**Práctica 2 : ApiRest Reserva de Bicicletas**

**Asignatura:** Sistemas Distribuidos

**Autores:** Daniel Requena Garrido

Raúl Rodríguez López-Rey

**INDICE**

[Resumen de la arquitectura: 3](#_Toc104471326)

[Entidades 4](#_Toc104471327)

[Repositorios: 5](#_Toc104471328)

[Services: 5](#_Toc104471329)

[Controlador: 5](#_Toc104471330)

[Modelo Conceptual Base de datos: 7](#_Toc104471331)

Esta memoria corresponde con el proyecto que se tiene que ejecutar en el puerto 8080, ApiRest Reserva de Bicicletas

# Resumen de la arquitectura:

La arquitectura de este proyecto se basa en:

* 2 entidades: Se encuentran dentro del paquete model (Estacion.java, Bicicleta.java).
* 1RestController: ReservaRestController.java(se encuentra en el paquete restController).
* 2 repositorio: Encontrado en el paquete repository (RepoBicicletas.java, RepoEstacion.java)
* 2 service: En el paquete service EstacionService y BicicletaService. Aquí se implementan algunos métodos que nos ayuda a trabajar con los repositorios.
* 1 inicializador de Bases de Datos: En el paquete principal y denominado DBInitializer.
* 1 clase denominada IDs en la clase principal que sus atributos son los ids de los usuarios, bicis y estaciones que posteriormente usaremos en el restController.

**Entidades**:

* estacionBicicletas: Al igual que Usuario, contamos con un atributo id de tipo long, dos tipos integer llamado numeroSerie y capacidad, 3 strings estado, coordenadas , fechaInstalacion y un list de Bicicletas con la etiqueta @OneToMany que se relaciona con la entidad Bicicletas.

El constructor recibe el numero de serie, coordenadas y la capacidad (que será 5 o 10).El atributo bicis se inicia como una lista vacia, estado por defecto activo y la fecha de creación del objeto.

Geters y Seters además de un agregarBici y eliminarBici que lo usamos para debuggear más que nada .

* Bicicleta: El id\_b de tipo long, numeroSerie, modelo, fechaAlta y estado de tipo String y un tipo estacionBicicletas llamado estación etiquetada con @OneToOne que se relaciona con la entidad estacionBicicletas.

Tres constructores, por defecto, uno que recibe el numeroSerie, modelo estado se estipula Sin-Base y la fecha de creación de objeto) y el tercero(que recibe lo mismo + el estado).

Geters y Seters necesarios.

# **Repositorios**:

* RepoEstacion: Interface que extiende JpaRepository<Estacion,Long>.
* RepoBicicletas: Interface que extiende JpaRepository<Bicicleta,Long>.

# **Services**:

En los services en resumidas cuentas usamos métodos de los mismos repositorios o métodos que implican varios métodos del repositorio y los juntamos en uno solo. En el controladore usamos métodos de los repositorios y services indistintivamente.

# Controlador:

**ReservaRestController**:

Se establece como RequestMapping la dirección “api/reservas”.

EL método de reservar, con *Postman* hacemos una petición de put a “/” y en el cuerpo de la petición tenemos que poner los ids del usuario que va a reservar, el id de la bici que se va a reservar y el id de la estación en donde está la bici.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La anterior imagen muestra el formato.

Si no se puede procesar el método porque no se dan las condiciones necesarias se responde con ResponseEntity.*unprocessableEntity*().build();

Si por el contrario no se puede encontrar al usuario se responde con notFound como en el resto de los métodos ( de este proyecto y del otro). Este método utiliza un restTemplate a "http://localhost:8081/api/usuarios/"+ids.getId\_usuario() de esta manera podemos obtener y modificar el saldo del usuario que pretende reservar la bicicleta(usaremos un ObjectNode para conseguirlo). Se cambia el estado de la bici y se marca como reservada(y se quita de la lista de bicis de estación)

El método de liberar o devoluciones , funciona de igual manera que el anterior. Haremos un put a “/devoluciones” con el body de la request con los ids como hemos mencionado anteriormente. El método devolverá mediante el restTemplate la fianza al usuario y asignará la bici a una estación y la marcará como en base.

# Modelo Conceptual Base de datos:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente