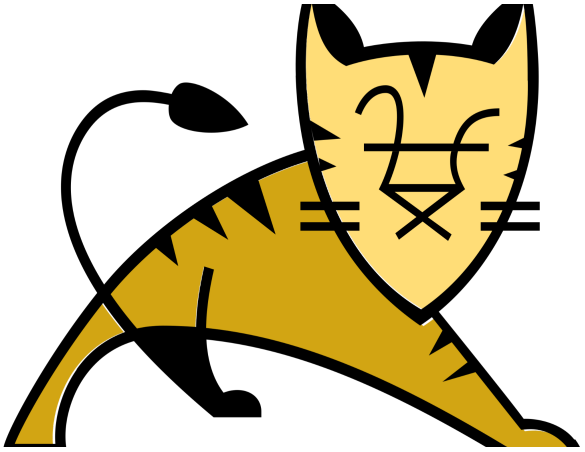
UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

Práctica 3

Desarrollo de una aplicación web



3º Ingeniería Informática
“Programación Web”
Curso 2022-2023

Componentes del grupo:

- Daniel Hinojosa Sánchez - i02hisad@uco.es
- Martín Del Río Jiménez - i02rijim@uco.es
- Juan Antonio Gálvez Jiménez - i02gajia@uco.es
- Marta Rubio Sánchez - i82rusam@uco.es
- Miguel Castro Martín - i82casmm@uco.es

Índice

Introducción

Vamos a explicar cada una de las partes que han hecho posible este proyecto, especificando de manera más detallada los aspectos más importantes de las mismas.

1.-Instrucciones de acceso

El acceso a nuestro sistema se llevará a cabo a través de los controladores y vistas de login y registro en el inicio. En la base de datos se han añadido numerosos usuarios de ambos tipos(cliente y administrador). Un ejemplo de cuentas de cada tipo son:

- i02rijim@uco.es / i02rijim : correo y contraseña de tipo administrador
- rafadiaz@uco.es / rafadiaz : correo y contraseña de tipo cliente

Además de esto en la base de datos ya están creadas distintas pistas, karts y reservas de distintos tipos, dificultades,fechas.etc. Estas servirán para probar las funcionalidades en la defensa.

2.-Tecnología usada en el desarrollo

Para el desarrollo de esta práctica hemos usado los siguientes recursos:

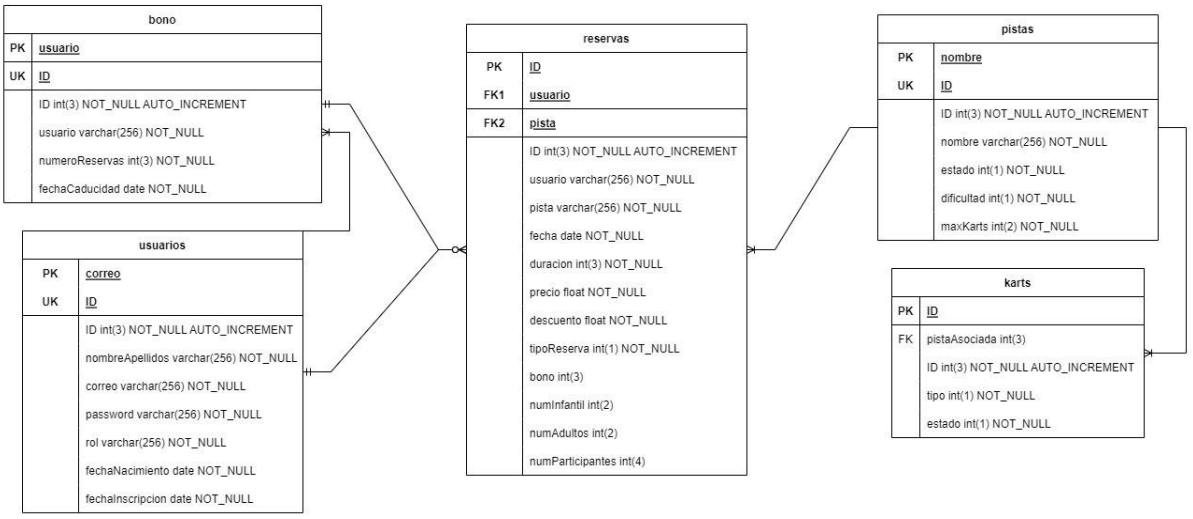
- Apache TomCat v8.5
- Draw.io
- Visual Paradigm
- JavaScript
- HTML
- CSS
- Java
- Google Documents
- Eclipse IDE for Web Developers

3.- Base de Datos

Para el desarrollo de esta práctica se ha necesitado modificar la tabla usuarios de la base de datos. A esta se le ha añadido el campo rol que determinará, mediante una variable tipo varchar(256), si el usuario es de tipo administrador o cliente.

El modelo entidad-relación de la base de datos es el siguiente:

Para el acceso a la base de datos se ha seguido usando la estructura de DAO/DTO de la práctica anterior, aunque con algunos cambios en los métodos.



