Práctica 1 : BiciUrjc

Asignatura: Sistemas Distribuidos

Autores: Daniel Requena Garrido
Raúl Rodríguez López-Rey

INDICE

Resumen de la arquitectura:

La arquitectura de este proyecto se basa en:

- 3 entidades: Se encuentran dentro del paquete model (Bicicleta.java, estacionBicicletas.java y Usuario.java).
- 4 controladores: BiciUrjcController.java(se encuentra en el paquete principal y se encargar del index), BicicletaController.java, EstacionBicicletasController.java y UsuarioController.java que se encuentran en el paquete controller y se encargan del modulo de gestión de bicicletas, estaciones y usuario respectivamente.
- 3 repositorios: Encontrados en el paquete repository y se llaman ReploBicicletas.java, RepoEstacionBicis.java y RepoUsuario.java
- 3 services: En el paquete service, BicicletaService, EstacionService y UserService. 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.

Comentario Adicional:

La aplicación biciUrjc la hemos subido a un Ubuntu Server en .jar y redireccionado a una url. La aplicación se puede encontrar introduciendo en un navegador … http://www.raulrodriguezlr.es/

La aplicación esta hecha en java 17 y comprobado en un navegador Chrome con Zoom al 100%.

Entidades:

- Usuario: Sus atributos son una variable tipo long que será el id (generado de manera automática), nombre, apellido, contraseña, estado y fecha de alta, estos últimos de tipo String.

  El constructor recibe como parámetros 3 strings, nombre, apellido y contraseña. El estado por defecto se pone en activo y la fecha de alta se coge el momento en el que es creado el objeto Usuario

  Geters y seters de los atributos.

- 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 porque la agregación de bicis lo hacemos con el repositorio

- Bicicleta: El id 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:

- RepoUsuario: Interface que extiende JpaRepository<Usuario,Long>.

  4 modificaciones de query marcados con @Modifying y @Query que se encarga de hacer el update de nombre, apellido , estado y contraseña By Id.

- RepoEstacionBicis: Interface que extiende JpaRepository<estacionBicicletas,Long>.

  2 modificaciones de query marcados con @Modifying y @Query que se encarga de hacer el update de coordenadas y estado By Id.

- RepoBicicletas: Interface que extiende JpaRepository<Bicicleta,Long>.

  2 modificaciones de query marcados con @Modifying y @Query que se encarga de hacer el update de estado y estación By Id. Esta última,updateEstacionById conecta la entidad Bicicletas con estacionBicicletas, la clave primaria de estacionBicicletas es a su vez clave foránea de bicicleta

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 los controladores usamos métodos de los repositorios y services indistintivamente.

Controladores:

- BiciUrjcController: muestra el index

- UsuarioController: Hace los autowired del repositorioUsuario y serviceUsuario y se encarga de mostrar la pantalla principal de gestión de Usuarios que es un listado con los usuarios, mostrar los detalles de cada usuario, agregar usuarios, darlos de baja y alta y editar algunos de sus atributos haciendo uso de métodos del repositorio y de services. Los id para identificar a los usuarios dentro de la base de datos los recibimos mediante GetMapping con {id} y @ParthVariable.

