

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

16-6-2020

TAREAS APP

PRÁCTICA 2

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and sweep upwards and to the right.

Luisa Cogollos Perucho y Rocío Sosa Ruiz
UFV, TECNOLOGÍAS AVANZADAS DE PROGRAMACIÓN

ÍNDICE

1. INTRODUCCIÓN	2
2. DESARROLLO	3
2.1. CONTROL DE VERSIONES - GIT	3
2.2. DIAGRAMA DE CLASES	4
2.3. DIAGRAMA DE FLUJO	7
2.4. DISEÑO DEL SOFTWARE DE LA APLICACIÓN	8
2.5. INTERFAZ DE USUARIO	10
2.6. BASE DE DATOS H2	14
2.7. REPORTE DE SONARQUBE	16
3. CONCLUSIONES	20
4. BIBLIOGRAFÍA	21

1. INTRODUCCIÓN

En este documento se muestran las explicaciones del desarrollo de la práctica 2 de la asignatura de Tecnologías Avanzadas de Programación.

El objetivo de dicha práctica es comprender y afianzar todos los conocimientos vistos en la asignatura, que nos permitirán estudiar las distintas estrategias avanzadas de programación orientadas al uso de marcos de trabajo y profundizando en la Arquitectura Orientada a Servicios, aprovechando para resolver problemas reales de manejo de información en la vida empresarial.

La práctica consiste en el desarrollo de una aplicación en la que se ponga de manifiesto el aprendizaje de los principios de diseño de software y de la programación orientada a objetos, así como las técnicas de refactoring y la arquitectura de una aplicación web cliente-servidor.

La aplicación de tareas que consistirá en un “GTD: Getting Things Done”, desarrollará un frontend y un backend independientes, cuya comunicación se realizara por HTTP. (Desarrollo de una API. JSON.)

Requisitos de Usuario:

- Se podrán crear tareas con una descripción
- Se podrá asignar prioridad a una tarea. (Default, low, medium, high)
- Se podrá asignar una fecha límite “deadline” a las tareas.
- Las tareas se podrán marcar como completadas
- Las tareas se podrán organizar en diferentes listas.

Requisitos aplicación:

- El lenguaje de programación a utilizar será Java.
- El frontend se realizará con el framework Java Vaadin
- El backend se realizará con el framework Java Spring.
- El backend dispondrá de una base de datos MySQL para persistir la información.
- El gestor de dependencias a utilizar en ambos proyectos será Maven.

2. DESARROLLO

Este apartado consistirá en la explicación de como se ha ido desarrollando todo el proceso de la creación de la aplicación web de manera que quede claro todos los pasos realizados y el funcionamiento de la aplicación obtenida. Se han usado diferentes herramientas y métodos que se procederán a explicar a continuación.

2.1. CONTROL DE VERSIONES - GIT

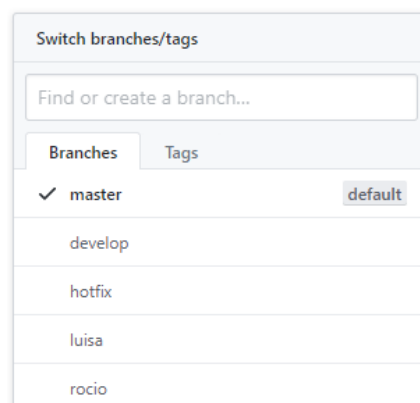
Durante todo el proceso del desarrollo de la práctica se ha hecho uso de la herramienta Git para facilitar el trabajo en equipo y la gestión de versiones del proyecto.

Se ha creado un repositorio privado (Practica-2-TAP-Ir) en el que se ha dado acceso a los integrantes del grupo y al profesor mediante la herramienta Github.

<https://github.com/Rochsos/Practica-2-TAP-Ir>

Para el buen uso de Git se ha usado la metodología Gitflow, que consiste en un flujo de trabajo que define un modelo estricto de ramificación proporcionando un marco sólido para gestionar proyectos grandes. Por lo que la estructura de ramificación se ha basado en el modelo de esta metodología haciendo uso de 5 ramas de trabajo que se procederán a explicar:

- **Máster:** es la rama principal y solo se subirán los cambios que estén preparados para subir a producción.
- **Develop:** se crea y se integra a partir de la rama Máster y estará el código que conformará la siguiente versión planificada del proyecto.
- **Hotfix:** se origina y se integra de Máster y se usará para corregir errores y bugs en el código en producción.
- **Luisa (Feature):** se origina e incorpora a Develop y se usa para desarrollar nuevas características de la aplicación y pertenece a la desarrolladora Luisa.
- **Rocío (Feature):** se origina e incorpora a Develop y se usa para desarrollar nuevas características de la aplicación y pertenece a la desarrolladora Rocío.



2.2. DIAGRAMA DE CLASES

La aplicación que se ha desarrollado está compuesta de:

- La clase principal Application, donde ejecutará toda la programación a partir del Spring.
- Un paquete Backend, donde se crearán y se definirán los modelos, controladores y repositorios que gestionarán la BBDD para el traspaso de datos y la definición de la estructura.

Este paquete está compuesto de:

- Un paquete Modelo, donde se encuentran las clases que representan las entidades de la BBDD.

Este paquete está compuesto de:

- La clase abstracta EntidadAbstracta, que gestionará el ID de cada elemento de las entidades para que no se haga en cada una de las entidades.
- La clase ListaTareas, que representa la entidad de las listas de las tareas que tendrán de propiedades el nombre de la lista y las tareas que se le asignarán
- La clase Tarea, que representa la entidad de las tareas que tendrán de propiedades el nombre de la tarea, una descripción, una prioridad (entre ALTA, MEDIA, BAJA), una fecha límite, un estado (entre COMPLETA, INCOMPLETA) y una lista.

- Un paquete Controlador, donde se encuentran las clases que van a contener las funcionalidades para gestionar las tareas de la aplicación.

Este paquete está compuesto de:

- Una clase ControladorListaTarea, donde están las funcionalidades para gestionar las listas de la aplicación.
- Una clase ControladorTarea, donde están las funcionalidades para gestionar las tareas de la aplicación.

- Un paquete Repositorio, donde se encuentran las interfaces que se encargaran de acceder y modificar la BBDD de la aplicación.

Este paquete está compuesto de:

- Una interfaz RepositorioLista, que se encargará de gestionar la tabla de la BBDD donde se guardan las listas de la aplicación.
- Una interfaz RepositorioTarea, que se encargará de gestionar la tabla de la BBDD donde se guardan las tareas de la aplicación.

- Un paquete UI, donde se encuentran las clases que definirán las plantillas HTML para la visualización de la aplicación y donde se recibirán las peticiones de los usuarios para realizar la gestión de la BBDD.

