



# **MEMORIA DE ARQUITECTURA**

**2018/2019**

**CONTENIDO**

Arquitectura Multicapa ..... 2

Patrones De Diseño..... 3

## ARQUITECTURA MULTICAPA

Nuestro proyecto se ha desarrollado siguiendo una **arquitectura multicapa** y en él hemos empleado los patrones fundamentales de ésta. Por un lado, la capa de presentación gestiona las interfaces con el usuario. Por su parte, la capa de negocio es la encargada de la lógica y el procesamiento del programa. Además, está la capa de integración que comunica al resto de componentes con recursos y sistemas externos, como por ejemplo la base de datos.

La **capa de Presentación** consta de varias clases:



**Main**, **Controller**, **VentanaPrincipal**, **VentanaIni**, **VentanaRegistro**, **VentanaPerfilRestaurante**, **VentanaPerfilCliente** y **VentanaEditarPerfilRestaurante**.

La **capa de Negocio** está compuesta por las siguientes clases:



**TransferUsuario**, de la que heredan **TransferRestaurante** y **TransferCliente**, **Pedido**, **Comida**, **Mesa**. Y también consta de la *interfaz* **SA** implementada por la clase **SAImp**.

La **capa de Integración** contiene dos interfaces:





**DAORestaurante** y **DAOCiente** con las respectivas clases que las implementan, **DAORestaurante** y **DAOCiente**.


## PATRONES DE DISEÑO

Los patrones de diseño que hemos utilizado a la hora de programar CómoComo son los siguientes:


El Patrón **TRANSFER** muy útil para pasar datos con múltiples atributos de una vez desde el cliente al servidor. Nos ha servido para pasar datos entre las distintas capas que componen nuestro proyecto.

El Patrón **DAO** para poder acceder a la base de datos  en la que tenemos almacenada toda la información. Con él hemos conseguido que el acceso a los datos sea independiente de su procesamiento.

El Patrón **SA** (Servicio de Aplicación)  contiene toda la lógica del negocio permitiendo así una centralización de ésta.

El **MODELO VISTA CONTROLADOR** ha sido de vital importancia a la hora de programar lo respectivo a la capa de presentación. Nos ha permitido almacenar la funcionalidad básica y los datos en el modelo. 

Por su parte, es la vista la que se encarga de almacenar y recoger la información del usuario, y los controladores son los intermediarios entre las vistas y el modelo.

Por último, el Patrón **SINGLETON** lo hemos utilizado para tener una única Instancia del controlador, proporcionando así un  único punto de acceso a ella.