PROYECTOS DE PROGRAMACIÓN

DANIEL GONZÁLEZ CERVIÑO

23.01.2021

INTRODUCCIÓN

Este es el portafolio del trabajo realizado en la asignatura **proyectos de programación** del **grado de ingeniería informática** de la **Universidad Internacional de Valencia**.

A lo largo de este proyecto hemos desarrollado un sistema que permite:

1. Gestionar usuarios, centros de trabajo y trabajos.
2. Asignar trabajos a centros.
3. Simular el paso del tiempo.

DISCLAIMER

No todas las funcionalidades han sido completadas.

Al ser un grupo de una única persona y al haber sido tan ambicioso en cuanto a la tecnología utilizada no ha sido posible completar estas partes.

1. Han quedado pendiente implementar los requisitos de seguridad del tipo limitar cierta funcionalidad para administradores de centro y usuarios normales.
2. Han quedado parcialmente sin finalizar la información actual del sistema.

TECNOLOGÍA

En este apartado queremos exponer cuales son las herramientas y librerías más importantes que hemos utilizado en este proyecto.

Hemos querido ser muy ambiciosos en cuanto a la selección de tecnologías que hemos usado, pero era una oportunidad para adquirir conocimientos que tengan la mayor aplicación posible en el mundo profesional.

No obstante consultamos primero al profesor y aceptó la posibilidad de utilizar Spring Boot.

SPRING BOOT

Hemos decidido utilizar un **framework de desarrollo web** muy popular en la actualidad **Spring Boot**[1].

Spring Boot promete hacer fácil el desarrollo de aplicaciones basadas en Spring[2].

Spring Boot es un framework web y por lo tanto se ejecutará a través de un servidor de aplicaciones como por ejemplo tomcat[3] y se visualizará a través de un navegador web.

Spring Boot permite empaquetar el proyecto junto con todas sus dependencias en un fichero sólo jar.

MAVEN

Spring Boot tiene una serie de dependencias que es necesario instalar pero está pensado para ser utilizado junto a un gestor de dependencias como Maven[4] o Gradle[5]. En nuestro caso hemos decidido utilizar Maven.

Esto nos permitirá resolver e instalar las dependencias de una forma sencilla.

THYMELEAF

Como sistema de plantillas hemos decidido usar Thymeleaf[6]. Es un sistema basado en HTML5, fácil de usar y que se utiliza habitualmente junto a Spring Boot.

BOOTSTRAP + ADMIN LTE

Queríamos dar al proyecto un aspecto agradable a la vista sin que esto supusiera dedicarle un montón de tiempo.

Por eso nos decidimos a utilizar Bootstrap[7] y Admin LTE[8].

---

[1] https://spring.io/projects/spring-boot
[2] https://spring.io/
[3] http://tomcat.apache.org/
[4] https://maven.apache.org/
[5] https://gradle.org/
[6] https://www.thymeleaf.org/
[7] https://getbootstrap.com/
[8] https://adminlte.io/

Bootstrap es un framework css y Admin LTE es una plantilla basada en este framework.

En la figura 2 puede verse el resultado.

