



# TennisLab

Mohamed Asidah & Rocío Palao

	1
<b>Descripción</b>	<b>3</b>
<b>Diagrama de Clases</b>	<b>3</b>
<b>Propuesta de solución</b>	<b>3</b>
<b>Diseño</b>	<b>4</b>
<b>Clases</b>	<b>4</b>
Modelos	4
Usuarios Clientes	4
Usuarios Trabajadores	4
Máquinas Encordadora	5
Máquinas Personalizadora	5
Pedidos Tarea Encordado	5
Pedidos Tarea Personalizado	5
Pedidos Tarea Adquisición	5
Pedidos Productos	6
Pedidos Tareas	6
Pedidos Pedidos	6
Mapeadores (Exposed)	7
Mapeador de Usuarios	7
Mapeador de Productos	7
Mapeador de Pedidos	7
Mapeador de Máquinas	7
Controladores	8
Controlador de Clientes	8
Controlador Trabajadores	8
Controlador Máquinas	8
Controlador Productos	8
Controlador Tareas	9
Controlador Pedidos	9
<b>Realización Ficheros Json</b>	<b>10</b>
<b>Testing</b>	<b>10</b>
<b>Realización de Clases</b>	<b>11</b>
Entidades	11
Mapeadores	11
Excepciones	11
Modelos	11
Repositorios	11
Controladores	11

Utils	11
Vista	11
<b>Realización de Clases Test</b>	<b>12</b>
Repositorios	12
Controladores	12

## Descripción

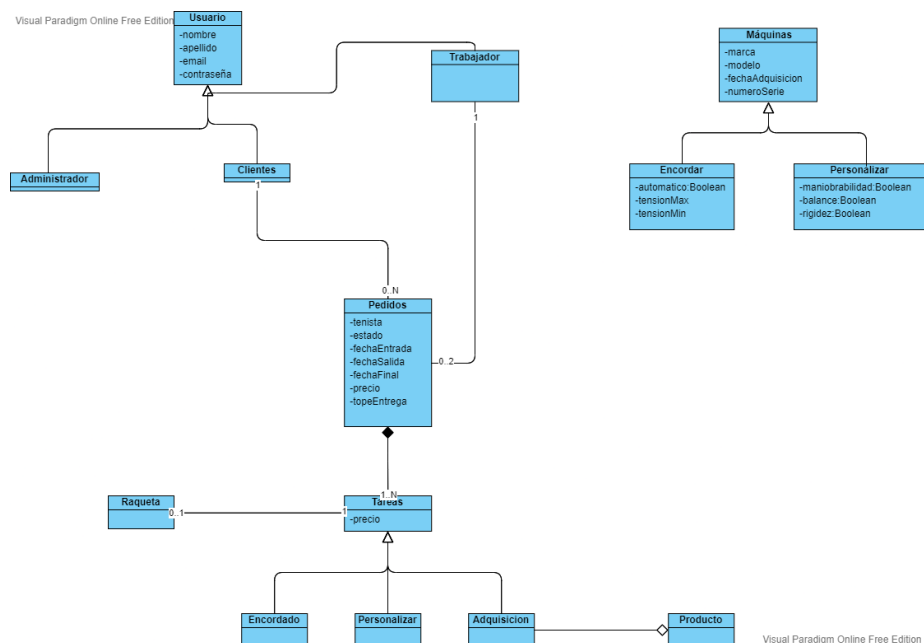
Proyecto sobre la administración de una tienda de tenis, que ofrece servicios y productos.

Realizado con BDD, Exposed y Hibernate

Nuestro programa debe:

- Realizar CRUD de las clases necesarias.
- Guardar en ficheros Json los datos.

## Diagrama de Clases



## Propuesta de solución

Para la solución dividiremos el problema en varias partes.

1. CRUD Usuarios.
2. CRUD Máquinas.
3. CRUD Pedidos.
4. CRUD Productos.
5. Escritura en ficheros Json.

## Diseño

El diseño utilizado en este programa es Modelo-Vista-Controlador.

- Modelo: Clases encargadas del almacenamiento de los datos.
- Controlador: Tenemos varios, encargados del control del programa, controlan los repositorios y los modelos.
- Vista: Encargado de mostrar al usuario los datos.

## Clases










Las clases usadas en este proyecto están divididas en:

### Modelos

Encargados del almacenamiento de los datos.











#### Usuarios Clientes

Almacena en base de datos los datos de clientes que usarán nuestra aplicación

  data	Cliente
 password	String
 pedidos	Pedido?
 email	String
 nombre	String
 apellido	String
 id	Int?
 uuid	UUID?

#### Usuarios Trabajadores

Almacena en la base de datos los trabajadores que hay, pueden ser administradores

  data	Trabajador
 password	String
 email	String
 nombre	String
 administrador	Boolean
 apellido	String
 id	Int?
 disponible	Boolean
 uuid	UUID?

