

# DOCUMENTACIÓN TÉCNICA Y FUNCIONAL

Proyecto gestión dual

# Índice

1.FUNCIÓN Y TECNOLOGÍA UTILIZADA.....	1
1.1Funcionamiento.....	1
1.2 Tecnologías e IDES.....	1
2.Modelo de datos.....	1
2.1 Relaciones.....	1
3. Aplicación (BACK-END).....	2
3.1 Capa de controladores.....	2
3.2 Capa de persistencias.....	4
3.3 Capa de servicios.....	5
4.Aplicación (FRONT-END).....	5

# 1.Función y Tecnología utilizada

## 1.1Funcionamiento

La aplicación es desarrollada para solventar un problema en la gestión de la formación dual, siendo capaz no solo de devolvernos la información que busquemos de manera rápida y sencilla, sino también de crear nuevos registros y almacenarlos en la base de datos que posteriormente atacaremos desde la parte del cliente.

## 1.2 Tecnologías e IDES

El proyecto está construido bajo la arquitectura **Maven**, que nos permite gestionar las dependencias de manera rápida y sencilla.

La aplicación hace uso del framework de **Spring**, implementando así los siguientes modulos:

- Spring Core.
- Spring Data.
- Spring Boot.
- Spring Mvc.

Contiene un sistema de log, proporcionado por **LogBack**, el cual guarda la traza de ejecución del programa en un archivo, llamado logs.log .

Como entorno de desarrollo integrado usados, tenemos **Eclipse** para desarrollar las clases en lenguaje Java, y **Visual Studio Code** para realizar las modificaciones en los htmls y css.

