



UNIVERSIDAD DE CASTILLA-LA MANCHA ESCUELA SUPERIOR DE INFORMÁTICA

Memoria del Proyecto

Álvaro Ávila Cabello
Daniel López Paredes
Luis Miguel Barreiro Cabello
Víctor Lay Gómez

Asignatura: Ingeniería del Software II

Titulación: Grado en Ingeniería Informática

Fecha: 15 de octubre del 2023

ÍNDICE

1. Introducción	1
1.1 Enunciado de la práctica	1
2. Análisis de requisitos	1
3. Casos de usos y diagrama clase.....	2
3.1. Diagrama de casos de usos.....	2
3.2 Diagrama de clase	3
4. Construcción de la estructura	3
4.1 Maven y Spring-Boot:.....	3
5. Control del proyecto.....	4
6. Gestión de calidad	7
6.1 Quality Gates.....	8
7. Testing.....	9
7.1 Pruebas unitarias	9
7.2 Informes de análisis.....	12
8. Mantenibilidad	12
8.1 Plan de mantenimiento	12

1. Introducción

1.1 Enunciado de la práctica

Se pretende desarrollar un sistema web para la gestión de una biblioteca con los siguientes requisitos, que lo pueden usar tres usuarios diferentes con la siguiente funcionalidad

2. Análisis de requisitos

Se pretende desarrollar un sistema web para la gestión de una biblioteca con los siguientes requisitos, que lo pueden usar tres usuarios diferentes con la siguiente funcionalidad

- Administrador de biblioteca
 - Da de alta a nuevos títulos en el catálogo, para lo cual es requisito indispensable que al menos añada un ejemplar.
 - Añadir nuevos ejemplares, es decir, un nuevo libro (copia) de un título.
 - Borrar o actualizar un título existente.
 - Borrar un ejemplar (por ejemplo, por estar en mal estado, extravío, etc.)
- Bibliotecario/a:
 - Prestar ejemplar. En este caso el sistema comprobará que el usuario no tiene el cupo de libros para sacar completo, ni tiene penalizaciones pendientes.
 - Gestionar devolución de un ejemplar. En el caso de devolución con retardo, el sistema aplicará una penalización al usuario que se tendrá en cuenta para futuros préstamos.
 - El trabajador podrá hacer la reserva de un ejemplar cuando un usuario desee un ejemplar que no está disponible. Al realizar un préstamo, todas las reservas para dicho ejemplar se borrarán.
- Usuario:
 - A través del terminal en la biblioteca podrá de forma autónoma hacer la misma funcionalidad que el trabajador de la biblioteca.