### Máquinas Encordadora

Almacena en la base de datos las máquinas de tipo encordadoras

data Encordador	
marca	String
modelo	String
tensionMinima	Int
tensionMaxima	Int
fechaAdquisicion	LocalDate
automatico	TipoMaquina
disponible	Boolean
uuid	UUID?

### Máquinas Personalizadora

Almacena en la base de datos las máquinas de tipo personalizadoras

data Personalizadora	
marca	String
balance	Boolean
modelo	String
rigidez	Boolean
fechaAdquisicion	LocalDate
id	Int?
disponible	Boolean
maniobrabilidad	Boolean
uuid	UUID?

### Pedidos Tarea Encordado

Almacena en la base de datos las tareas de tipo encordado, necesarias para realizar tareas.

data TareaEncordado	
NNudos	Int
tensionVertical	Int
tensionHorizontal	Int
cordajeVertical	Producto
cordajeHorizontal	Producto

### Pedidos Tarea Personalizado

Almacena en la base de datos las tareas de tipo personalizado, necesarias para realizar tareas.

data TareaPersonalizado	
balance	Float
pero	Int
rigidez	Int

### Pedidos Tarea Adquisición

Almacena en la base de datos las tareas de tipo adquisición, necesarias para realizar tareas.

data TareaAdquisicion	
productos	ListaProductos
precio	Float

## Pedidos Productos

Almacena en la base de datos los productos, necesarios para realizar la tarea de adquisición.

data Producto	
tipo	TipoProduct
marca	String
precio	Float
modelo	String
stock	Int
id	Int?
uuid	UUID?

## Pedidos Tareas

Almacena en la base de datos, la tarea a realizar, necesaria para realizar el pedido. Dependiendo del tipo de tarea guardado, necesitamos tarea adquisición, tarea encordado o tarea personalizado. Los datos específicos sobre lo que va a realizar la tarea se guardan en la descripción como un objeto embebido.

data Tarea	
descripcion	String
idTrabajador	Trabajador?
tipoTarea	TipoTarea
id	Int?
precio	Long
idMaquina	UUID?
disponible	Boolean
uuid	UUID?

## Pedidos Pedidos

Almacena en la base de datos, el pedido del cliente. Contiene una lista de tareas que componen el pedido.

data Pedido	
topeEntrega	LocalDate
fechaEntrada	LocalDate
cliente	Cliente?
fechaFinal	LocalDate?
precioTotal	Float
tareas	ArrayList <Tarea>
estado	Estado
fechaSalida	LocalDate?
uuid	UUID?

## Mapeadores (Exposed)

### Mapeador de Usuarios

Encargado de convertir las entidades de la BDD de Clientes y Trabajadores a modelos

ClienteMapperKt	TrabajadorMapperKt
fromClienteDaoToCliente(ClienteDAO) Cliente	fromTrabajadorDaoToTrabajador(TrabajadorDAO) Trabajador

### Mapeador de Productos

Encargado de convertir la entidad de Producto a su modelo.

ProductoMapperKt
fromProductoDaoToProducto(ProductoDAO) Producto

### Mapeador de Pedidos

Encargado de convertir las entidades de la BDD de Pedido y Tarea a sus modelos.

PedidosMapperKt
fromPedidoDaoToPedido(PedidoDAO) Pedido
fromTareaDaoToTarea(TareaDAO) Tarea

### Mapeador de Máquinas

Encargado de convertir las entidades de la BDD de Encordado y Personalizado a sus modelos.

MaquinaMapperKt
fromEncordadorDaoToEncordador(EncordadorDAO) Encordador
fromMaquinaDaoToMaquina(MaquinaDAO) Maquina
fromPersonalizadorDaoToPersonalizadora(PersonalizadorDAO) lizadora
fromPersonalizadorToPersonalizadorDao(Personalizadora) alizadorDAO








## Controladores

### Controlador de Clientes

Encargado de controlar la lógica del programa que necesitan los clientes.






Para su control entre los modelos, entidades, mapeador y repositorio.

ClientesController		
	<code>addCliente(Cliente)</code>	Cliente
	<code>getClienteByEmailAndPassword(String, String)</code>	Cliente?
	<code>deleteCliente(Cliente)</code>	Boolean
	<code>updateCliente(Cliente)</code>	Unit
	<code>getClienteByUUID(UUID)</code>	Cliente

### Controlador Trabajadores

Encargado de controlar la lógica del programa que necesitan los trabajadores.









Para su control entre los modelos, entidades, mapeador y repositorio.

TrabajadoresController		
	<code>deleteTrabajador(Trabajador)</code>	Boolean
	<code>updateTrabajador(Trabajador)</code>	Unit
	<code>getTrabajadorByUUID(UUID)</code>	Trabajador
	<code>addTrabajador(Trabajador)</code>	Trabajador
	<code>getTrabajadorByEmailAndPassword(String, String)</code>	Trabajador?