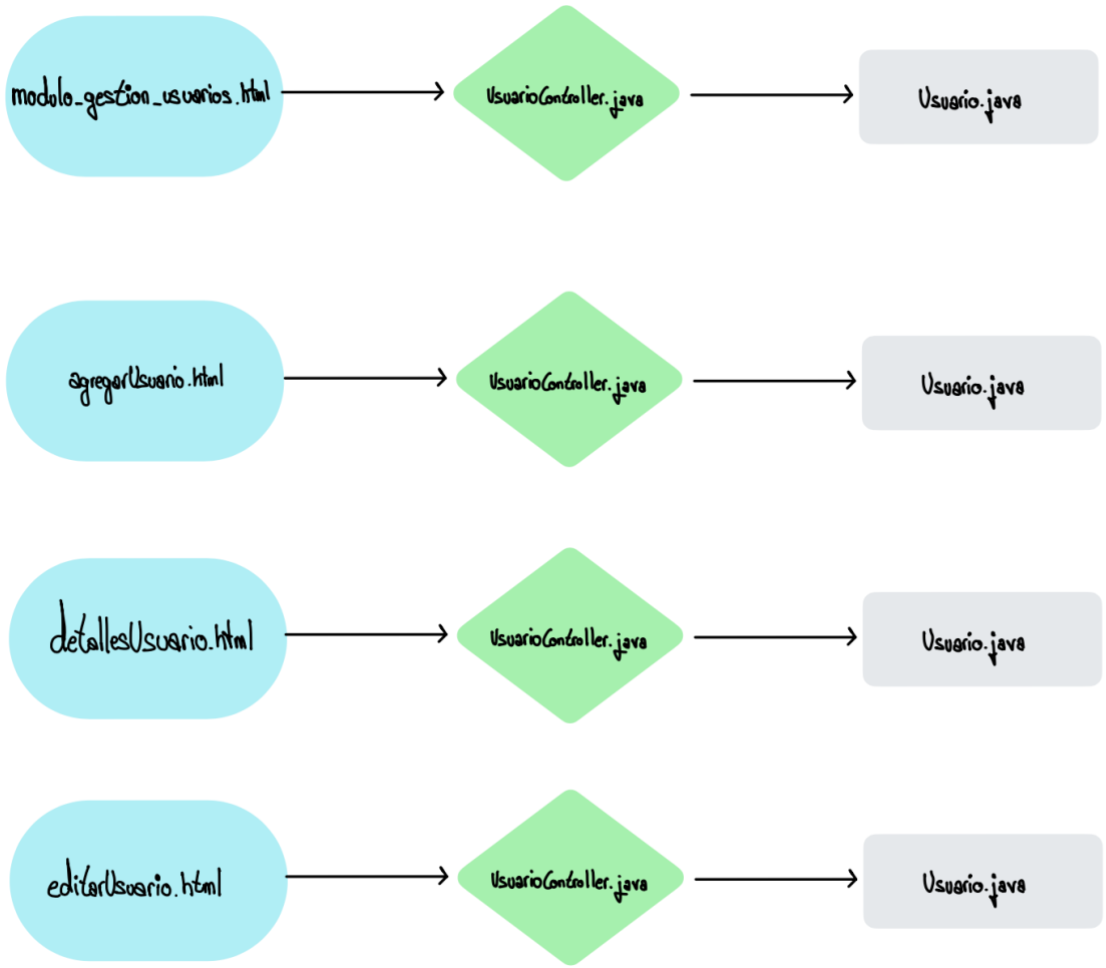
- EstacionBicicletasController: Hace los autowired de RepoEstacionBicis, RepoBicicletas, EstacionService y BicicletaService. Al estar relacionadas las entidades de bicis y de estaciones es necesario hacer uso de ambos repositorios y services.