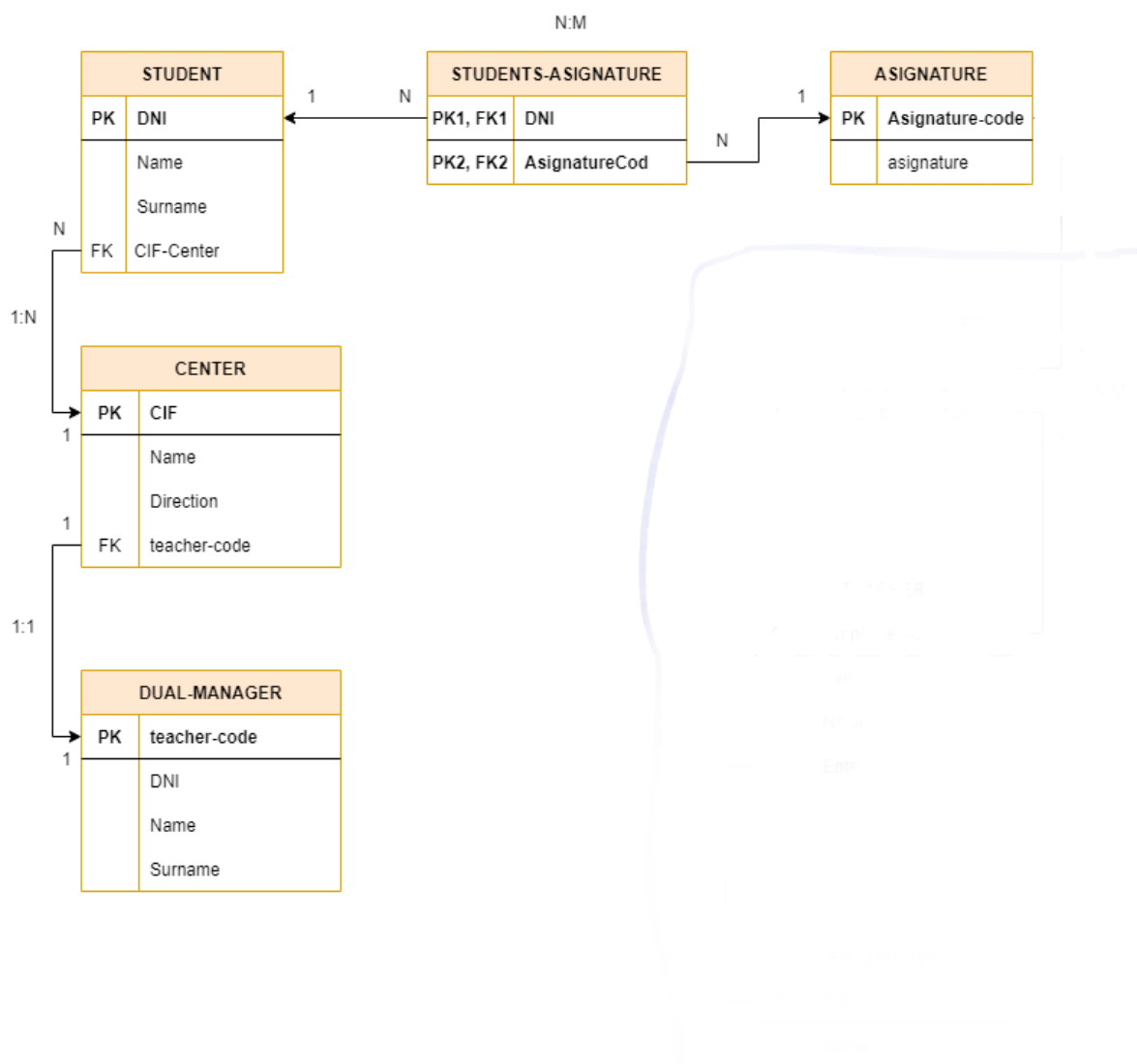
## 2.Modelo de datos

La base de datos contiene 4 tablas:

- Signature.
- Students.
- Center.
- DualManager.

### 2.1 Relaciones

Las relaciones entre las tablas son las siguientes:



## 3. Aplicación (BACK-END)

La aplicación consta de 3 capas bien definidas:

### 3.1 Capa de controladores

Se encarga de gestionar el envío de datos entre la parte front y la parte del back.

En ella nos encontraremos 3 clases:

**ControlerException**, se encarga de la gestión de excepciones en la capa de los controladores, es decir, si un controlador genera una excepción, entrará en único método que esta clase contiene. Devolviendo en la pantalla del navegador un texto (Algo está mal...).

```
package com.nttdata.nttdata_sevilla_eclipse_dualgestion_ejerciciofinal.controller;

import javax.servlet.http.HttpServletRequest;

/**
 * Proyecto Dual Gestion.
 * Controlador de excepciones global.
 * @author Javier Jiménez Montesino.
 */

@ControllerAdvice
public class ControlerException {

    /* Logger */
    final static Logger LOG = LoggerFactory.getLogger(ControlerException.class);

    @ExceptionHandler(Exception.class)
    public @ResponseBody String handleException(HttpServletRequest req, Exception e) {
        LOG.error(
            "Ha entrado en el metodo handleException de la clase ControlerException, a ocurrido un fallo en la aplicacion.");
        // Respuesta.
        final String responseBody = "Algo está mal...";
        System.out.println(responseBody);

        return responseBody;
    }
}
```

**ControlerWelcome**, este controlador está a la escucha de peticiones en la ruta `home` dicha clase contiene un método el cual se ejecutará una vez realices una petición a la ruta `home/welcome`, devolviéndote así el html con la pantalla principal.

```
@Controller
@RequestMapping("/home/")
public class ControlerWelcome {

    /* Logger */
    final static Logger LOG = LoggerFactory.getLogger(ControlerWelcome.class);

    /**
     * respuesta hacia vista la vista index.
     * Capta cualquier solicitud a /home/welcome
     * @return String
     */
    @GetMapping("/welcome")
    public String oneIndexView() {
        LOG.info("Entrada al metodo oneIndexView de la clase ControlerWelcome");
        // Respuesta.
        final String responseBody = "index";
        System.out.println(responseBody);
        LOG.info("Salida del metodo oneIndexView de la clase ControlerWelcome");
        return responseBody;
    }
}
```

**ControlerButton**, este controlador está a la escucha de peticiones en la ruta `home/welcome`, dentro de esta clase tenemos varios métodos, que a fin de cuenta lo único que hacen es devolverte vistas, dependiendo del botón al que clickes e intercambiar información entre dichas vistas mediante el model.

```
@GetMapping("/searchCenters")
public String oneCenterSeachView() {
    LOG.info("Entrada en el metodo oneCenterSeachView de la clase ControlesButtons");
    // Respuesta.
    final String responseBody = "centerSearch";
    System.out.println(responseBody);
    LOG.info("Saliendo del metodo oneCenterSeachView de la clase ControlesButtons");
    return responseBody;
}
```

## 3.2 Capa de persistencias

Esta capa contiene los beans que representan una tabla en la base de datos y sus interfaces e implementaciones directas. Esta capa está protegida, ya que solo se puede acceder a ella mediante la capa de servicios, siendo llamada desde la misma.



La clase AbstractEntity es creada para que todas las clases hijos de ella tengan sus 2 atributos, por tanto cada bean que representa una tabla, extiende de él.

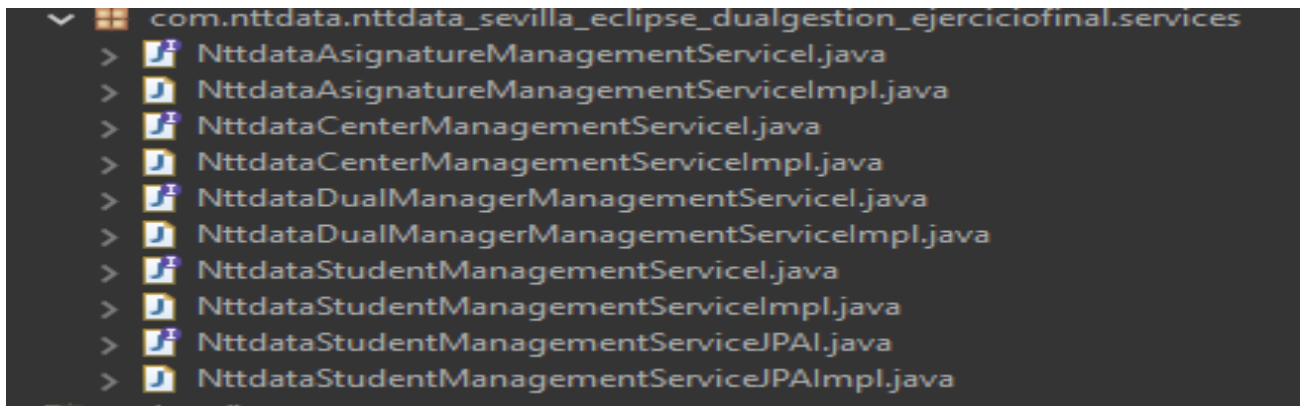
Las interfaces de cada bean representativo extiende de JpaRepository, que contiene todos los métodos cruds básico, esto lo podemos hacer gracias al módulo de Spring - Spring Data.

Se ha optado a realizar una interfaz propia la cual define un método que a la hora de implementarse contiene una consulta JPA criteria, para crearla en la interfaz propia del bean y tener que desarrollar todos los métodos que nos ofrece el repositorio de JPA y no perdiendo así el fuerte ofrecido por Spring Data.

### 3.3 Capa de servicios

La capa de servicio contiene las interfaces y las clases que implementan a esas interfaces, las cuales son llamadas desde la main para realizar cualquier operación CRUD o consulta.

En la clase main siempre se consumen los servicios y nunca tocaremos las interfaces e implementaciones que pertenezcan a la capa persistence.