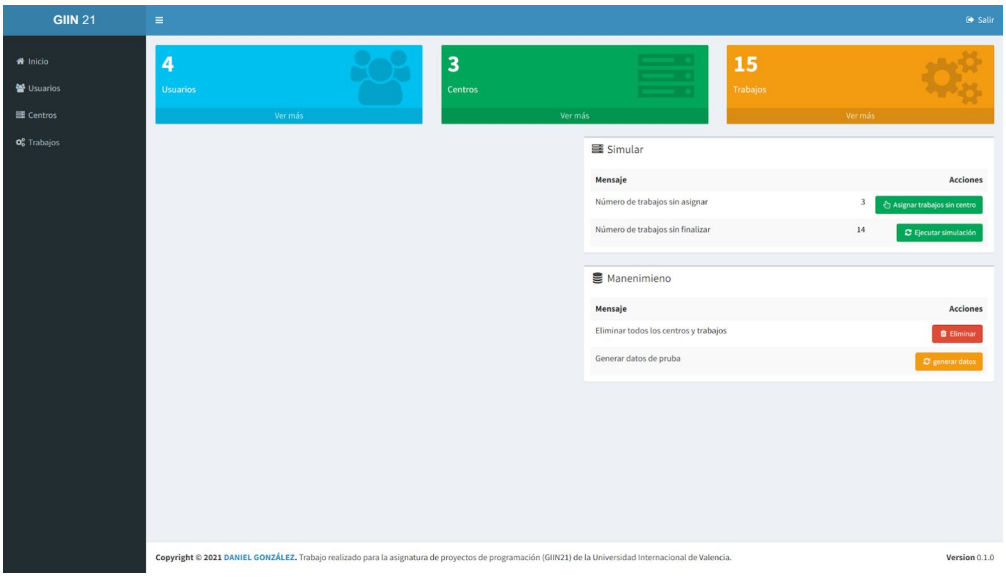

Figura 2: panel de control

INTELLIJ IDEA ULTIMATE

Intellij idea es uno de los entornos de desarrollo más utilizados en el entorno profesional.

Nos decidimos por este IDE por la integración que tiene con Spring Boot.

VERSIONES

En este apartado están disponibles las versiones de los componentes más importantes del proyecto.

Componente	Versión
Java	11
Spring Boot	2.4
Maven	3.6
Thymeleaf	3.0

Bootstrap	3.7
Intellij Idea Ultimate	2020.3
Ubuntu	20.04

METODOLOGÍA

Para este proyecto hemos decidido utilizar una RAD[9] o desarrollo rápido de aplicaciones.

En contraposición a las metodologías tradicionales, en este tipo de metodologías no se realizan fases secuenciales en las que cada una depende de la anterior.

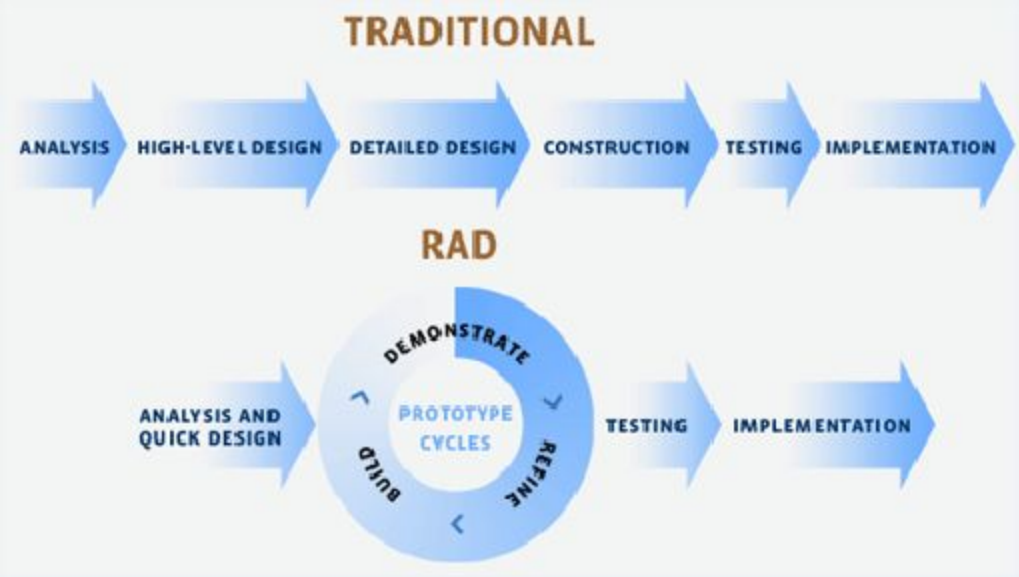

Figura 1: no se ha podido determinar la fuente original.

En lugar de eso se realizan iteraciones que contienen todas las fases y que entregan una pequeña porción de software, para continuar con una nueva iteración en la que se añaden nuevos requisitos.

CASOS DE USO

Un diagrama de casos de uso permite identificar de forma gráfica los diferentes actores así como la funcionalidad que estará disponible para cada uno de ellos.

[9] https://es.wikipedia.org/wiki/Desarrollo_r%C3%A1pido_de_aplicaciones

Aunque ya hemos explicado en el disclaimer que no toda la funcionalidad ha sido completada, en este análisis aparece documentada el análisis completo de los requisitos.

Nota: el diagrama de clases está disponible en la carpeta doc del proyecto.

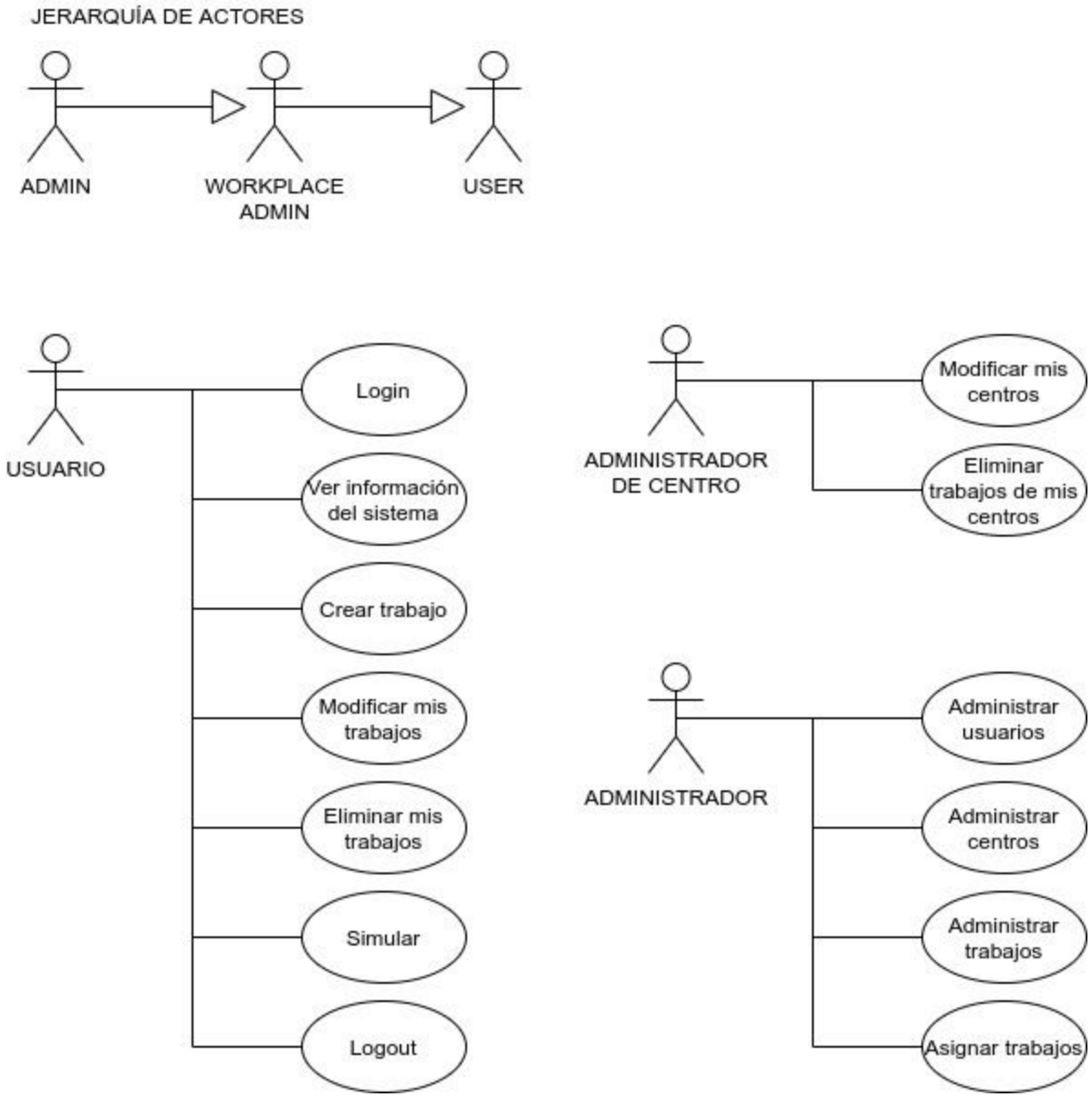

Figura 2: casos de uso

JERARQUIA DE ACTORES

En los requisitos se identifican tres actores: usuario, administrador de centro y administrador.

USUARIO

Los usuarios de la aplicación pueden hacer login y logout, consultar la información del sistema, crear trabajos, y editar / eliminar sus trabajos.

En los requisitos no se define que usuarios pueden realizar la simulación, por lo que hemos indicado que todos los usuarios podrán realizar la simulación del paso del tiempo.

ADMINISTRADOR DE CENTRO

El administrador de centro puede administrar sus centros así como eliminar trabajos de sus centros.

ADMINISTRADOR

El administrador podrá administrar usuarios, administrar centros, administrar trabajos y asignar trabajos a centros mediante el criterio que aparece en los requisitos.