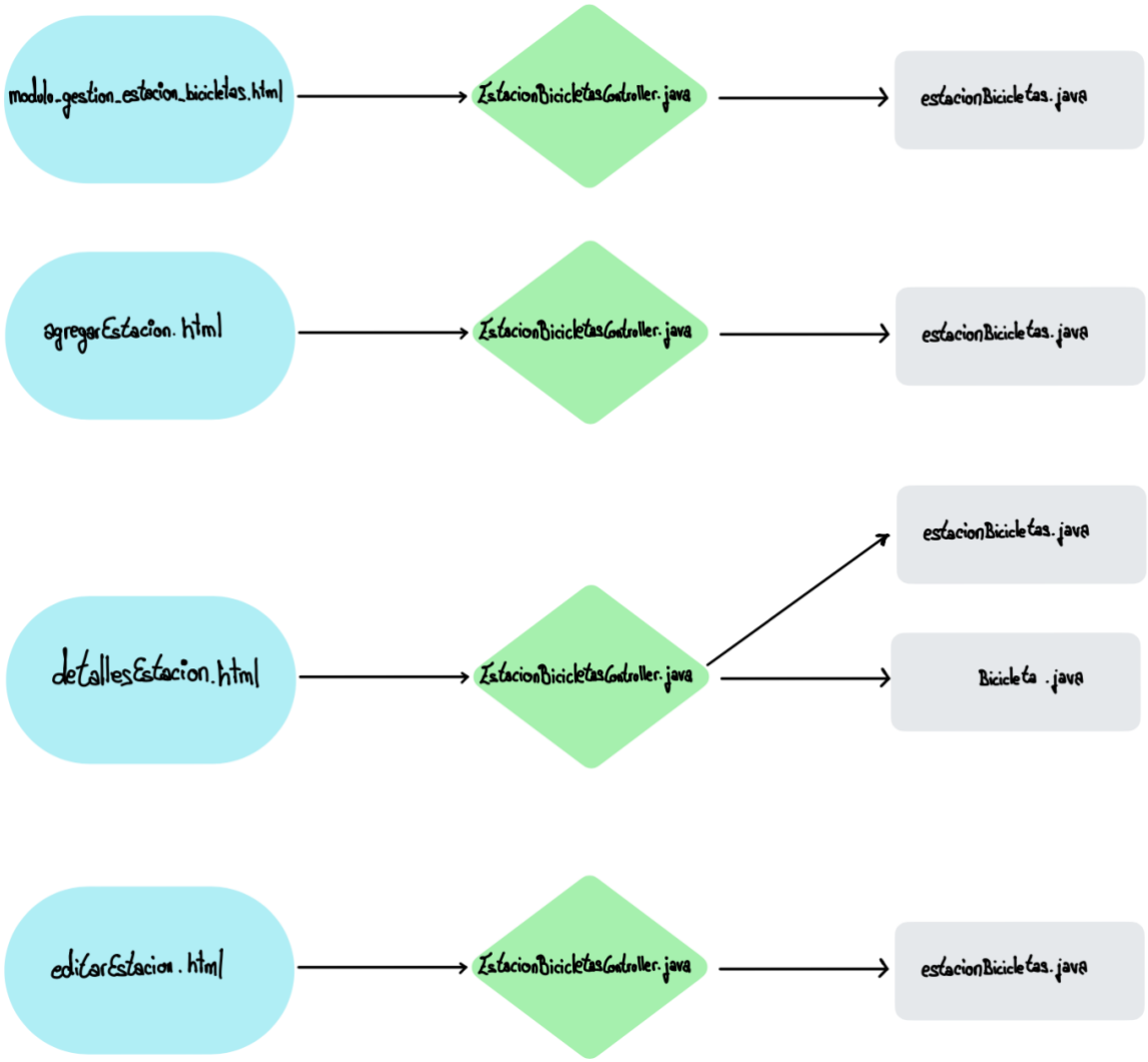
  Gestiona el modulo de gestión de estaciones de bicicletas, la pantalla principal es un listado de las estaciones con la opción de agregar más estaciones, en cada fila de la lista de estaciones da la posibilidad de ver los detalles y eso incluye editar su localización, darlos de baja y de alta, así como los detalles de las bicicletas que están asignadas a dicha estación y ubicación .