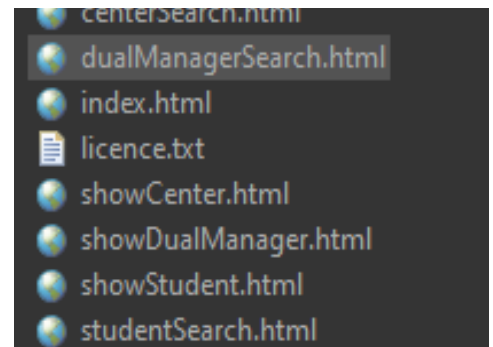


Dentro de cada implementación comprobaremos la nulabilidad de los parámetros recibidos desde la main, añadiendo así una capa mas de seguridad.

## 4.Aplicación (FRONT-END)

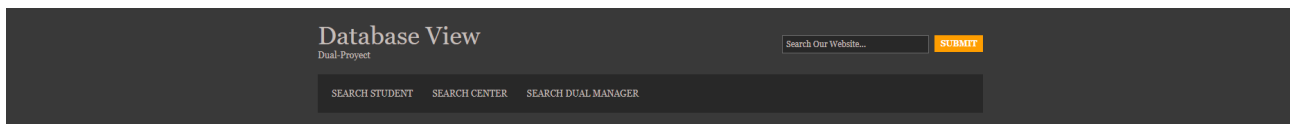
La parte front de la aplicación consta de 7 vistas en total.

La vista index, es la vista principal, la cual contiene los botones que te redirigirán a las otras vistas mediante los controladores nombrados anteriormente. Las vistas acabadas en Search, son las que reciben los datos para realizar la búsqueda, y las show son las que los muestra.





INDEX.HTML(Contiene los botones de redirección).



The screenshot shows the top part of a web application. It has a dark background. On the left, the text 'Database View' is displayed in a light color, with 'Dual-Project' underneath it. To the right of this text is a search bar with the placeholder text 'Search Our Website...' and an orange 'SUBMIT' button. Below the search bar, there is a horizontal menu with three items: 'SEARCH STUDENT', 'SEARCH CENTER', and 'SEARCH DUAL MANAGER'.

Copyright © 2018 - All Rights Reserved - Domain Name

Hecho por : Javier Jiménez Montesinos

```
<ul>  
  <li><a href="welcome/searchStudents">Search Student</a></li>  
  <li><a href="welcome/searchCenters">Search Center</a></li>  
  <li><a href="welcome/searchDualManagers">Search Dual Manager</a></li>  
</ul>
```

SEARCH.HTML(Envia información a la vista show enlazada).



The screenshot shows a form titled 'Search Student.' in a dark theme. Below the title, there is a label 'Student DNI' followed by a text input field containing the value '200458998'. At the bottom of the form is a green 'Search' button.

```
<form th:action="@{./studentFound}" method="POST">  
  <div class="form-group">  
    <label for="SearchStudent">Student DNI</label>  
    <input type="text" required="required" class="form-control" id="SearchStudent" name="DNI" placeholder="200458998">  
  </div>  
  <button type="submit" name="btnSearchStudent" class="btn btn-success">Search</button>  
</form>
```

SHOW.HTML(Muestra los datos que hemos elegido de la consulta realizada por detrás.



The screenshot shows a form titled 'Alumno encontrado.' in a dark theme. It displays the details of a student found. There are four input fields, each with a label to its left: 'Name' with the value 'Javier', 'DNI' with the value '200618998', and 'Center' with the value 'Hnos Machado'. The fourth field is empty.

```
<form>
  <div class="form-group">
    <label for="inpCarRegistration">Name</label>
    <input type="text" disabled="disabled" class="form-control" id="inpTcarRegistration" th:value="{student.name}">
  </div>
  <div class="form-group">
    <label for="inpMake">DNI</label>
    <input type="text" disabled="disabled" class="form-control" id="inpTmake" th:value="{student.DNI}">
  </div>
  <div class="form-group">
    <label for="inpModel">Center</label>
    <input type="text" disabled="disabled" class="form-control" id="inpTmodel" th:value="{student.center.nameCenter}">
  </div>
</form>
```