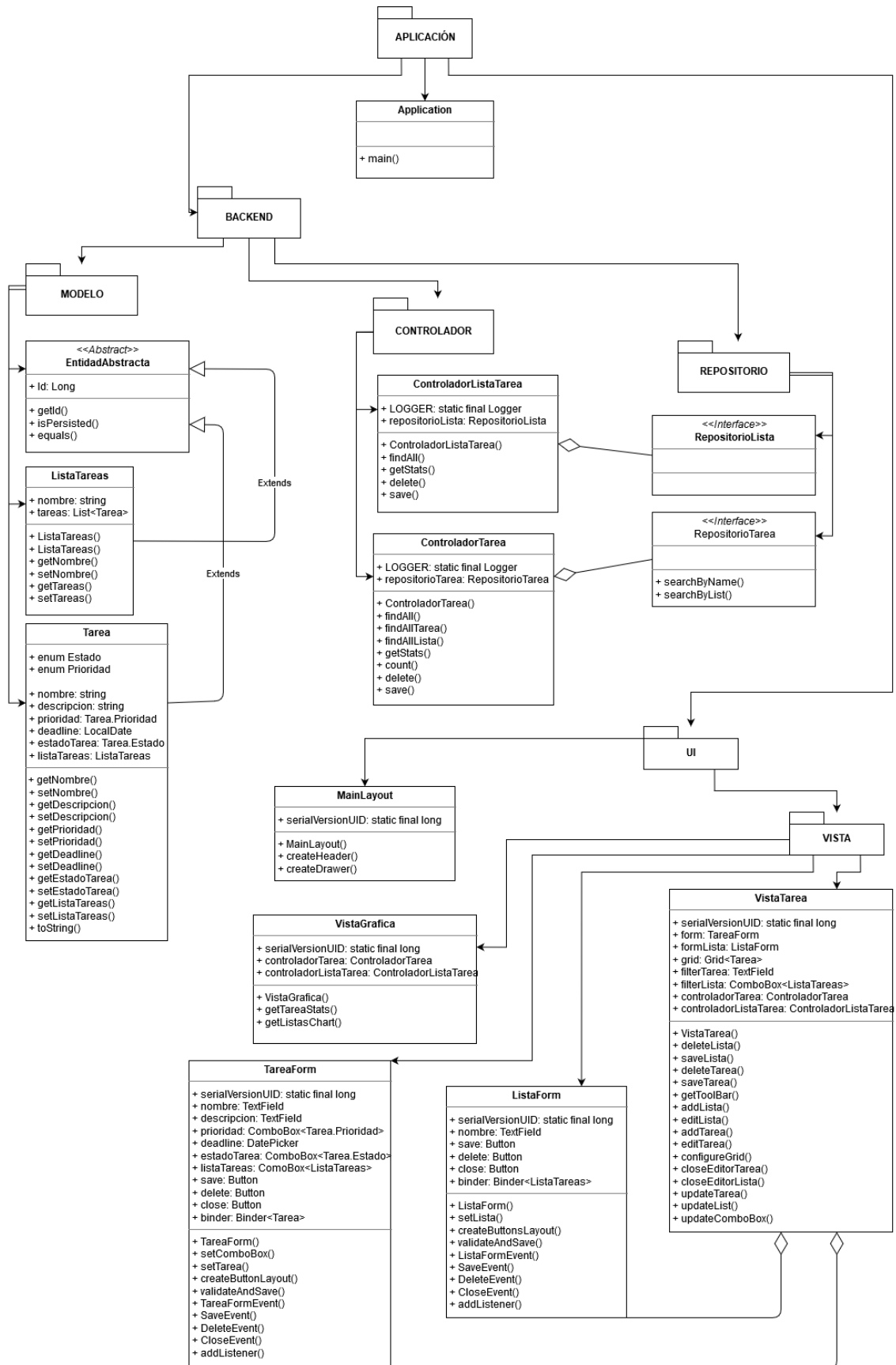
Este paquete está compuesto de:

- Una clase principal MainLayout, que se encargará de visualizar los diferentes componentes de la aplicación.
- Un paquete Vista, donde se encuentran las clases que definirán los componentes que tendrá la aplicación para que el usuario interactúe.

Este paquete está compuesto de:

- Una clase VistaGráfica, en donde se mostrará una gráfica sobre el número de tareas que tiene cada lista creada por el usuario.
- Una clase VistaTarea, en donde se mostrará una tabla con las tareas que estén almacenadas en la BBDD y donde el usuario podrá realizar las funcionalidades principales de añadir, eliminar, modificar y buscar en su tabla de tareas.
- Una clase TareaForm, en donde se define el formulario que deberá rellenar el usuario cuando desee añadir una tarea a la BBDD. También se definen los botones de guardar, eliminar o cancelar, para que el usuario decida según lo que quiera hacer.
- Una clase ListaForm, en donde se define el formulario que deberá rellenar el usuario cuando desee añadir una lista a la BBDD. También se definen los botones de guardar, eliminar o cancelar, para que el usuario decida según lo que quiera hacer.

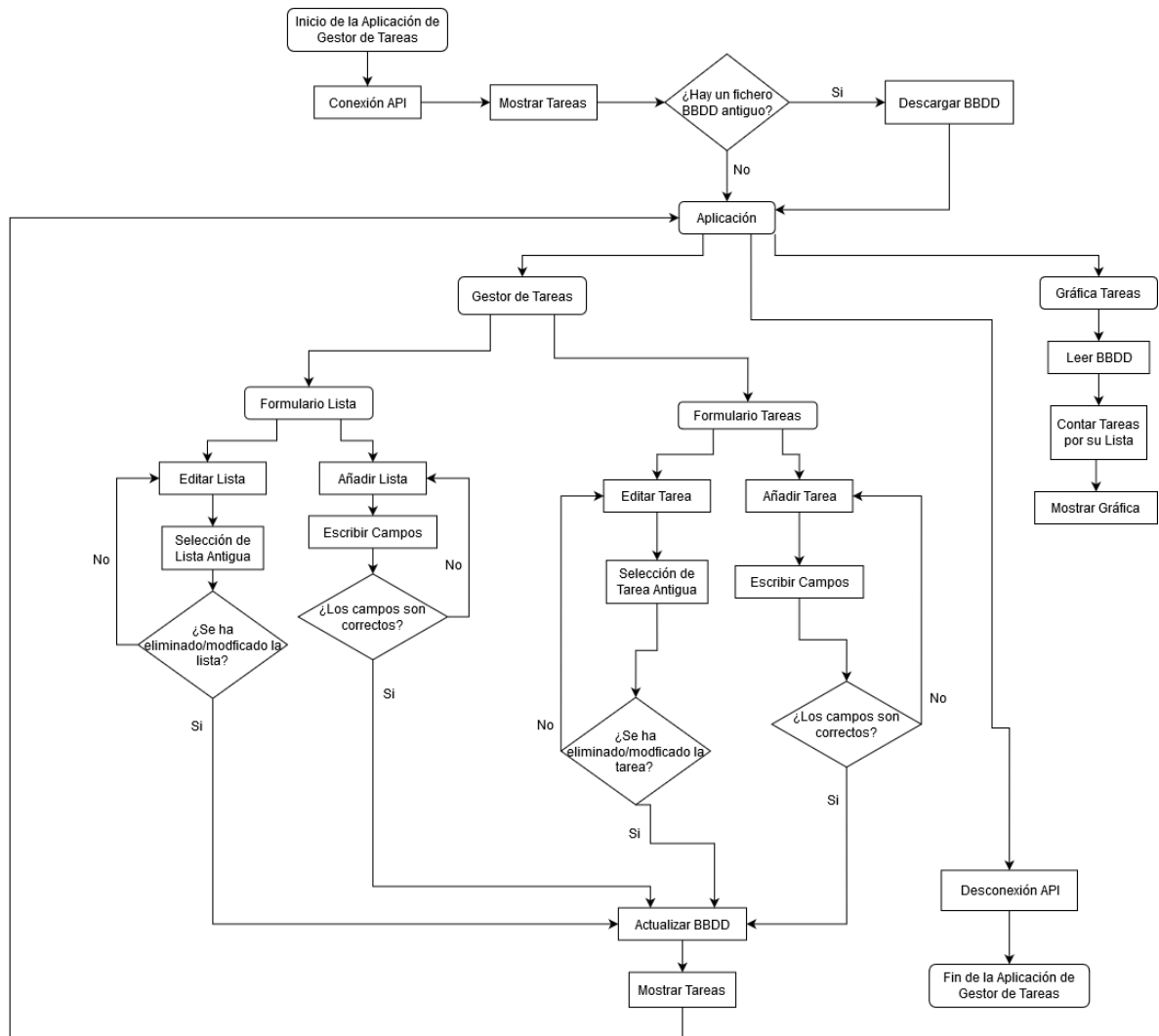
En la siguiente imagen se puede observar un diagrama de clases, donde se muestra la conexión entre las diferentes clases que componen la aplicación.



2.3. DIAGRAMA DE FLUJO

Un diagrama de flujo es una representación esquemática de los distintos pasos de un programa que nos sirve de forma complementaria en el proceso de creación de la estructura del programa antes de ponernos a programar.

En la siguiente imagen se puede observar un diagrama de flujo que indica el funcionamiento de la aplicación de gestión de tareas:



En resumen, cuando se entra en la aplicación lo primero que aparece es una tabla de las tareas que dispone el usuario, en el caso de que tenga algo guardado en la BBDD.

Después, para su aplicación de lista de tareas puede ir añadiendo listas (para agrupar sus tareas) o tareas, modificándolas e incluso eliminándolas como desee el usuario.

Además, el usuario dispone de una gráfica que mostrará cuantas tareas tiene por cada lista que haya creado, así verá en que lista tiene más tareas o en cual menos.

Todo esto, se explica más en detalle en el apartado Interfaz De Usuario de este mismo documento.

2.4. DISEÑO DEL SOFTWARE DE LA APLICACIÓN

En este apartado se procederá a explicar los principios o patrones usados referenciando a los conceptos adquiridos en la asignatura, de manera que expliquemos en que nos hemos apoyado o basado a la hora de desarrollar nuestro código.

Para empezar, al hablar de Programación Orientada a Objetos, es obvio referirnos a los **cuatro pilares fundamentales** que se cumplen y en los que nos hemos apoyado para el desarrollo correcto de nuestro código:

- Abstracción: los objetos representados en nuestro programa contienen tan solo los atributos y comportamientos principales en el contexto del problema, de manera que ignora otras variables menos importantes para no profundizar en detalles no relevantes.
- Encapsulación: nuestro programa contiene objetos encapsulados, de manera que ocultan partes de su estado y comportamiento a otros objetos para que únicamente sean accesibles desde los métodos de la propia clase. Además, las interfaces y clases abstractas usadas en nuestro programa están basadas en los conceptos ya descritos de abstracción y encapsulación.
- Herencia: este es el pilar más fuerte que asegura la reutilización de código, ya que a partir de esta característica es posible reutilizar (heredar) las características y comportamientos de una clase superior llamada clase padre, a sus clases hijas. Nuestro código está constantemente haciendo uso de este pilar de manera que se hace un código muy reutilizable por otras clases o métodos.
- Polimorfismo: son los diferentes comportamientos que tiene un método dependiendo desde la clase que sea llamado en nuestro programa.

Los **principios de diseño de software** y de la programación orientada a objetos que se han empleado para el desarrollo de esta aplicación son:

- Encapsulate what varies: se han encapsulado los aspectos que podían cambiar más, si se necesitaran modificaciones, y separarlos de los que se mantienen igual, de esta manera se evitaría cambiar todo el código.
- Program to an Interface, not an Implementation: al tener diferentes clases con varias responsabilidades, para evitar la dependencia entre clases concretas, se ha hecho uso de interfaces o clases abstractas en varias ocasiones para que fuera más sencillo y evitar que se rompa el código.
- Don't Repeat Yourself: se ha evitado repetir código en las diferentes partes de nuestra aplicación, de esta manera, se ha conseguido reutilizar varias veces el código con solo llamarlo de la manera correcta cuando era necesario.
- Single Layer of Abstraction Principle: se ha evitado que los métodos de las diferentes clases de la aplicación fueran extensos de contenido, de esta manera, se tenían bastantes métodos pero que realizaban cada uno una única función.

- De los principios S.O.L.I.D. se han podido cumplir:
 - Single Responsibility: cada clase (abstracta o no) o interfaz de la aplicación solo asumía una única responsabilidad, ya sea para diseñar el formulario, para gestionar la base de datos, para definir los componentes de los objetos....
 - Open/Closed: las entidades de la aplicación están abiertas para su extensión, pero cerradas para su modificación, así se evita que esas clases principales tengan muchas responsabilidades, dejando a otros objetos hagan la función necesaria sin saber la clase principal cómo lo hacen.
 - Interface Segregation: en vez de crear pocas interfaces que asumen muchas responsabilidades obligando a que las clases hereden funciones que no vayan a utilizar, se han hecho diferentes interfaces centradas en una única función para que las clases solo usen aquellas que necesiten.
 - Dependency Inversión: la gestión de la base de datos ha sido realizada por interfaces, consiguiendo que las clases principales eviten asumir dicha responsabilidad y no dependan ni conozcan dicho funcionamiento, solo la fiabilidad de que funciona para guardar los datos.

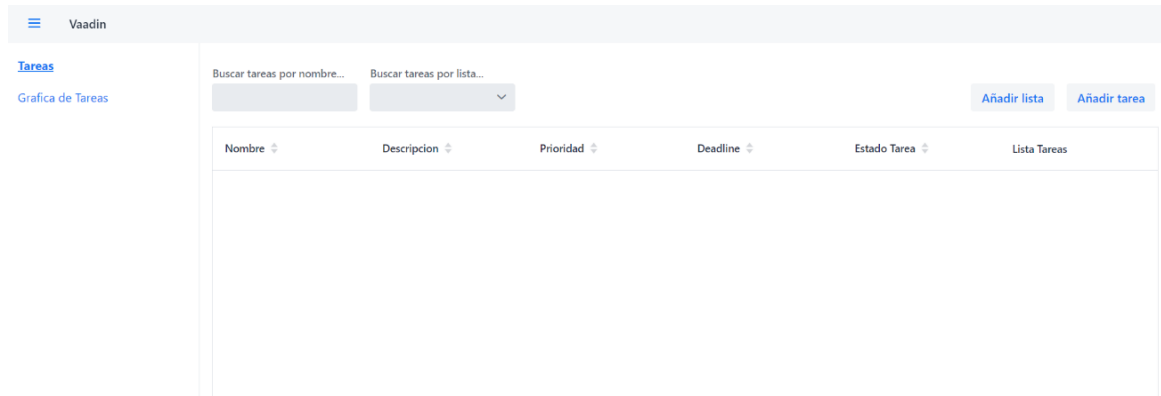
En cuanto a los patrones de diseño que nos facilitan una posible solución correcta para un problema de diseño en un contexto dado solo hemos hecho uso de uno:

- Observer: es un patrón de comportamiento que define una dependencia del tipo uno a muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. De manera que, por ejemplo, este patrón nos permite tener las interfaces de usuario actualizadas en todo momento y para así mantener la consistencia en todo el código.

2.5. INTERFAZ DE USUARIO

En este apartado mostraremos todas las funcionalidades de nuestra aplicación de manera que recorreremos cada interfaz con todas las acciones explicando cada una.

Al entrar en la aplicación, lo primero que aparecerá es la siguiente imagen:



En la anterior imagen aparece la tabla de tareas vacía debido a que no hay nada almacenado en la BBDD, en el caso de que si hubiera algo se vería de la siguiente manera:

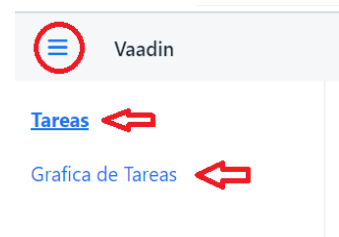
Nombre	Descripción	Prioridad	Deadline	Estado Tarea	Lista Tareas
Usuarios	Crear nuevos usuarios	MEDIA	2020-06-30	INCOMPLETA	Trabajo
Reunion	Hacer reunión con el departamento	ALTA	2020-06-11	INCOMPLETA	Trabajo
Limpiar	Limpiar el garaje	BAJA	2020-06-13	INCOMPLETA	Casa
Verduras	Comprar verduras	MEDIA	2020-06-10	INCOMPLETA	Compras

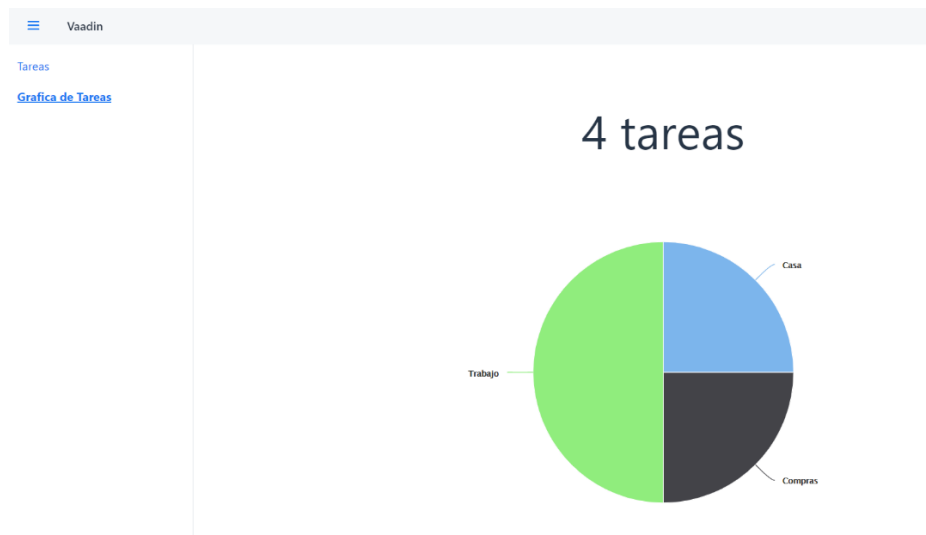
Las diferentes funcionalidades que dispone esta aplicación son las siguientes:

- Un menú para moverse por las diferentes pestañas que dispone la aplicación, además de un botón para poder ocultarlo.

Las pestañas que dispone el menú son:

- Una pestaña para la gestión de tareas (que corresponde a la página que se muestra nada más abrir la aplicación).
- Una pestaña para observar la gráfica de tareas, que determinará el número de tareas que tiene cada lista creada en la BBDD.





- Dos buscadores para poder buscar en la tabla de tareas ya sea por el nombre de la tarea o por la lista que tiene asignada.

Buscar tareas por nombre...
 Buscar tareas por lista...

- Cuando se utiliza el buscador por el nombre de la tarea, en la tabla solo se mostrarán las tareas que tenga su nombre parecido al que el usuario ha escrito.

Buscar tareas por nombre...
 Buscar tareas por lista...

Usuarios
 Añadir lista
 Añadir tarea

Nombre	Descripción	Prioridad	Deadline	Estado Tarea	Lista Tareas
Usuarios	Crear nuevos usuarios	MEDIA	2020-06-30	INCOMPLETA	Trabajo

Además, independientemente de la búsqueda, si el usuario selecciona una de las tareas de la tabla, se abrirá el formulario de tareas, por si el usuario desea modificar o eliminar la tarea seleccionada.

Buscar tareas por nombre...
 Buscar tareas por lista...

Añadir lista
 Añadir tarea

Nombre	Descripción	Prioridad	Deadline	Estado Tarea	Lista Tareas
Usuarios	Crear nuevos usuarios	MEDIA	2020-06-30	INCOMPLETA	Trabajo
Reunion	Hacer reunión con el departamento	ALTA	2020-06-11	INCOMPLETA	Trabajo
Limpiar	Limpiar el garaje	BAJA	2020-06-13	INCOMPLETA	Casa
Peluquería	Llamar a la peluquería	MEDIA	2020-06-15	INCOMPLETA	Otros
Verduras	Comprar verduras	MEDIA	2020-06-10	INCOMPLETA	Compr...

Nombre de la tarea

Descripción

Prioridad

Fecha límite

Estado

Lista a la que pertenece

- Cuando se utiliza el buscador por la lista de la tarea, en la tabla solo se mostrarán las tareas que tengan asignada dicha lista, además se abrirá el formulario de lista por si el usuario desea eliminar la lista o modificar el nombre de la lista.

Buscar tareas por nombre... [Buscar tareas por lista...](#)

✕ ▼

[Añadir lista](#) [Añadir tarea](#)

Nombre	Descripcion	Prioridad	Deadline	Estado Tarea	Lista Tar
Usuarios	Crear nuevos usuarios	MEDIA	2020-06-30	INCOMPLETA	Trabajo
Reunion	Hacer reunión con el departamento	ALTA	2020-06-11	INCOMPLETA	Trabajo

Nombre de la lista

[Save](#) [Delete](#) [Cancel](#)

- Dos botones para poder añadir una lista nueva o añadir una nueva tarea.

[Añadir lista](#)[Añadir tarea](#)

- Cuando se pulsa el botón de añadir lista, se abrirá su correspondiente formulario en donde el usuario deberá rellenar el nombre de la lista para poder crearla.
Además, en el formulario incluye 3 botones para guardar, eliminar o cancelar según lo que el usuario quiera hacer.

Nombre de la lista

[Save](#) [Delete](#) [Cancel](#)

- Cuando se pulsa el botón de añadir tarea, se abrirá su correspondiente formulario en donde el usuario deberá rellenar todos sus campos para poder crear la tarea.
Además, en el formulario incluye 3 botones para guardar, eliminar o cancelar según lo que el usuario quiera hacer.

Nombre de la tarea

Descripcion

Prioridad

Fecha límite

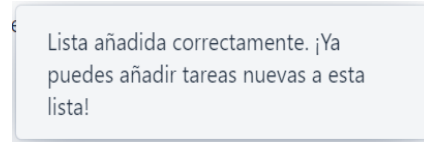
Estado

Lista a la que pertenece

[Save](#) [Delete](#) [Cancel](#)

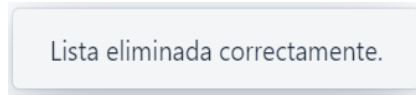
Además, mientras el usuario use las diferentes funcionalidades podrá ver diferentes notificaciones que confirmarán que está consiguiendo sus objetivos:

- Notificación de lista añadida o lista modificada

A light blue rounded rectangular notification box with a thin border. It contains the text: "Lista añadida correctamente. ¡Ya puedes añadir tareas nuevas a esta lista!"

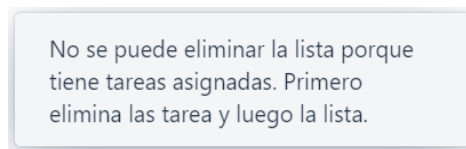
Lista añadida correctamente. ¡Ya puedes añadir tareas nuevas a esta lista!

- Notificación de lista eliminada

A light blue rounded rectangular notification box with a thin border. It contains the text: "Lista eliminada correctamente."

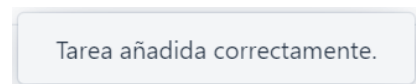
Lista eliminada correctamente.

- Notificación de que hay un problema al eliminar la lista

A light blue rounded rectangular notification box with a thin border. It contains the text: "No se puede eliminar la lista porque tiene tareas asignadas. Primero elimina las tarea y luego la lista."

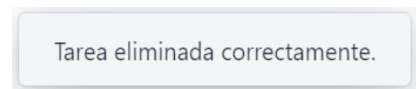
No se puede eliminar la lista porque tiene tareas asignadas. Primero elimina las tarea y luego la lista.

- Notificación de tarea añadida o tarea modificada

A light blue rounded rectangular notification box with a thin border. It contains the text: "Tarea añadida correctamente."

Tarea añadida correctamente.

- Notificación de tarea eliminada

A light blue rounded rectangular notification box with a thin border. It contains the text: "Tarea eliminada correctamente."

Tarea eliminada correctamente.

2.6. BASE DE DATOS H2

El motor de base de datos que ha sido utilizada en nuestro programa es H2, sistema Open Source escrito en su totalidad en Java y creado principalmente para entornos de desarrollo.

El propósito de esta base de datos es agilizar el proceso de desarrollo, ya que H2 puede crear la estructura de las tablas basándose en nuestras entitys JPA, cargar datos iniciales para las pruebas, hacer transacciones CRUD (aunque temporales)...

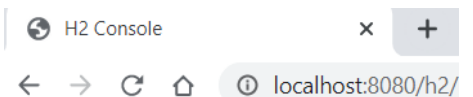
Hemos usado este sistema de administrador de bases de datos relacionales debido a que tuvimos algún problema con MySQL y debido a que había mucha información de la compatibilidad de Spring Boot con H2.

Las características de esta herramienta son:

- Alta integración al estar implementada en Java.
- Uso en diferentes plataformas
- Rápida y de gran velocidad debido a su estrategia de optimización basada en costes.
- Tamaño reducido (el JAR ocupa tan solo 1MB)
- Modo embebido realizando la gestión de los datos en archivos haciendo uso de una pequeña parte de la memoria. (Es una opción)
- Modo en memoria realizando la gestión de los datos directamente sobre la memoria, lo que acelera enormemente las operaciones realizadas.

Hay dos modos de usar la consola para hacer consultas o inserciones:

- Mediante la ejecución del programa instalado H2 console, e introduciendo los datos correctos para conectarnos a nuestra Base de Datos.
- Mediante la ejecución de nuestro programa y en la url (localhost:8080) añadir el parámetro h2.



Los datos correctos para conectarnos a nuestra base de datos son los siguientes:

- URL JDBC: jdbc:h2:file:~/tapdb
- Usuario: root
- Contraseña: root

A screenshot of the H2 Console 'Registrar' (Register) form. The form has a title bar with 'English', 'Preferencias', 'Tools', and 'Ayuda'. The main content area is titled 'Registrar'. It contains several fields: 'Configuraciones guardadas:' with a dropdown menu showing 'Generic H2 (Server)'; 'Nombre de la configuración:' with a text input field containing 'Generic H2 (Server)' and buttons 'Guardar' and 'Eliminar'; 'Controlador:' with a text input field containing 'org.h2.Driver'; 'URL JDBC:' with a text input field containing 'jdbc:h2:file:~/tapdb'; 'Nombre de usuario:' with a text input field containing 'root'; and 'Contraseña:' with a text input field containing '****'. At the bottom, there are two buttons: 'Conectar' and 'Probar la conexión'.

Para el correcto funcionamiento de la conexión entre la aplicación y la base de datos se ha modificado el archivo *application.properties* de esta manera:

```

1 # PARAMETROS DE CONEXION A LA BASE DE DATOS
2
3 #Habilitar la consola H2
4 spring.h2.console.enabled=true
5 spring.h2.console.path=/h2
6
7 #Url donde esta el servicio de tu mysql y el nombre de la base de datos
8 spring.datasource.url=jdbc:h2:file:~/tapdb;IFEXISTS=FALSE;DB_CLOSE_DELAY=-1;
9 #Indica el driver/lib para conectar java a mysql
10 spring.datasource.driverClassName=org.h2.Driver
11 #spring.datasource.driver-class-name=org.h2.Driver
12 #Usuario y contraseña para tu base de datos
13 spring.datasource.username=root
14 spring.datasource.password=root
15 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
16 spring.jpa.hibernate.ddl-auto=update
17 #Imprime en tu consola las instrucciones hechas en tu base de datos.
18 spring.jpa.show-sql = true
19

```

Y el archivo sql por defecto para la creación de las tablas de la base de datos es:

```

1 DROP TABLE IF EXISTS LISTA_TAREAS;
2 CREATE TABLE LISTA_TAREAS(
3     ID BIGINT NOT NULL,
4     NOMBRE VARCHAR(255) NOT NULL,
5     PRIMARY KEY(ID));
6
7 DROP TABLE IF EXISTS TAREA;
8 CREATE TABLE TAREA(
9     ID BIGINT NOT NULL,
10    DEADLINE DATE NOT NULL,
11    DESCRIPCION VARCHAR(255) NOT NULL,
12    ESTADO_TAREA VARCHAR(255) NOT NULL,
13    NOMBRE VARCHAR(255) NOT NULL,
14    PRIORIDAD VARCHAR (255) NOT NULL,
15    LISTATAREAS_ID BIGINT NOT NULL,
16    PRIMARY KEY(ID));
17

```

Al añadir unos datos de prueba en nuestra aplicación, podemos comprobar como al hacer una consulta a la base de datos también se han guardado:

The screenshot shows the H2 Console interface. On the left, the database structure is displayed, including the LISTA_TAREAS table with columns ID (BIGINT NOT NULL), NOMBRE (VARCHAR(255)), and LISTATAREAS_ID (BIGINT NOT NULL). The TAREA table is also shown with columns ID, DEADLINE, DESCRIPCION, ESTADO_TAREA, NOMBRE, PRIORIDAD, and LISTATAREAS_ID. The main panel shows a query result for 'select * from lista_tareas'. The results are as follows:

ID	NOMBRE
5	Trabajo
6	Casa
7	Otros
12	Compras

The screenshot shows the H2 Console interface. On the left, the database structure is displayed, including the TAREA table with columns ID, DEADLINE, DESCRIPCION, ESTADO_TAREA, NOMBRE, PRIORIDAD, and LISTATAREAS_ID. The main panel shows a query result for 'select * from tarea;'. The results are as follows:

ID	DEADLINE	DESCRIPCION	ESTADO_TAREA	NOMBRE	PRIORIDAD	LISTATAREAS_ID
8	2020-06-30	Crear nuevos usuarios	INCOMPLETA	Usuarios	MEDIA	5
9	2020-06-11	Hacer reunión con el departamento	INCOMPLETA	Reunion	ALTA	5
10	2020-06-13	Limpiar el garaje	INCOMPLETA	Limpiar	BAJA	6
11	2020-06-15	Llamar a la peluquería	INCOMPLETA	Peluquería	MEDIA	7

Si volviéramos a ejecutar podemos comprobar como la base de datos es persistente y los datos siguen guardados en un archivo por defecto.

2.7. REPORTE DE SONARQUBE

Sonarqube es una plataforma para evaluar código fuente con software libre y usa diversas herramientas de análisis estático de código fuente para obtener métricas que puedan ayudar a mejorar la calidad del código de un programa.

Algunos de los análisis que saca esta herramienta pueden ser:

- informar sobre código duplicado
- estándares de codificación
- falta de pruebas unitarias
- cobertura de código
- code smells
- malas prácticas
- comentarios
- ...

Esta herramienta de trabajo nos ha permitido cambiar nuestro código internamente, pero sin alterar la estructura externa, es decir como refactorizarlo. Aunque la calidad interna es algo no visible al usuario, llega un momento que tener una mala calidad de código interna pasa a ser un problema externo, ya que al hacer un cambio lleva mucho más tiempo del que debería.

Hay dos formas de utilizar SonarQube o mediante un cliente llamado SonarQube Scanner o con un plugin de Maven. En nuestro caso lo hemos usado con el plugin de Maven, aunque a su vez hemos usado el IDE SonarLint que nos iba ayudando a la hora de realizar el código.

El servidor se ha instalado usando una imagen de Docker para agilizar todo el proceso. Primero descargamos la imagen subida a Docker Hub de SonarQube con un Docker pull. Y observamos con Docker images como se ha instalado correctamente.

```
C:\WINDOWS\system32>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
c4bde7a5bc2a: Pull complete
b35a2496a4dd: Pull complete
a10cefb8b455: Pull complete
Status: Downloaded newer image for sonarqube:latest
docker.io/library/sonarqube:latest

C:\WINDOWS\system32>docker images
```

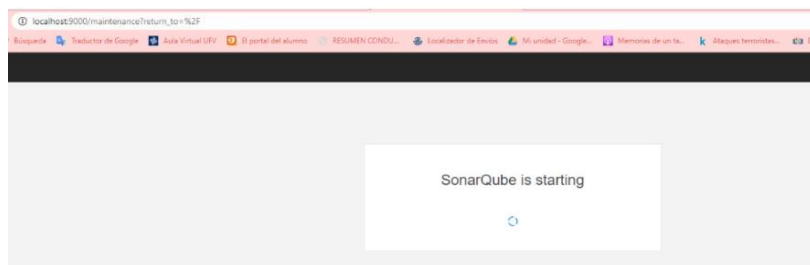
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sonarqube	latest	d05d6af6a6b7	4 weeks ago	457MB
jenkinsmaven	latest	4dc5d1a9e41d	6 weeks ago	808MB
jenkinspractica	latest	4dc5d1a9e41d	6 weeks ago	808MB
ubuntu	vim_1	236b57621dd9	6 weeks ago	153MB
<none>	<none>	42080f626c2f	6 weeks ago	127MB
nginx	latest	e791337790a6	7 weeks ago	127MB
jenkins/jenkins	lts	7e250da768ed	2 months ago	619MB
alpine	latest	a187dde48cd2	2 months ago	5.6MB
ubuntu	latest	4e5021d210f6	2 months ago	64.2MB

Luego ejecutamos la imagen para crear un contenedor corriendo de SonarQube e indicamos el puerto en el que queremos ejecutarlo. Observamos como el contenedor está corriendo con Docker ps.

```
C:\WINDOWS\system32>docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube
aa3482c8c0dda7c52eda508fa944b6988914f71b99994a35cc10795a948ec71b

C:\WINDOWS\system32>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
aa3482c8c0dd        sonarqube          "bin/run.sh bin/sona..."   6 seconds ago      Up 5 seconds        0.0.0.0:9000->9000/tcp, 0.0.0.0:9092->9092/tcp   sonarqube
```

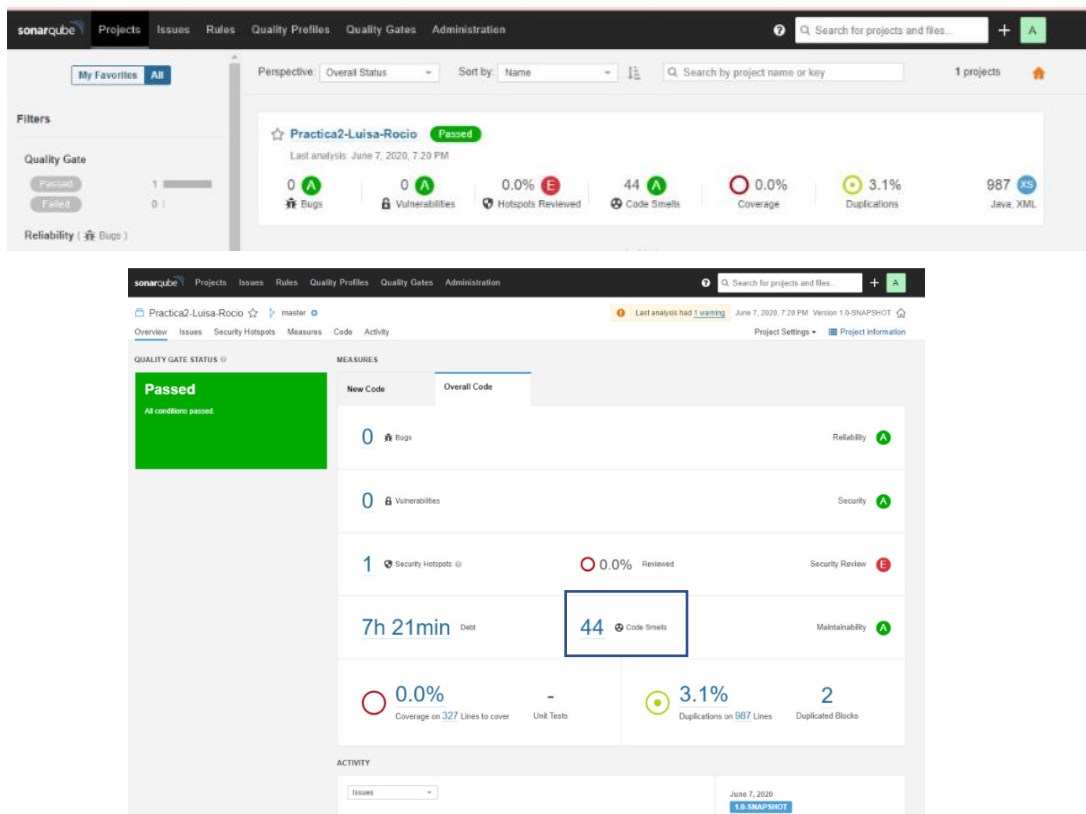
Ejecutamos en Maven build el siguiente comando: clean install sonar:sonar y ya podríamos entrar en la url: localhost:9000 y vemos como se ejecuta correctamente SonarQube.



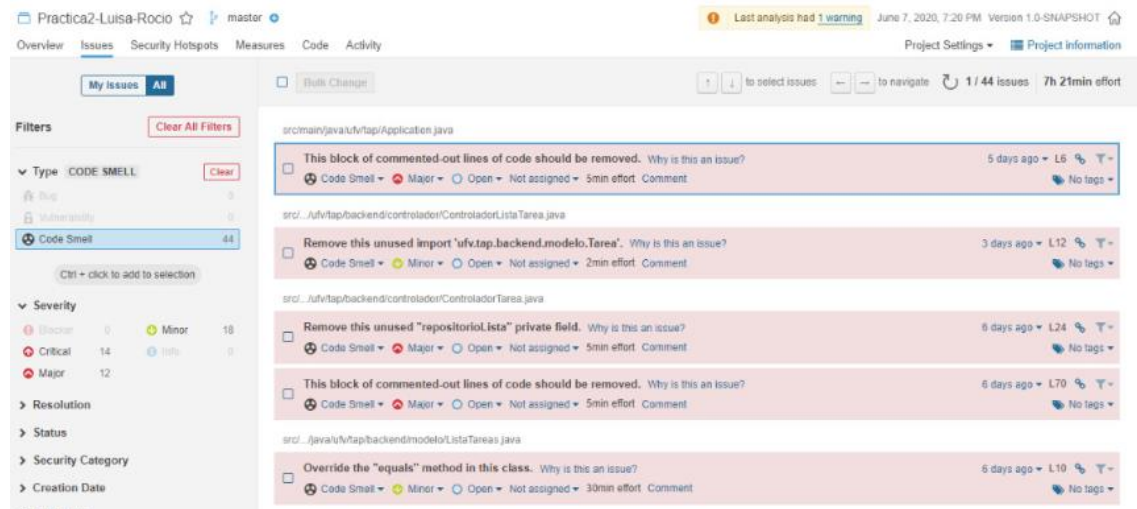
Una vez se ejecuta podremos ver en la consola como se ha ejecutado correctamente y nos indica que veamos más detalles en la url indicada.

```
[INFO] Analysis report compressed in 2308ms, zip size=63 KB
[INFO] Analysis report uploaded in 533ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=ufv.tap%3A practica2-luisa-rocio
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://localhost:9000/api/ce/task?id=AXKPzIEHh3rvRzEGrtb3
[INFO] Analysis total time: 41.378 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:49 min
[INFO] Finished at: 2020-06-07T19:20:45+02:00
[INFO] -----
```