DATOS DE ACCESO

Estos son los principales datos de acceso que se han utilizado para el desarrollo de este proyecto.

Elemento	url	usuario	password
Código fuente	https://github.com/desarrolla2/viu_22_proyectos_de_programacion	~	~
Base de datos	giin21.cjgn2iccqrmq.us-east-2.rds.amazonaws.com:3306	viu_giin_21	viu_giin_21

INSTALACIÓN Y EJECUCIÓN

En este apartado queremos describir los puntos necesarios para instalar y ejecutar el proyecto.

DESCARGAR EL CÓDIGO FUENTE

El control de versiones del código se ha realizado mediante el sistema de control de versiones Git[10].

Además puesto una copia del repositorio disponible en github[11]. Para obtener una copia local del mismo, tan sólo es necesario ejecutar.

```
git clone git@github.com:desarrolla2/viu_22_proyectos_de_programacion.git
```

DESCARGAR LAS DEPENDENCIAS Y COMPILAR

Para instalar las dependencias y compilar el proyecto tan sólo es necesario ejecutar.

```
./mvnw install
```

EJECUTAR

Para ejecutar el proyecto tan sólo es necesario ejecutar.

```
java -jar target/giin21-0.0.1-SNAPSHOT.jar
```

DESCARGAR

Para facilitar que el proyecto pueda probarse sin la necesidad de instalar dependencias y/o compilar hemos generado una copia del ejecutable. Para obtenerla tan sólo es necesario ejecutar.

[10] https://git-scm.com/
[11] https://github.com/

```
wget https://www.dropbox.com/s/uxkkv53ng7ilb8o/giin21-0.0.1-SNAPSHOT.jar
```

DEMO

En esta apartado queremos mostrar una demo del proyecto, para facilitar que se pueda comprobar y entender el trabajo realizado.

LOGIN

Cuando iniciamos la aplicación será necesario autenticarse con uno de los usuarios de la base de datos.

Utilizaremos el usuario admin / admin para acceder.

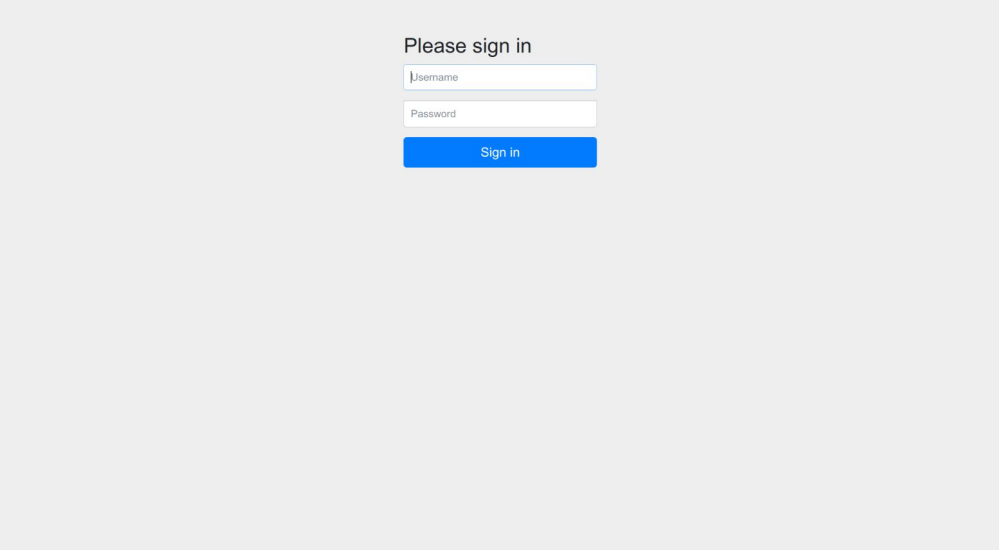

Figura 3: login de usuario

PANEL DE CONTROL

En el panel de control se muestran los datos más importantes de la aplicación, así como un acceso directo a las funcionalidades más importantes.

Ver demo http://nimb.ws/WnT8MG

GESTIÓN DE USUARIOS

En el módulo de usuarios podremos dar de alta, editar y eliminar usuarios.

Ver demo http://nimb.ws/zeLmBU

GESTIÓN DE CENTROS DE TRABAJO

En el módulo de centros de trabajo podremos dar de alta, editar y eliminar centros de trabajo.

Ver demo http://nimb.ws/FAgGxn

GESTIÓN DE TRABAJOS

En el módulo de trabajos podremos dar de alta, editar y eliminar trabajos.

Ver demo http://nimb.ws/lnZpTJ

MANTENIMIENTO DE LA BASE DE DATOS

Permite limpiar la base de datos y generar datos de prueba.

Ver demo http://nimb.ws/TiCctw

ASIGNAR TRABAJOS

En el módulo de asignación de trabajos, el sistema asignará los trabajos según los requisitos que se indican en la actividad, hasta que todos los centros tengan su cola completa.

Ver demo http://nimb.ws/fRETv3

SIMULAR

En el módulo de simulación de paso del tiempo, el sistema simulará como se cargan los trabajos en los centros, como estos van ejecutando los trabajos hasta que estos son completados y finalmente como los trabajos son descargados y comienza la carga de un nuevo proceso.

Ver demo http://nimb.ws/dyZqE4

SALIR

Este módulo nos permite cerrar nuestra sesión.

Ver demo http://nimb.ws/JpfcE3

ARQUITECTURA DE LA APLICACIÓN

La aplicación contiene principalmente las siguientes capas.

CONTROLADORES

Los controladores son los responsables de recibir las peticiones del navegador.

Habitualmente un controlador realiza una o más llamadas a servicios para obtener la información que necesita.

A continuación utiliza el sistema de plantillas para construir un documento HTML.

Finalmente envía una respuesta al navegador con el documento que ha generado y las cabeceras oportunas.

PLANTILLAS

Las plantillas son las responsables de generar la capa de presentación del proyecto al usuario. En este caso documentos HTML.

MODELOS

Los modelos son las representaciones en objetos de las entidades en la base de datos.

A través de una técnica conocida como Mapeo Objeto Relacional[12] nos permite abstraernos de la base de datos que se está utilizando.

REPOSITORIOS

Un repositorio es la representación en objetos de las tablas de la base de datos.

Como hemos visto en el apartado de modelos, nos permite abstraernos de la base de datos que se está utilizando.

---

[12] https://es.wikipedia.org/wiki/Asignaci%C3%B3n_objeto-relacional

SERVICIOS

La capa de servicios es la que contiene la lógica de negocio.

Habitualmente son llamados por los controladores y utilizan los repositorios y los modelos para realizar sus funciones.

OBJETOS DE TRANSFERENCIA DE DATOS

Trabajar con modelos ofrece determinados riesgos.

Por ejemplo un servicio podría devolver una lista de usuarios y estos ser modificados en algún punto fuera de este. Para evitar que esto suceda los servicios devuelven un tipo de objetos que se denominan DTOs o Data Transfer Object[13] y que son representaciones de los modelos.

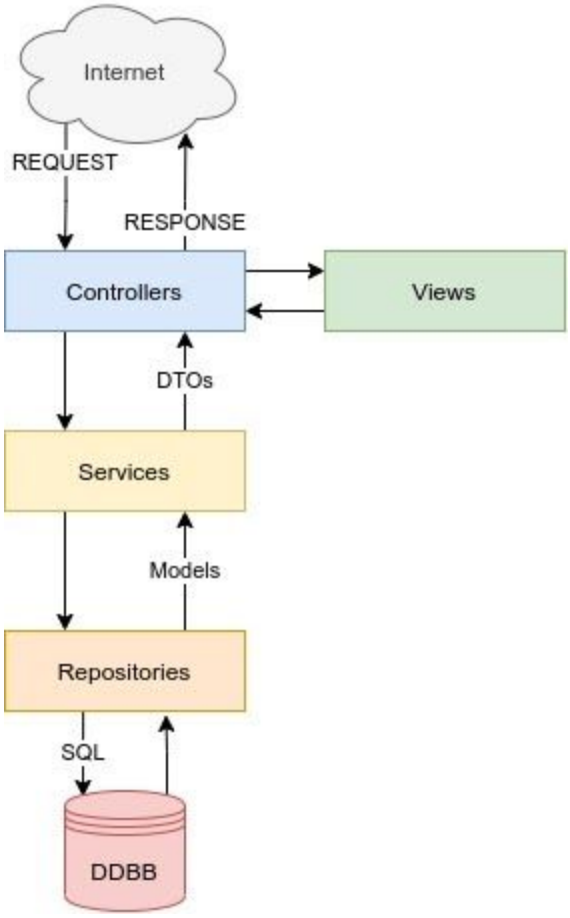

Figura 4: arquitectura de la aplicación

[13] https://es.wikipedia.org/wiki/Objeto_de_transferencia_de_datos

DIAGRAMA DE CLASES

Un diagrama de clases UML es un diagrama que describe la estructura de clases de un sistema mostrando tanto sus métodos y atributos cómo las relaciones entre ellas.

Para la realización de este diagrama de clases hemos utilizado una utilizad del entorno de desarrollo que nos permite crear el diagrama a partir de la ingeniería inversa, lo cual es una herramienta muy útil para entender qué clases tiene nuestro proyecto y cuales son las relaciones entre ellas.

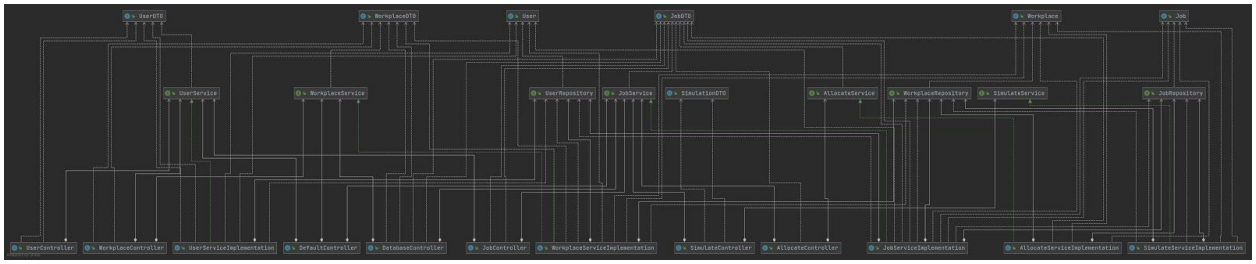

Figura 5: diagrama de clases UML.

Como es muy difícil que se puedan apreciar las clases, hemos añadido una imagen en alta resolución en la carpeta doc del proyecto y que puede descargarse directamente desde este enlace: https://www.dropbox.com/s/otryf9qlx5yrfy5/giin21.jpg?dl=0

JAVADOC

Javadoc es una utilidad para la generación de documentación HTML a partir de código fuente Java. Javadoc es el estándar de la industria para documentar clases de Java. La mayoría de los IDEs los generan automáticamente.

Hemos generado el Javadoc utilizando una utilidad del entorno de desarrollo que permite generar fácilmente.

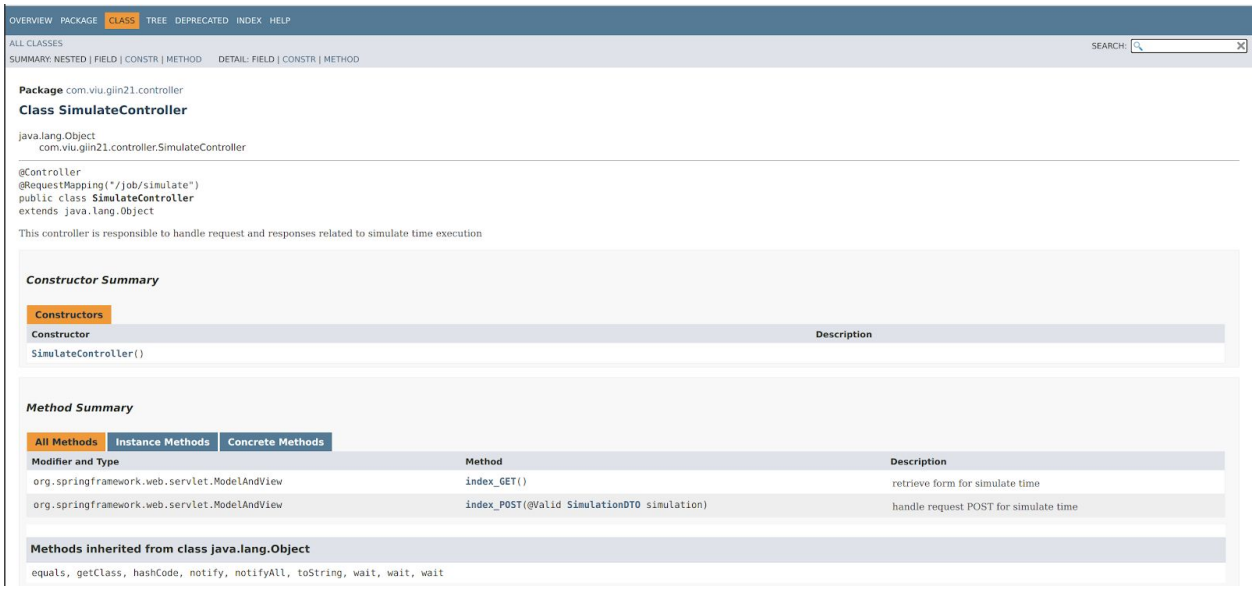

Figura 6: documentación de la clase SimulateController

Todo el javadoc está disponible en la carpeta doc/javadoc del repositorio.