A continuación se listarán los DAOs junto a sus respectivos métodos:

Clase	**DAOusuario**	
Métodos		
Guardar(DTOusuario, properties)	void	Guarda un usuario en la base de datos
Modificar(DTOusuario,correo, properties)	void	Modifica el usuario en la base de datos
Borrar(DTOusuario,properties)	boolean	Borrar el usuario de la base de datos
ListaUsuarios(properties)	ArrayList<DTOusuario>	Obtiene todos los usuarios de la base de datos
buscarUsuario(correo,properties)	DTOusuario	Busca y devuelve los datos del usuario

Clase	**DAOkart**	
Métodos		
Guardar(DTOkart, properties)	void	Guarda un kart en la base de datos
Modificar(DTOkart, id, properties)	void	Modifica el kart en la base de datos
Borrar(DTOkart,properties)	boolean	Borrar el kart de la base de datos
ListaKarts(properties)	ArrayList<DTOkart>	Obtiene todos los karts de la base de datos
buscarKartID(id,properties)	DTOkart	Busca y devuelve los datos del kart
buscarKartEstado(estado,properties)	ArrayList<DTOkart>	Busca los karts con el estado como parámetro
buscarKartPista(pistaAsociada, properties)	ArrayList<DTOkart>	Busca los karts con la pista asociada como parámetro
buscarNumeroKartsPista(pistaAsociada,properties	int	Busca los karts con la pista asociada como parámetro y devuelve el número de karts asociados

existeKartID(id, properties)	boolean	Busca el kart por id como parámetro
AsociarKart(id, pistaAsociada, properties)	void	Asocia al kart con el id con la pista que se le pasa como parámetro

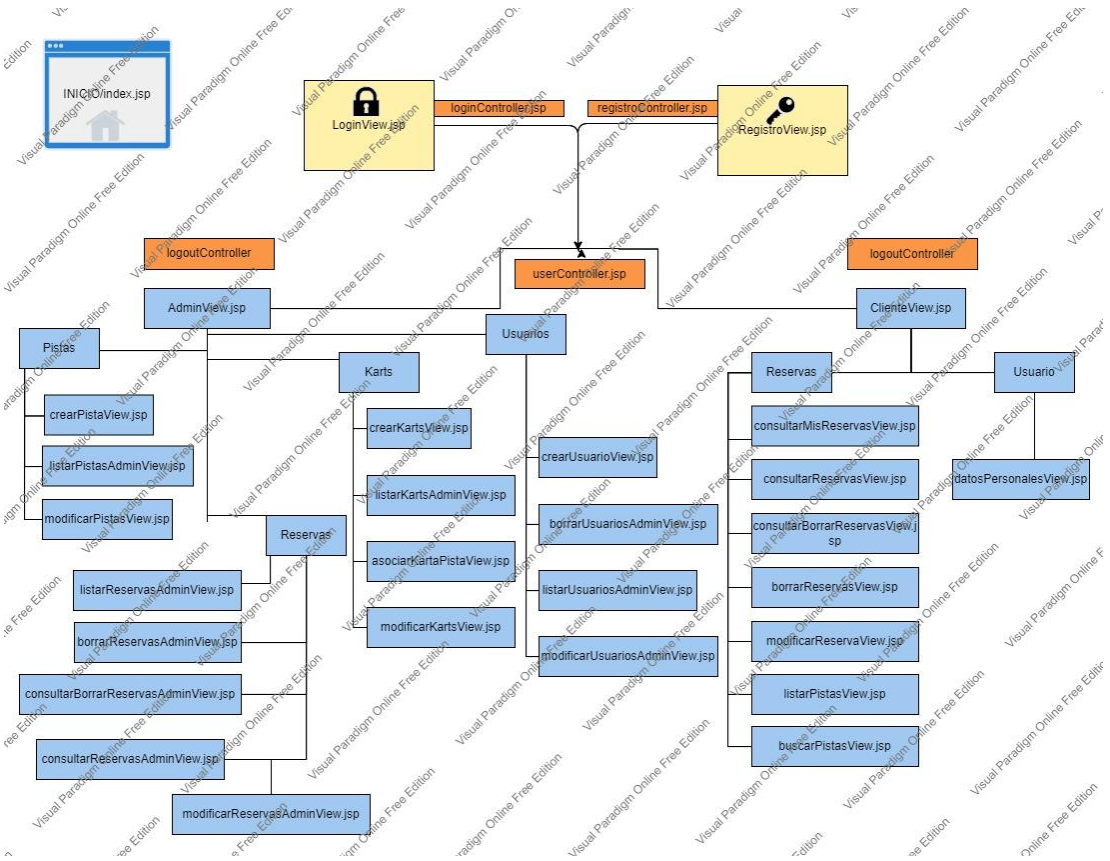
Clase	**DAOpista**	
Métodos		
Guardar(DTOpista, properties)	void	Guarda una pista en la base de datos
Modificar(DTOpista, nombre, properties)	void	Modifica la pista en la base de datos
Modificar(DTOpista, id, properties)	boolean	Modifica la pista en la base de datos
Borrar(nombre,properties)	boolean	Borra la pista de la base de datos
ListaPistas(properties)	ArrayList<DTOpista>	Obtiene todas pistas de la base de datos
buscarPistatID(id,properties)	DTOpista	Busca y devuelve los datos del kart
ListaPistasTipo(dificultad,properties)	ArrayList<DTOpista>	Busca las pistas con la dificultad como parámetro
ListaPistasKarts(numKarts, properties)	ArrayList<DTOpista>	Busca las pistas con un número de karts igual o mayor como parámetro
buscarPista(nombre,properties)	DTOpista	Busca la pista con un nombre como parámetro
buscarPista(id,properties)	DTOpista	Busca la pista con un identificador como parámetro
existePistaID(id, properties)	boolean	Busca la pista por id como parámetro

