|  |
| --- |
| **Práctica 1**: Desarrollo de la Aplicación BicisURJC  **Sistemas Distribuidos** – Grado en Ingeniería Informática |



Índice de **contenidos**

1. [Introducción 3](#_bookmark0)
2. [Objetivos](#_bookmark4) 3
3. [Paquete SRC](#_bookmark5) 4
   1. [Exceptions](#_bookmark6) 4
   2. [Model](#_bookmark7) 4
   3. [Repository](#_bookmark12) 5
   4. Controller 5
   5. Services………………………………………………………………………………………………………………………6
4. Paquete Resources 6
   1. [CSS](#_bookmark15) 6
   2. [Templates](#_bookmark16) 6
   3. Application.Properties 7
5. [Esquema de componentes Modelo-Vista-Controlador](#_bookmark21) 7
6. Conclusiones8

# Introducción.

A lo largo de esta primera parte del segundo cuatrimestre del curso 2021-2022 hemos trabajado una gran cantidad de contenidos en la asignatura de Sistemas Distribuidos entre los que se encuentra CSS, BootStrap, HTML, etc. Por lo tanto, para ponernos a prueba sobre dichos conocimientos nombrados anteriormente la Escuela Técnica Superior de Ingeniería Informática de la Universidad Rey Juan Carlos nos ha pedido el desarrollo de una plataforma llamada BiciUrjc, basada en un plan de movilidad sostenible para desplazarse por el área de la universitaria. La aplicación ha necesitado de varios Sistemas Software para la gestión del parque de bicicletas y la gestión de reserva del alquiler de las mismas por parte de los usuarios para ello hemos tenido en cuenta durante su desarrollo los siguientes puntos:

* + Se pretende un diseño innovador e intuitivo, mucho más agradable para el cliente.
  + En la misma línea, la aplicación será multiplataforma.
  + Buscará una eficiencia **real** y no la falsa eficiencia que las alternativas dicen ofrecer.
  + Se perseguirá una interfaz en la que destaquen la accesibilidad y la fluidez.

Durante este informe que vamos a redactar a continuación realizaremos breves descripciones sobre los diferentes procesos que hemos llevado a cabo durante la realización de nuestra aplicación **BicisUrjc.**

# Objetivos.

En la aplicación se nos han propuesto números requisitos los cuales hemos tenido que cumplir para un correcto desarrollo de esta entre los que destacamos:

* + La realización de unos determinados módulos de gestión entre los que se encuentra: el de usuarios de la plataforma, de estaciones de bicicleta, de Bicicletas y su ciclo de vida.

En cuanto a lao que se refiere a la gestión de usuario de la plataforma lo que se ha conseguido es listar a los usuarios de esta mostrando los diferentes atributos de estos especificados en el documento de la práctica, dar de alta a un usuario (añadir un usuario), modificar los usuarios exceptuando el identificador de este ya que es único y su fecha de alta y dar de baja que pasara a ser inactivo.

En segundo lugar al hablar de la gestión de las estaciones de las bicicletas los diferentes objetivos que se nos han propuesto y hemos complido son los de mostrar en la pantalla del módulo principal del módulo de gestión de Estaciones las diferentes estaciones que se van a encontrar disponible y al pinchar sobre dichas estaciones disponibles que se muestren los diferentes detalles de estas, además también se ha conseguido dentro de esta misma pantalla lograr dar de alta una estación y su borrado y modificación de todas aquellas las cuales se encentran disponibles.

Por último, en el último modelo de gestión que se refiere al ciclo de vida de las bicicletas lo que se ha conseguido es mostrar un listado de estas mostrando los diferentes detalles de estas incluido el estado en el que se encuentran y también conseguir tanto dar de alta como dar de baja una bicicleta.

# Paquete SRC.

En el paquete SRC hemos creado los diversos archivos fuente codificados para este trabajo en lenguaje Java(extensión . java). A su vez dentro de ellos encontramos los siguientes paquetes:

# 3.1 Exceptions.

Dentro de este paquete el cual se encuentra dentro de la carpeta controller hemos recogido las diferentes excepciones que hemos tratado durante la aplicación:

* + IllegalStateChange: utilizada con el fin de lanzar una excepción en el caso de que ocurra un problema con el acceso a datos.

A su vez también hemos hecho uso de IncorrectStationCapacity y IllegalStationAssociation.

# 3.2 Model.

Dentro de esta carpeta se encuentran los diferentes objetos que vamos a utilizar en nuestra aplicación:

* + Bicicleta.java: dentro de este objeto vamos a crear sus diferentes atributos y además implementaremos los gets and sets de estos y su correspondiente toString para mostrar los atributos.
  + Coords.java: este objeto va a servir para mostrar y modificar las diferentes coordenadas.
  + Estación.java: Está compuesto por objetos de la clase coord. Como ya hemos nombrado anteriormente con el fin de mostrar la ubicación de estas y además en dicho objeto se implementan tanto los gets como los sets además del toString para mostrar los atributos.
  + Usuario.java: dicho objeto va a estar compuesto por los diferentes atributos que se piden en la práctica y con sus gets y sets correspondientes, y también el toString con el fin de poder mostrar la información de dichos atributos. Además, también se hace uso de DateTimeFormatter para indicar la fecha de alta.
  + EstadoBicicleta.java: va a ser un enumerado con los diferentes estados en los que se va a encontrar una bicicleta.
  + EstadoEstación.java: va a ser un enumerado con los diferentes estados en los que se va a encontrar una estación.
  + EstadoUsuario.java: Va a ser un enumerado con los diferentes estados en los que se va a encontrar un Usuario.
  + CapsulaEstado.java: Implementa Serializable y va a servir para generar una lista de estados.

# 3.3 Repository

En esta carpeta hemos realizado los diferentes repositorios de consultas las cuales van a servir para realizar la base de datos de nuestra aplicación:

* UsuariosRepository.java: en este repositorio hemos realizado una consulta basada en coger el nombre del usuario.
* EstacionRepository: en este caso este repositorio hace referencia a las estaciones y en este caso el atributo que se coge es el número de serie.
* DataBasePopulator: este repositorio se ha realizado con el fin de crear los datos de precarga en el caso de las bicicletas hemos creado dos con el nombre de BMX y DH, en el de estación una con el nombre Hola y en el caso de usuarios hemos creado dos uno con el nombre de Juan y otro Pedro.
* BicicletasRepository: por último, este repositorio hace referencia a las bicicletas y en este caso hemos realizado dos consultas una con el número de serie o modelo y otra con el número de serie solo.

# 3.4 Controller.

En esta carpeta se encuentran los diferentes controladores para cada archivo HTML, es decir, se encuentran las diferentes funciones que van a ser llamadas dentro de los html’s que hemos creado:

* BicicletasController.java: dentro de este archivo se crea estación y bicicleta con los repositorios de Bicicletas y estaciones generando así la implementación de los programas con sus correspondientes GetMapping, en este caso se realiza: mostrarbicicletas, busquedabicicletas, añadirbicicletas, asignarbasebicicletas, modificarbicicletas y detallesbicicleta.
* EstacionesController.java: en este controlador se crea estación y bicicleta igual que en el anterior a los repositorios de Bicicletas y Estaciones y además se hace los GetMapping de los diferentes métodos que vamos a necesitar en este caso: msotrarestaciones, búsqueda, añadirestacion, eliminarestacion y detallestacions. Este controlador se parece al anterior.
* UsuariosController.java: en dicho controlador se crea usuarios a través del repositorio de usuarios y dentro de este controlador se han diversos GetMapping creando así los siguientes métodos: mostrarusuarios, búsqueda, añadirusuario, modificarusuario, eliminarusuario y detallesusuario.
* InicioController.java: simplemente se hace un GetMapping que devuelve un String que se llama inicio y lo generamos con /.

# 3.5 Services.

En esta carpeta vamos a tener tres archivos los cuales van a contener diversos métodos que van a lanzar las excepciones que han sido nombradas en el paquete Exceptipns:

* CicloVidaBicicletas.java: compuesto por dos métodos: el asignarBase, que va a comprobar el estado de la bicicleta si se encuentra con base o sin base y va a asignarla o lanzar una excepción en caso de que esta tenga base o este de baja, y luego por otro lado el cambiarEstado en el cual se van a analizar los. Diversos estados en los que se encuentre la bicicleta y dependiendo de cómo se encuentre se va a lanzar o no una IllegalStateChange.
* GestionUsuarios: es el más sencillo de los tres que hemos realizado, simplemente en él se han implementado dos métodos uno que recibe el nombre de removeUser que hace que el estado del usuario cambie a inactivo y el otro que recibe el nombre de editUser que edita el nombre y la contraseña del usuario.
* GestionEstaciones: por ultimo tenemos el Services que hace referencia a estaciones y en este caso se han implementado tres métodos: detailSt que muestra a través del toString creado en el model los diferentes atributos de una estación, addBike que va a añadir una bicicleta después de crear un objeto a partir de CicloVidaBicicletas empleando así el método de asignarBase y por último el removeBicicleta que va a eliminar una bicicleta cambiando el estado empleando el método cambiar estado después de haber creado un objeto de CicloVidaBiciletas.

# Paquete Resources.

En este paquete hemos almacenado los recursos pueda necesitar las clases java de nuestro proyecto.

# 4.1 CSS.

Hemos metido un archivo de tipo css en el cual hemos introducido los diferentes parámetros relacionados con el diseño de nuestra página web.

# 4.2 Templates.

Esta contiene una de las partes fundamentales de la práctica que es los diferentes html´s que hemos implementado para generar la página web BicisUrjc en todas hemos seguido más o menos el mismo diseño adaptándonos a lo que se nos pedía en cada caso generando así un total de 10 archivos HTML's que en lo general todos siguen los mismos parámetros de creación.

# 4.3 Application Properties.

logging.level.es.urjc=TRACE

logging.level.org.springframework.web=TRACE

spring.mustache.suffix=.html

spring.mustache.prefix=classpath:/templates/

spring.h2.console.enabled=true

spring.datasource.url=jdbc:h2:mem:bdSDPractica1

spring.profiles.active=local

# [Esquema de componentes Modelo-Vista-Controlador](#_bookmark21).

Para realizar una mejor aclaración de nuestra practica de como la hemos realizado y como hemos puesto en realización los diferentes archivos que hemos implementado en los diferentes paquetes vamos a realizar un esquema que va a estar compuesto por tres columnas una de Modelo otra de Vista y otra de Controlador en las que se van a incluir los nombres de los diferentes archivos que hemos realizado y como están realizados unos con otros.

Debemos de tener en cuenta que lo que ponemos en una misma fila es porque en esa vista se emplean tanto los modelos como los controladores nombrados:

|  |  |  |
| --- | --- | --- |
| **MODELO** | **VISTA** | **CONTROLADOR** |
| Bicicleta  Estación  EstadoBicicletas  Coords  Usuario  EsatdoUsuario  EstadoEstacion | *INICIO.HTML* | inicioController  EstacionesController  BicicletasController  UsuariosController |
| Bicicleta  Estación  EstadoBicicleta | *BUSQUEDABICICLETAS.HTML* | BicicletasController |
| Estación  Bicicleta  Coord  EstadoEstacion  EstadoBicicleta | *BUSQUEDAESTACION.HTML* | EstacionesController |
| Usuario  EstadoUsuario | *BUSQUEDAUSUARIOS.HTML* | UsuariosController |
| Bicicleta  Estación  EstadoBicicleta | *DETALLESBICICLETA.HTML* | BicicletasController |
| Estación  Bicicleta  Coord  EstadoEstacion  EstadoBicicleta | *DETALLESESTACION.HTML* | EstacionesController |
| Usuario  EstadoUsuario | *DETALLESUSUARIO.HTML* | UsuariosController |
| Bicicleta  Estación  EstadoBicicleta | *BICICLETASLIST.HTML* | BicicletasController |
| Estación  Bicicleta  Coord  EstadoEstacion  EstadoBicicleta | *ESTACIONESLIST.HTML* | EstacionesController |
| Usuario  EstadoUsuario | *USUARIOSLIST.HTM* | UsuariosController |

# [Conclusiones](#_bookmark21).