BASE DE DATOS

ESQUEMA

La base de datos se genera automáticamente a partir de los modelos que contiene el proyecto.

Si se indica una conexión a una base de datos y se ejecuta el proyecto hibernate se encargará de crear las tablas necesarias para una correcta ejecución.

MEJORAR LA VELOCIDAD

Al ser una base de datos remota el proyecto funciona muy lento. Si se quieren hacer pruebas con cierta fluidez se recomienda cambiar por una base de datos local. Para ello tan sólo es necesario editar el fichero *src/main/resources/application.properties*

Este paso requiere compilar el proyecto.

PLAN DE CALIDAD

Todo el código de esta aplicación se ha desarrollado siguiendo los criterios descritos en el libro clean code[14] de Robert C Martin.

Veamos alguno de los puntos más importantes.

USO INTERFACES

Se han definido interfaces para los **servicios** y **repositorios**. Esto nos permitiría modificar el proyecto cambiando unas clases por otras que cumplieran la misma interfaz.

INYECCIÓN DE DEPENDENCIAS

La inyección de dependencias[15] es una técnica que permite delegar a una clase, normalmente conocida como contenedor de dependencias la instanciación y configuración de las clases del proyecto.

De esta forma se reduce el acoplamiento entre clases y permite sustituir fácilmente unas clases por otras siempre que cumplan una misma interfaz.

NOMBRES

Cuando se desarrolla software es muy importante dedicar todo el tiempo necesario para elegir buenos nombres.

Entre otras hemos seguido las siguientes reglas:

1. Evitar el uso de contracciones.
2. Se ha seguido una estrategia de nombres consistentes.
3. Utilizar el contexto para dar significado a un nombre.
4. Los nombres de variables corresponden a aquello que contienen.
5. Los nombres de métodos ayudan a entender cómo deben usarse.

COMENTARIOS

Una de las reglas que más controversia suelen generar es la regla de evitar el uso de comentarios.

---

[14] https://www.oreilly.com/library/view/clean-code-a/9780136083238/
[15] https://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias

El autor propone que un comentario, suele significar que el código no es lo suficientemente legible por sí mismo, y que debería sustituirse el uso de comentarios, por una nueva redacción del código que no los haga necesarios.

Nota: es por esto que no se ha incluido el javadoc.

ORGANIZACIÓN

Todo el código está organizado de la misma forma, siguiendo unas reglas de estilo constantes.

LICENCIA

Este proyecto se ha desarrollado bajo la licencia MIT. Puede encontrarse la licencia completa en la carpeta raíz del repositorio.

MEJORAS

Se han identificado las siguientes mejoras que se habrían realizado de haber tenido suficiente tiempo disponible.

TEST

Una de las características que proporcionan al código robustez es una batería de test automáticos, que permita probar que todo funciona correctamente con cada cambio que se haga.

INTERFAZ

La interfaz de usuario tiene ciertas mejoras que no hubieran sido difíciles de aplicar.

Por ejemplo

- Utilizar iconos y colores para representar los estados de los centros de trabajo y de los procesos.
- Formatear los números para facilitar su lectura.
    - Ej: 1000000 => 1,000,000
- Mostrar mensajes “flash”, informando que ha ocurrido.
    - Ej: “Se ha creado el usuario correctamente.”

- Paginación y filtros sobre los listados.

CONCLUSIONES

Ha sido un proyecto muy interesante en el que hemos podido aprender a trabajar con un montón de tecnologías con las que no teníamos experiencia.

Sin embargo, al ser un equipo de una sola persona no se ha podido completar toda la funcionalidad.

AGRADECIMIENTOS

A nuestras familias por comprender que dediquemos nuestras noches y fines de semana a la obtención del título del grado de esta profesión que tanto nos apasiona.

Al profesor de la asignatura por permitirnos realizar este proyecto.