**DISEÑO WISH LIST**

**By Betzy Salas Vergara**

1. **DISEÑO**
   1. **DIAGRAMA DE ARQUITECTURA**

**Diagrama

Descripción generada automáticamente**

* 1. **DIAGRAMA DE CASOS DE USO**

**Diagrama

Descripción generada automáticamente**

* 1. **DIAGRAMA DE CLASES**
     1. **DIAGRAMA DE CLASES MICRO SERVICIO MS-USERS**

**Diagrama

Descripción generada automáticamente**

* + 1. **DIAGRAMA DE CLASES MICRO SERVICIO MS-SEGURIDAD**

**Diagrama

Descripción generada automáticamente**

* 1. **DIAGRAMA DE COMPONENTES**

Diagrama

Descripción generada automáticamente

1. **BASE DE DATOS**
   1. **MODELO ENTIDAD RELACION**

Para el motor de base de datos se ha seleccionado utilizar el servicio de RDS de AWS, utilizando el motor de base de datos POSTGRESQL, instalada en un tipo de instancia t3.micro.

Las reglas InBound y OutBound del Grupo de seguridad asignado, quedaron autorizadas

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Información para conexión desde cualquier cliente sql

jdbc:postgresql://database-pruebas.c7t2t2hgr1yd.us-east-1.rds.amazonaws.com/database\_prueba

usuario : postgres

password: 987654321

Diagrama

Descripción generada automáticamente

* 1. **DISEÑO DE LAS APIS DE CADA MICROSERVICIO**
     1. **MICROSERVICIO USERS**



* + 1. **MICROSERVICIO SECURITY**



* + 1. **MICROSERVICIO WISH\_LIST**



* + 1. **MICROSERVICIO API\_BOOKS**

1. **DESARROLLO**
   1. **TECNOLOGIA**

* La tecnología seleccionada para el desarrollo es Java utilizando el framework Spring Boot, utilizando las siguientes dependencias:
  + spring-boot-starter-data-jpa
  + spring-boot-starter-web
  + spring-boot-starter-validation
  + postgresql
  + spring-boot-starter-test
  + spring-cloud-starter-openfeign
  + spring-cloud-starter-netflix-eureka-client
* Se utiliza Eureka para realizar descubrimiento de los diferentes microservicios
* Se utiliza Feign para realizar la comunicación entre los diferentes microservicios
* Se utiliza Spring cloud Gateway para crear un único punto de enlace para los microservicios
  1. **VERSIONAMIENTO**

Como herramienta de versionamiento se utiliza Git y repositorio GitHub

1. **DESARROLLO**

* Se desarrollan los microservicios en Java8:
  + Microservicio Users: Para el manejo de los usuarios. Crear, consultar
  + Microservicio wish: Para la administración de las Listas de Deseos y de los ítems relacionados con las listas de deseos
  + MicroServicio Security: Para generar y validar el token de sesión para controlar el acceso autorizado
  + Microservicio Books: Para conextarse con la Api de Google Books utilizando rest template para la extracción de la información
  + Microservicio Eureka: Para implementar el patrón Service Discovery para poder registrar y descubrir los microservicios y permitir la comunicación entre ellos sin tener que usar Rest Template
  + Micro servicio Gateway: Para implementar el patrón api Gateway y poder tener un único punto de acceso a los diferentes micro servicios
* Se utilizan algunos patrones de diseño tales como la inyección de dependencias, la segregación de interfaces, responsabilidades únicas.
* Se contenerizan los micro servicios y se crean dentro de una misma network
* No se realiza cliente web (no es mi fuerte)

1. **DISEÑO DE ARQUITECTURA EN LA NUBE PROPUESTO**

Diagrama

Descripción generada automáticamente