### Controlador Máquinas

Encargado de controlar la lógica del programa que necesitan las máquinas.





Para su control entre los modelos, entidades, mapeadores y repositorios.

MaquinasController		
	<code>getPersonalizadoraByUUID(UUID)</code>	Personalizadora
	<code>deletePersonalizadora(Personalizadora)</code>	Boolean
	<code>deleteEncordadora(Encordador)</code>	Boolean
	<code>addPersonalizadora(Personalizadora)</code>	Personalizadora
	<code>addEncordadora(Encordador)</code>	Encordador
	<code>updateEncordadora(Encordador)</code>	Encordador
	<code>getEncordadoraByUUID(UUID)</code>	Encordador
	<code>updatePersonalizadora(Personalizadora)</code>	Personalizadora

### Controlador Productos

Encargado de controlar la lógica del programa que necesitan los productos.

Para su control entre los modelos, entidades, mapeador y repositorio.

ProductosController		
	<code>getProductoByUUID(UUID)</code>	Producto
	<code>updateProducto(Producto)</code>	Producto
	<code>deleteProducto(Producto)</code>	Boolean
	<code>addProducto(Producto)</code>	Producto

### Controlador Tareas

Encargado de controlar la lógica del programa que necesitan las tareas.

Para su control entre los modelos, entidades, mapeador y repositorio.

TareasController		
#	addTarea(Tarea)	Tarea
#	addPedidoId(Tarea, Pedido)	Tarea
#	getTareaByUUID(UUID)	Tarea?
#	updateTarea(Tarea)	Tarea
#	deleteTarea(Tarea)	Boolean

### Controlador Pedidos

Encargado de controlar la lógica del programa que necesitan los pedidos.

Para su control entre los modelos, entidades, mapeador y repositorio.

PedidosController		
#	updatePedido(Pedido)	Pedido
#	addPedido(Pedido)	Pedido
#	deletePedido(Pedido)	Boolean
#	getPedidoByUUID(UUID)	Pedido?

## Realización Ficheros Json

Para esto, creamos unas clases que únicamente contienen las listas necesarias y que son serializables.

Estas clases, únicamente contienen una lista de sus respectivos modelos, escritos en cada clase.



Para este listado, tuvimos que crear una clase, ya que necesitábamos que contuviese varios datos de distintas partes del programa.



Cada fichero se escribe con la serialización de Json de Kotlin. Además se lanzan todos los guardados de ficheros a la vez en paralelo, por lo que les hemos añadido un mutex por si acceden a la misma clase de fichero varios a la vez.

## Testing

Testing de nuestro programa realizado con JUnit5 y Mockkito.

MockKito usado para testear el comportamiento de los controladores.

JUnit5 utilizado para testear de manera unitaria nuestras clases.

## Tecnologías Utilizadas

**Google guava** : Para la realización de encriptación de contraseña de los usuarios.

**Hibernate/JPA**: utilizado para gestionar las bases de datos relacionales. Es un ORM que facilita el trabajo a la hora de desarrollar aplicaciones con bases de datos disminuyendo la cantidad de código que hay que escribir y mejorando el rendimiento del programa.

**Kotlin Exposed**: una librería para SQL hecha para Kotlin. Una de sus principales ventajas es que otorga una API para acceder y manipular bases de datos. También cabe notar que es una librería que se integra fácilmente con otras librerías de kotlin, haciendo que sea más fácil y eficaz usarlo para hacer aplicaciones orientadas a bases de datos en kotlin.

**Kotlin Serialization**: herramienta utilizada para serializar las clases de Kotlin a JSON en este caso para así realizar los logs de ejecución, y los objetos embebidos en otras clases.

## Realización de Clases

### Entidades

- Usuarios → Rocío
- Maquinas → Mohamed
- Productos → Rocío
- Pedido y Tarea → Mohamed

### Mapeadores

Mohamed

### Excepciones

Rocío

### Modelos

- Listados → Rocío
- Usuarios → Rocío
- Maquinas → Rocío y Mohamed
- Pedidos → Mohamed

## Repositorios

- Usuarios → Rocío
- Máquinas → Mohamed y Rocío
- Productos → Rocío
- Pedidos → Mohamed

## Controladores

- Controladores de Usuarios → Rocío
- Controlador de Máquinas → Rocío
- Controlador de Productos → Rocío
- Controlador de Tareas → Mohamed
- Controlador de Pedidos → Mohamed

## Utils

Rocío

## Vista

Rocío y Mohamed

## Realización de Clases Test

### Repositorios

- Usuarios → Rocío
- Productos → Rocío
- Pedidos → Mohamed
- Máquinas → Mohamed y Rocío

### Controladores

- Trabajadores → Rocío
- Clientes → Rocío
- Máquinas → Rocío y Mohamed
- Productos → Rocío
- Tareas → Rocío y Mohamed
- Pedidos → Rocío y Mohamed