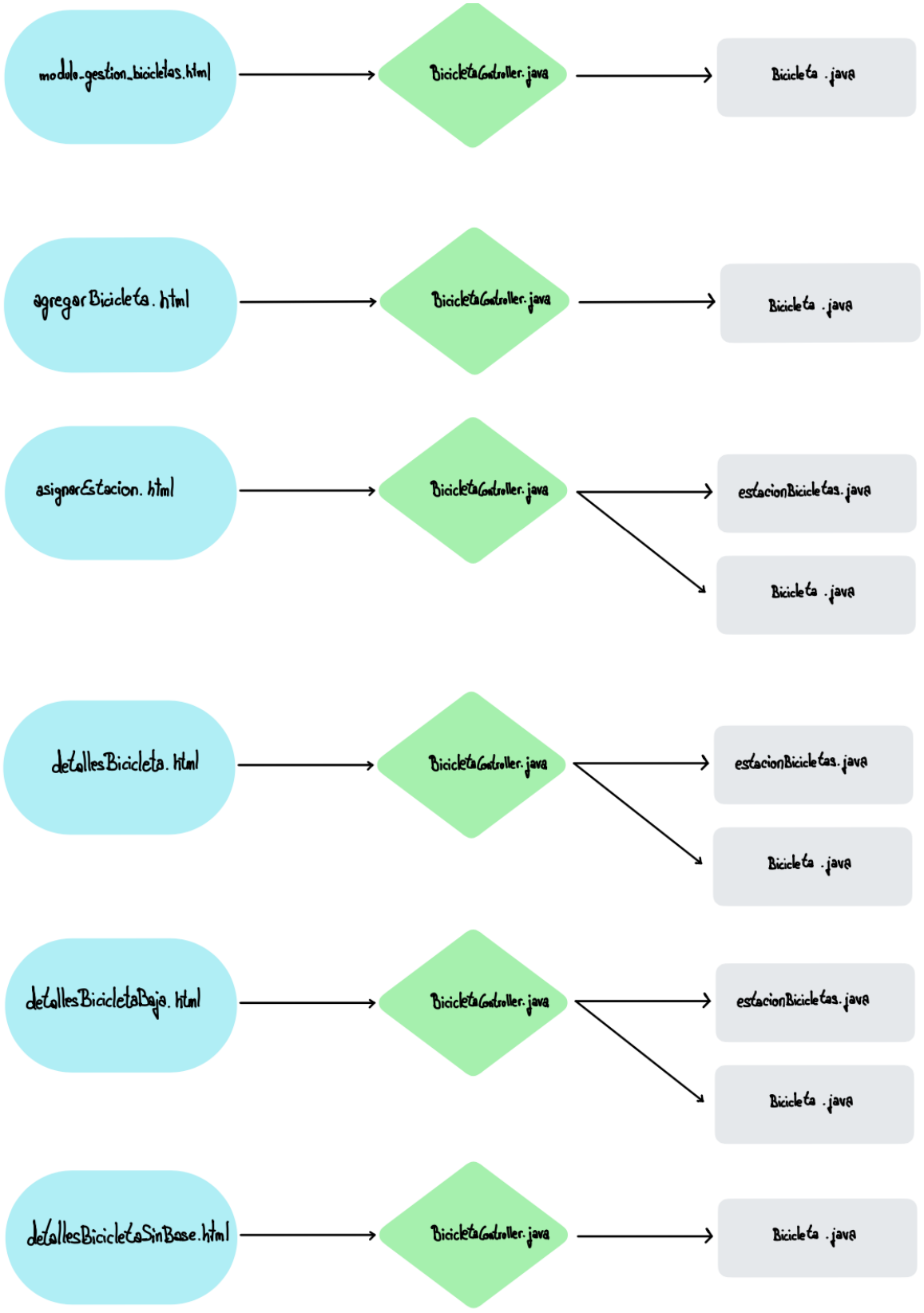
- BicicletaController: Hace los autowired de RepoEstacionBicis, RepoBicicletas, EstacionService y BicicletaService. Al estar relacionadas las entidades de bicis y de estaciones es necesario hacer uso de ambos repositorios y services.

  Gestiona el modulo de gestión de bicicletas, la pantalla principal es un listado con todas las bicicletas disponibles y la opción de ver sus detalles y agregar más bicicletas. Dentro de detalles da la opción de Asignar Base , dar de baja y el listado de transiciones de estado. Para asignar las bases a las bicis se hace uso de modificaciones de query para que la clave primaria de la estación sea la clave foránea de bicis.

Modelo – Vista - Controlador:

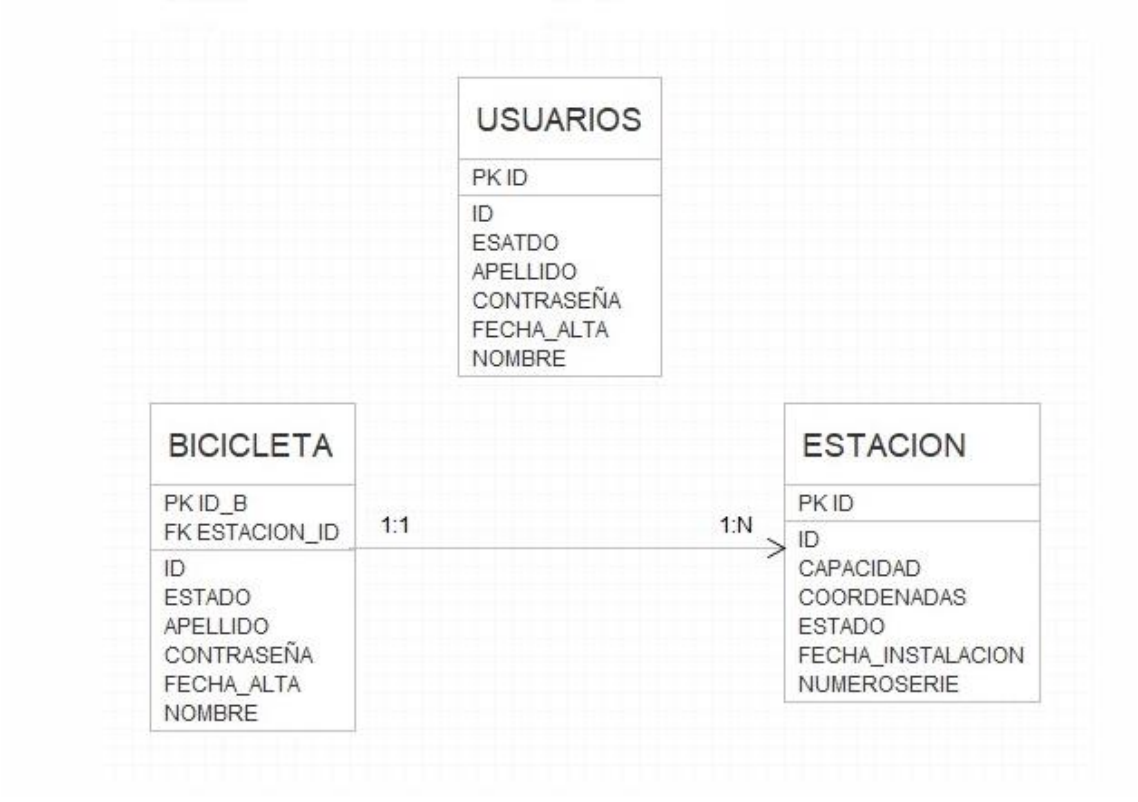


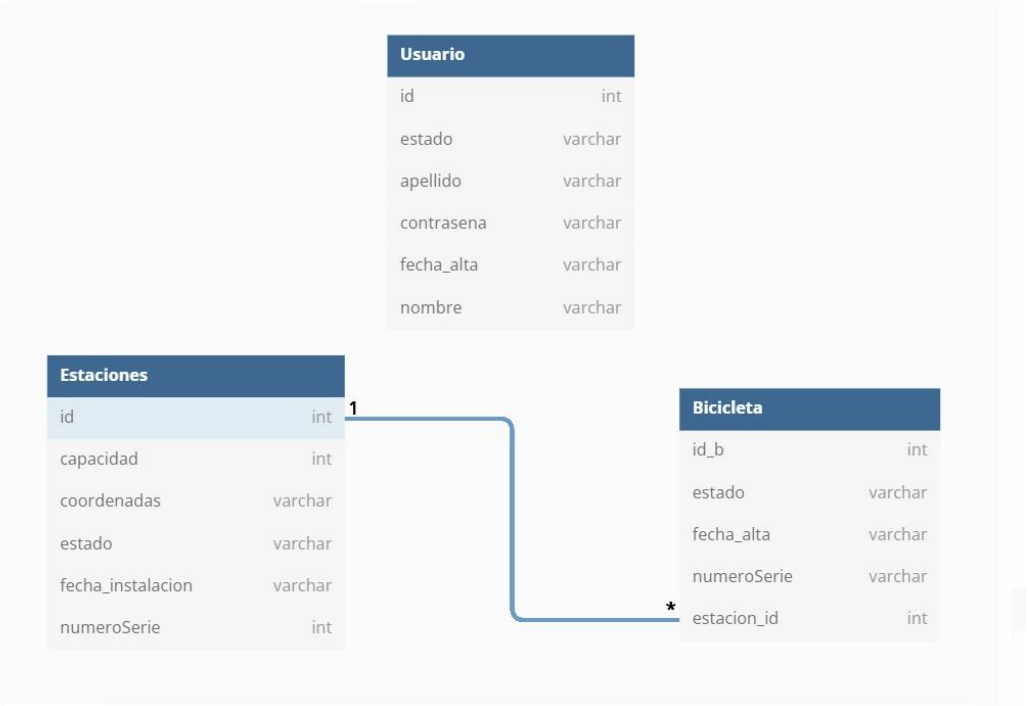

modulo_gestion_estacion_bicicletas.html
EstacionBicicletasController.java
estacionBicicletas.java
agregarEstacion.html
EstacionBicicletasController.java
estacionBicicletas.java
detallesEstacion.html
EstacionBicicletasController.java
estacionBicicletas.java
Bicicleta.java
editarEstacion.html
EstacionBicicletasController.java
estacionBicicletas.java



Daniel Requena Garrido
Raúl Rodríguez López-Rey

Modelo Conceptual Base de Datos:





Daniel Requena Garrido
Raúl Rodríguez López-Rey