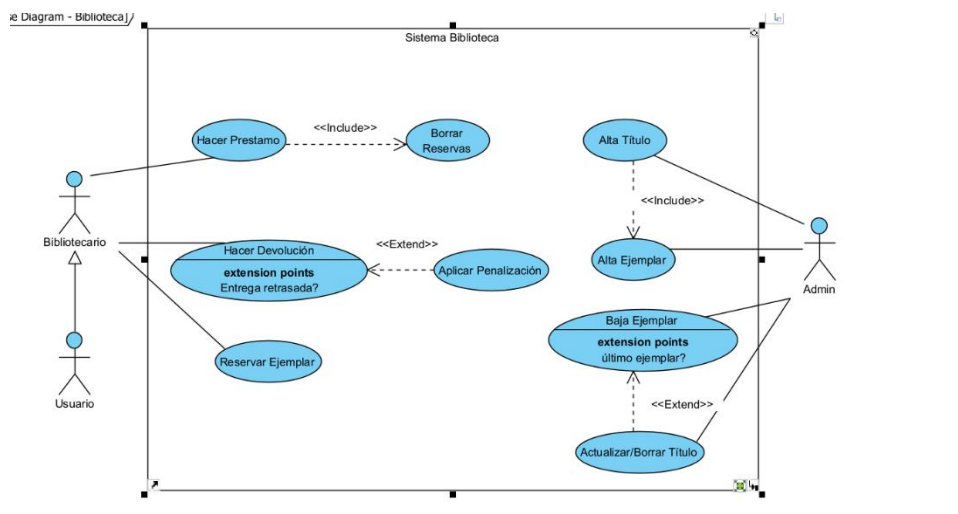
REQUISITOS FUNCIONALES

1. Añadir título
2. Borrar título
3. Actualizar título
4. Añadir ejemplar (copia de un título)
5. Borrar ejemplar (y si es el último borrar el título)
6. Prestar un ejemplar
7. Comprobar cupo de libros (núm. máx. de libros prestados por usuario)
8. Comprobar penalizaciones
9. Recoger devolución de ejemplar
10. Aplicar penalización por retardo de ejemplar
11. Reservar ejemplar cuando no esté disponible

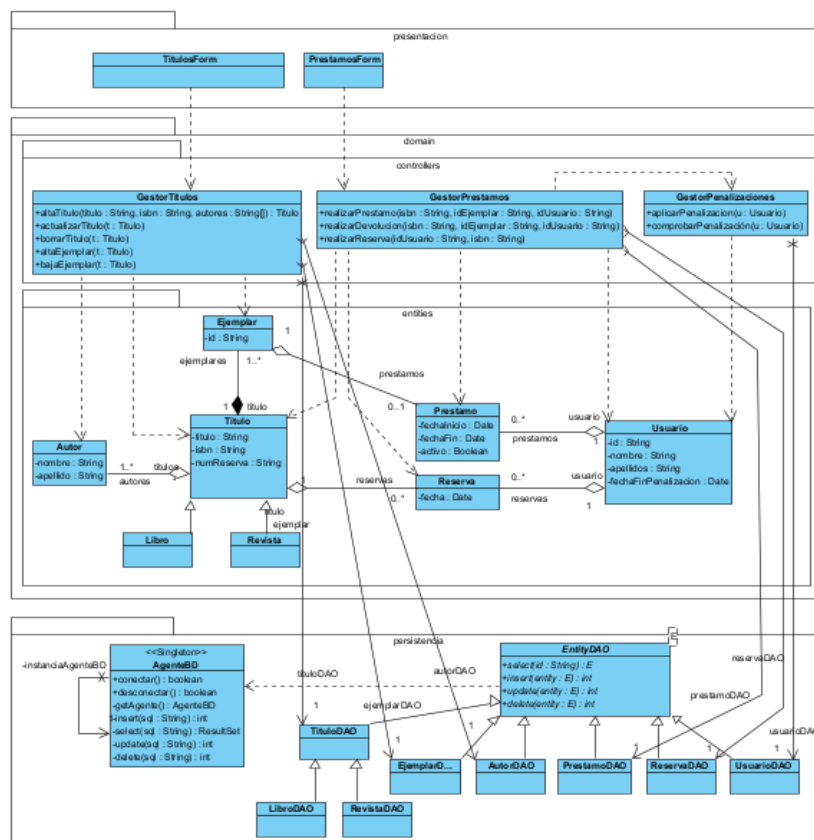
3. Casos de usos y diagrama clase

3.1. Diagrama de casos de usos

A continuación, se muestra el diagrama de caso de usos:



3.2 Diagrama de clase



4. Construcción de la estructura

4.1 Maven y Spring-Boot:
















MAVEN

Para la construcción de la infraestructura del proyecto, emplearemos Maven, ya que nos permite aplicar una serie de patrones para construir la infraestructura del proyecto con el fin de proporcionar una visión coherente.

Nos proporciona medios para ayudar a gestionar compilación, construcciones, documentación, informes, dependencias, configuración del software y las publicaciones.