Clase	**DAOreservas**	
Métodos		
GuardarIndividual(DTOreserva***T***, properties)	void	Guarda una reserva del tipo ***T*** con el campo bono a nulo en la base de datos
GuardarBono(DTOreserva***T***, properties)	void	Guarda una reserva del tipo ***T*** con el campo bono distinto de nulo en la base de datos
Modificar(DTOreserva***T***, properties)	boolean	Modifica la reserva del tipo ***T*** en la base de datos
Borrar(id,properties)	boolean	Borra la reserva dado el id como parámetro de la base de datos
BorrarUsuarioFecha(correo,fechaReserva,properties)	boolean	Borra la reserva del tipo ***T*** dado el correo del usuario y la fecha de la reserva como parámetro de la base de datos
ListaReservas(properties)	ArrayList<DTOreserva***T***>	Obtiene todas las reservas del tipo ***T*** de la base de datos

ListaReservasUsuario(correo, properties)	ArrayList<DTOreserva***T***>	Obtiene todas las reservas del tipo ***T*** del usuario dado como parámetro de la base de datos
ListaReservasUsuarioFecha(correo,fecha, properties)	ArrayList<DTOreserva***T***>	Obtiene todas las reservas del tipo ***T*** del usuario y fecha dado como parámetro de la base de datos
ListaReservasFecha(fecha, properties)	ArrayList<DTOreserva***T***>	Obtiene todas las reservas del tipo ***T*** en la fecha dada como parámetro de la base de datos
ListaReservasBono(bono, properties)	ArrayList<DTOreserva***T***>	Obtiene todas las reservas del tipo ***T*** de un bono en la base de datos
ListaReservasPista(nombrePista, properties)	ArrayList<DTOreserva***T***>	Obtiene todas las reservas del tipo ***T*** de la pista dada como parámetro de la base de datos
ListaReservasPistaFecha(nombrePista,fecha, properties)	ArrayList<DTOreserva***T***>	Obtiene todas las reservas del tipo ***T*** de la pista y fecha dada como parámetro de la base de datos
numeroReservasFuturas(properties)	int	Obtiene el número de reservas a partir de la hora actual del sistema
getUltimoIdBono(properties)	int	Obtiene el último id de bono de la base de datos
numeroReservasBono(properties)	int	Obtiene el número de reservas en un bono de la base de datos

Anotación: T tomará el valor de Infantil,Adulto o Familiar, pues cada función deberá trabajar con una instancia de estos tipos debido al patrón factoría y la restricción en el instanciamiento de reservas.

4.-Mapa de Navegación



5.-Decisiones de diseño e implementación

A continuación vamos a mencionar distintas decisiones y características de nuestro proyecto:

- Se ha cambiado la obtención de los datos del archivo properties para la conexión con la base de datos. Si bien previamente existía el método getSQLProperties() en la clase DBConnection, se ha decidido cambiar esto. Por tanto, ahora los datos de properties le llegarán como parámetro a cada uno de los métodos de los DAO.
- Se han realizado cambios significativos en la base de datos y en la interpretación de sus datos. Anteriormente los campos como dificultad o tipo eran de tipo enum o boolean. Esto se ha cambiado por campos de tipo entero para manejarlos más fácilmente y que surjan menos problemas.
- En los métodos modificar de los DAO, como los del DAOreservas se han añadido más campos de los que había previamente y hay ciertos campos que se pueden dejar vacíos o nulos y se controla desde dentro los métodos modificar. Un ejemplo de esta utilidad es al asignar el campo bono que puede ser nulo.
- Hemos preferido usar la definiciones de las url de los servlets desde el propio servlet que desde el mapeado del web.xml pues esta última opción nos ha dado problemas.

Además de estas decisiones respecto a la implementación y el diseño, se han añadido más funcionalidades tanto a administrador como a cliente de los que se especifican en el enunciado. Se han añadido:

- Modificar las reservas para admin
- Consultar pistas, reservas y karts para admin
- Consultar las reservas propias del cliente con sesión iniciada

6 - Bibliografía

https://www.w3schools.com
https://getbootstrap.com
https://es.stackoverflow.com
https://www.youtube.com/watch?v=oZa2Ut8u2S0
https://www.jose-aguilar.com/blog/boton-para-mostrar-contrasena-en-un-campo/
https://github.com/MarioBerrios/PW-project
https://docs.oracle.com/javase/8/docs