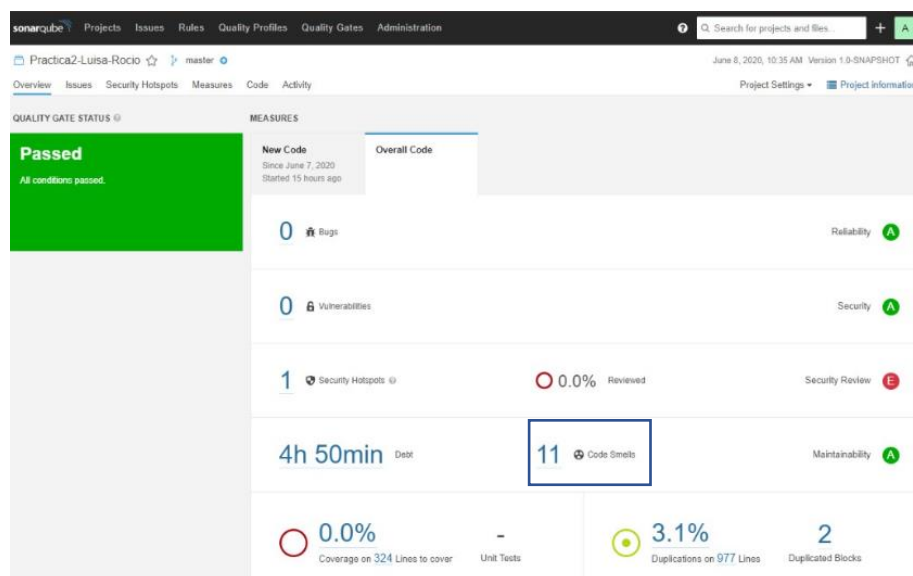
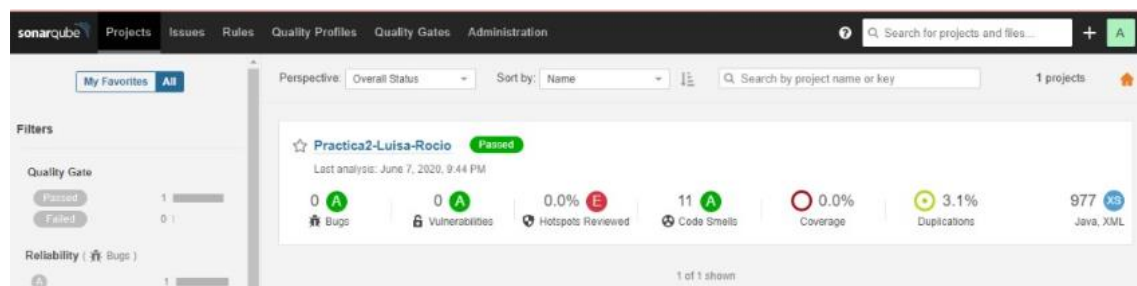
En la primera ejecución obtuvimos los siguientes resultados:



El resultado es bastante aceptable, pero decidimos intentar mejorar el número de Code Smells ya que eran 44, sobre todo los Critical. Algunos de estos eran:

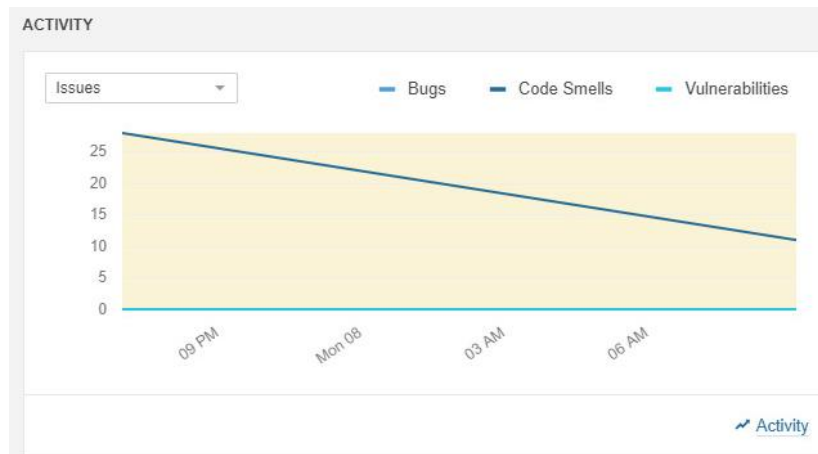


Por lo que modificamos nuestro código y obtuvimos el siguiente resultado:



Conseguimos reducir los Code Smells de 44 a 11, manteniendo el resto de los valores.

Tal y como podemos observar en esta gráfica que incluye la herramienta SonarQube, el número de Code Smells ha reducido considerablemente tras los siguientes reportes. El número de Bugs se ha mantenido a 0 al igual que las vulnerabilidades.



3. CONCLUSIONES

Finalmente, como conclusiones podemos mencionar que hemos podido desarrollar la práctica en grupo sin problemas entre los dos integrantes, aunque de una manera remota por la situación, esto no nos ha dificultado el desarrollo ya que tenemos experiencia por otros trabajos realizados por herramientas de llamadas en remoto.

Este trabajo nos ha permitido desarrollar más capacidades en herramientas que desconocíamos como SonarQube y ver realmente la importancia que tiene un reporte de la calidad para proyectos más grandes y que puedan ser reutilizables, además de la ayuda que nos ha facilitado esta herramienta para corregir o refactorizar el código.

Asimismo, hemos podido recordar los principios y patrones de diseño de software y de la programación orientada a objetos, además de los pilares de este. Lo hemos podido entender de una manera más práctica en su desarrollo.

También hemos repasado conceptos vistos a lo largo de la carrera como los diagramas de clase y de flujo usando la herramienta draw.io que nos facilita el tener una idea más clara de la lógica de la aplicación.

Los únicos problemas que citaríamos a la hora de realizar el desarrollo de la práctica son dos:

- Primero comenzamos la práctica usando como motor de base de datos MySQL, pero tuvimos problemas y por eso decidimos realizarla con H2.
- No hemos podido desplegar la aplicación en Heroku tal y como hemos intentado ya que la base de datos H2 no es compatible con la herramienta de despliegue Heroku, aunque al no ser un punto obligatorio en el enunciado de la práctica no es un problema como tal.

Práctica2-Luisa-Rocío

4. BIBLIOGRAFÍA

<https://moodleufv.ufv.es/>

<https://www.eclipse.org/>

<https://github.com/>

<https://spring.io/>

<https://vaadin.com/>

<https://maven.apache.org/>

<https://www.h2database.com/html/main.html>

<https://hub.docker.com/>

<https://www.sonarqube.org/>

<https://app.diagrams.net/>

<https://stackoverflow.com/>

<https://mvnrepository.com/>

<http://aprendegit.com/que-es-git-flow/>

<http://www.utn.edu.ec/reduca/programacion/poo/pilares.html#:~:text=Es%20el%20pi%20de%20la,y%20m%C3%A9todos%20de%20un%20objeto.>

<https://www.baeldung.com/spring-boot-h2-database>

<https://www.federico-toledo.com/analisis-de-codigo-con-sonarqube/>

<https://maresmewebdevelopers.wordpress.com/2017/11/09/todo-list-app-vamos-a-hacer-una-lista-de-tareas-con-base-de-datos-h2-en-el-orm-jpa/>