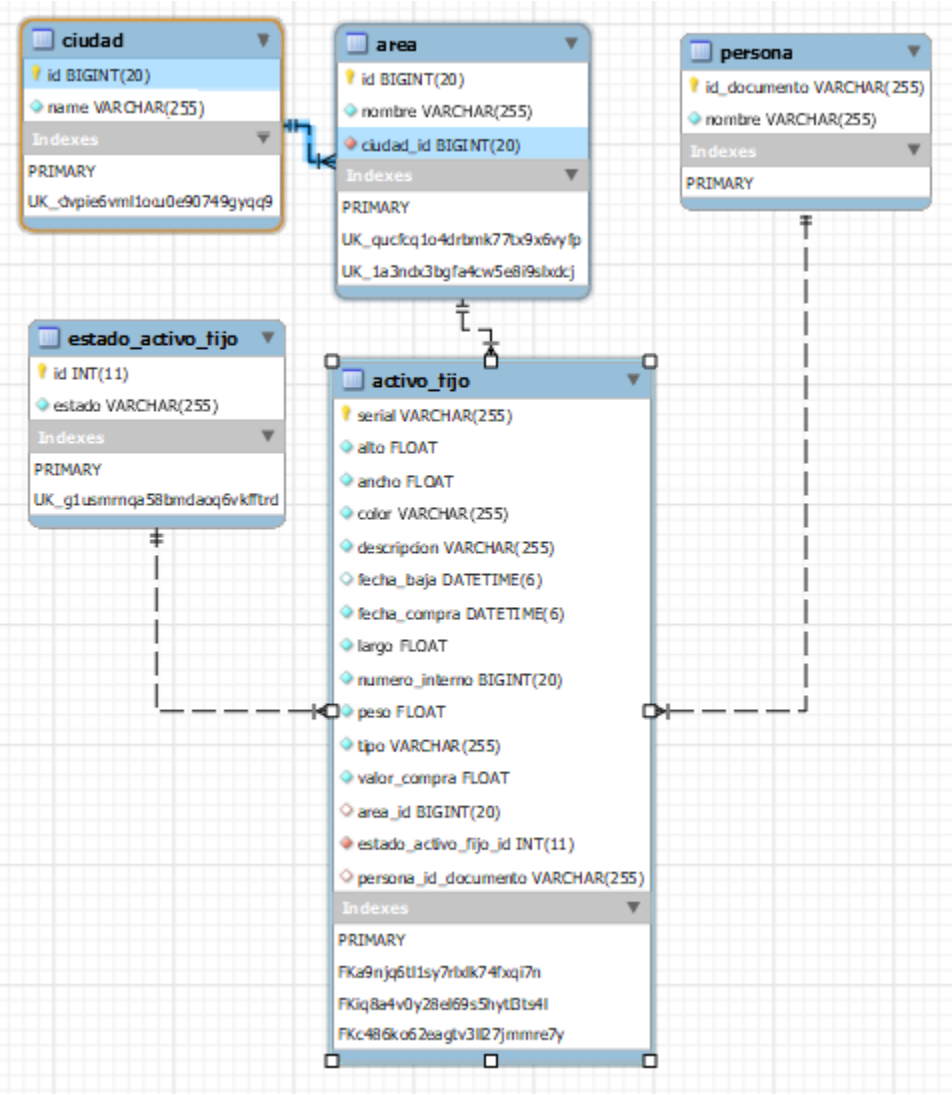
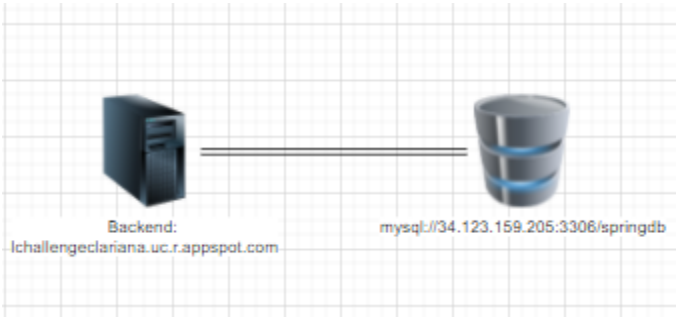
Modelo ER:

Módelo entidad relación de la base de datos:



Arquitectura:



Tecnologías:

- **Spring boot java:** es una infraestructura ligera que elimina la mayor parte del trabajo de configurar las aplicaciones basadas en Spring.
- **Spring Boot JPA:** es una especificación de Java para administrar datos relacionales en aplicaciones Java. Nos permite acceder y conservar datos entre el objeto / clase Java y la base de datos relacional.
- **Spring-boot-starter-web:** se utiliza para crear la aplicación web, incluidas las aplicaciones RESTful que utilizan Spring MVC. Utiliza Tomcat como contenedor integrado predeterminado.
- **Spring-boot-devtools:** el objetivo del módulo es intentar mejorar el tiempo de desarrollo mientras se trabaja con la aplicación Spring Boot. Spring Boot DevTools recoge los cambios y reinicia la aplicación.
- **Mysql-connector-java**: se utiliza un controlador JDBC para conectarse a una base de datos, en este caso a MySQL.
- **Spring Boot Maven plugin:** proporciona compatibilidad con Spring Boot en Apache Maven. Le permite empaquetar archivos jar o war ejecutables, ejecutar aplicaciones Spring Boot, generar información de compilación e iniciar su aplicación Spring Boot antes de ejecutar las pruebas de integración.
- **Spring tool suit:** es un IDE para desarrollar aplicaciones Spring. Es un entorno de desarrollo basado en Eclipse. Proporciona un entorno listo para usar para implementar, ejecutar, implementar y depurar la aplicación. Valida nuestra aplicación y proporciona soluciones rápidas para las aplicaciones.
- **Mysql Db:** la base de datos del proyecto.

Instructivo:

1. **Buscar todos los activos:** se debe ejecutar la siguiente Url:

   1.1. **URL**: https://challengeclariana.uc.r.appspot.com/api/activosFijos/obtenerActivosFijos

   1.2. **Tipo**: GET

2. **Obtener activos fijos por fecha de compra, tipo y serial:**

   2.1. URL: https://challengeclariana.uc.r.appspot.com/api/activosFijos/obtenerActivosFijosPorFecha

   2.2. Tipo: POST

   2.3. REQUEST (ejemplo):

```
{
  "serial": "1",
  "fechaCompra": "2020-09-18"
}
```

3. **Crear nuevos activos:**

   3.1. URL: https://challengeclariana.uc.r.appspot.com/api/activosFijos/crearActivo

   3.2. Tipo: POST

   3.3. REQUEST (ejemplo): este reques aplicaría para cuándo el activo se asigna a una persona:

```
{
    "serial": "8",
    "numeroInterno": 1,
    "tipo": "prueba",
    "descripcion": "prueba",
    "peso": 1.0,
    "alto": 1.0,
    "ancho": 1.0,
    "largo": 1.0,
```

```
    "valorCompra": 1.0,
    "fechaCompra": "2020-09-18T05:00:00.000+00:00",
    "fechaBaja": "2020-09-18T05:00:00.000+00:00",
    "estadoActual": {
        "id": 1
    },
    "persona": {
        "id_documento": "80834079"
    },
    "area": null,
    "color": "azul"
}
```


3.4. REQUEST (ejemplo): este reques aplicaría para cuándo el activo se asigna a una área:


```
{
  "serial": "9",
  "numeroInterno": 1,
  "tipo": "prueba",
  "descripcion": "prueba",
  "peso": 1.0,
  "alto": 1.0,
  "ancho": 1.0,
  "largo": 1.0,
  "valorCompra": 1.0,
  "fechaCompra": "2020-09-18T05:00:00.000+00:00",
  "fechaBaja": "2020-09-18T05:00:00.000+00:00",
  "estadoActual": {
```

```
    "id": 1
  },
  "persona": null,
  "area": {
    "id": 4
  },
  "color": "azul"
}
```


4. **Actualizar activos:** Solamente se puede actualizar el serial y la fecha de baja:

4.1. URL: https://challengeclariana.uc.r.appspot.com/api/activosFijos/actualizarSerialFechaActivos

4.2. Tipo: POST

4.3. REQUEST (ejemplo): formato de la fecha debe ser como el estandar del ejemplo:


```
{
  "serial":1,
  "numeroInterno": 300,
  "fechaBaja": "2020-09-18T05:00:00.000+00:00"
}
```


5. Obtener áreas: se debe ejecutar la siguiente Url:

1.1. **URL**: https://challengeclariana.uc.r.appspot.com/api/area/obtenerAreas

1.2. **Tipo**: GET

6. Obtener personas: se debe ejecutar la siguiente Url:

1.1. **URL**: https://challengeclariana.uc.r.appspot.com/api/persona/obtenerPersonas

1.2. **Tipo**: GET