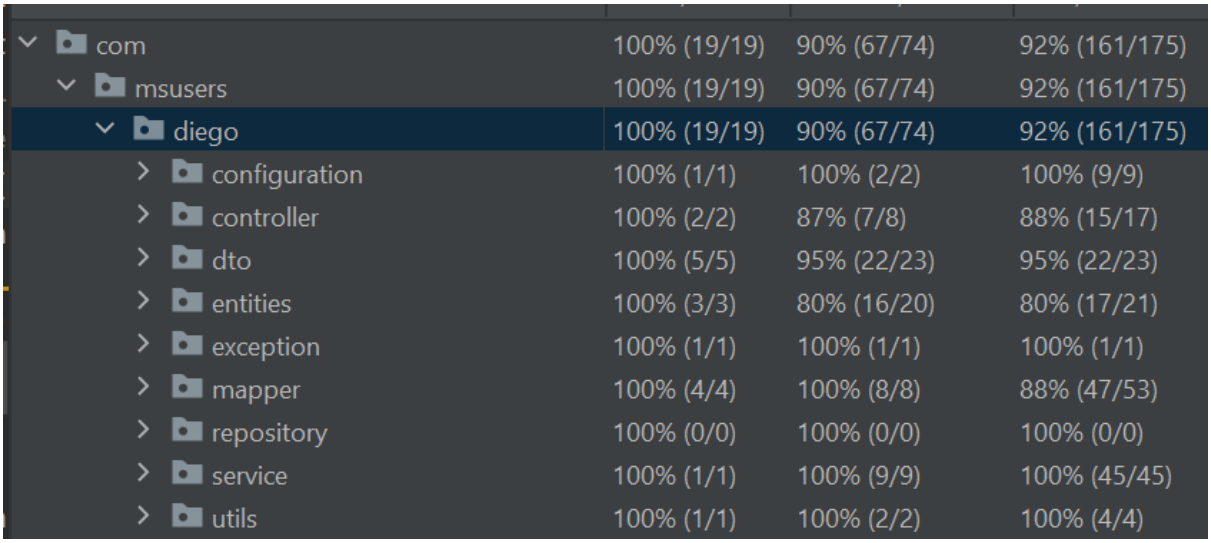
4 Métricas SonarQube

Se realizó un pequeño análisis de código con esta herramienta arrojandolos el siguiente resultado.



5 Cobertura de código:

Se realizaron pruebas unitarias y nos arrojó la siguiente cobertura.



6. Mejoras para siguientes versiones

- Personalizar mejor la excepción causada por email y teléfono para que den un mensaje más específico.
- Implementar todos los métodos de autorización(Login, logout,refresh token).
- Implementar más métodos para usuarios (eliminar,activar, desactivar,actualizar)
- Implementar base de datos externa al proyecto