La estructura del proyecto Maven es la mostrada en la imagen, como se puede apreciar la estructura Maven, permite obtener unos beneficios ya que esta estructura es la estándar, además de una configuración fácil y rápida, también la gestión de las dependencias mediante las descargas automáticas del repositorio, gestión del repositorio

Arquitectura de Maven: hay un archivo que es muy importante destacar en la construcción de la estructura del proyecto que es el POM (Project Object Model), contiene los metadatos sobre el proyecto, la localización de los directorios, desarrolladores y contribuidores, dependencias....

- ▼  ByteMasters_Library [ByteMasters_Library develop]
 - >  src/main/java
 - >  src/main/resources
 - >  src/test/java
 - >  JRE System Library [jre]
 - >  Maven Dependencies
 - >  database
 - >  src
 - >  target
 -  derby.log
 -  HELP.md
 -  mvnw
 -  mvnw.cmd
 -  pom.xml
 -  README.md

5. Control del proyecto

Para el sistema de control de versiones, utilizaremos la herramienta **git**, esta nos permite almacenar los cambios que se produzcan y continuar seguir trabajando, también poder volver hacia atrás a versiones anteriores, para garantizar la estabilidad del software y compartir distintas versiones realizadas por diferentes miembros del equipo de desarrollo y una de las características mas importantes es la seguridad frente a eliminaciones erróneas o fallos en los discos.

Git permite el desarrollo del software de forma no lineal , a través de las ramas en las cuales se pueden desarrollar las partes de las diferentes funcionalidades del software sin que se altere la estructura en otras.

Para ello se emplearan 4 tipo de ramas, la rama Master o main, la rama develop, la rama de funcionalidades y la rama de hotfix.

Como se puede ver en el proyecto las diferentes ramas, en el apartado de git se nos muestra toda la configuración de las ramas.

- ▼ Local
 - develop bbb9868 Merge pull request #51 añadida ventana inicio y login
 - feature-login 766a490 login hecho
- ▼ Remote Tracking
 - origin/develop bbb9868 Merge pull request #51 añadida ventana inicio y login
 - origin/feature-añadir-titulo ac79c03 añadir autores a un titulo separados por comas y verificar que no se generen duplicados en la base de datos
 - origin/feature-detalles-titulo 715fa1d Merge branch 'develop' into feature-detalles-titulo
 - origin/feature-editar-numero-ejemplares 5d3e6b6 cambio en la jerarquía de archivos
 - origin/feature-interfaz-inicio 1e91a44 actualizando gitignore
 - origin/feature-login 766a490 login hecho
 - origin/feature-persistencia c604ec6 añadido login
 - origin/feature-registrar-devolucion-ejemplar 48f1ed7 añadida feature para registrar la devolucion de un ejemplar
 - origin/feature-sonar 4cea830 Merge pull request #49 feature devolucion ejemplar
 - origin/feature-usuario-prestar-ejemplar 1c08596 feature pedir prestamo desde la ventana de usuario
 - origin/hotfixes fe61bdb README.md añadido
 - origin/main fe61bdb README.md añadido
 - origin/releases fe61bdb README.md añadido


A continuación, se muestra el historial en el cual se ve aprecian las ramas, y cuando se termina se hace merge con la rama de desarrollo

Id	Message	Author	Authored Date	Committer	Committed Date
09da128	empiece de interfaz, formulario de dar de alta titulo (solo nombre y isbn) pero no	Daniel	4 weeks ago	Daniel	4 weeks ago
ba1fe7	cambio application.properties y en entidades para solucionar problemas en la BBC	Daniel	4 weeks ago	Daniel	4 weeks ago
c1675b9	cambios persistencia entidad, estilos aplicados, base de datos cambio a create-	Daniel	4 weeks ago	Daniel	4 weeks ago
759ae5b	Merge pull request #29 from alvaro-avilac/develop	DaniLopez23	4 weeks ago	GitHub	4 weeks ago
e70c701	borradas entidades innecesarias	Daniel	4 weeks ago	Daniel	4 weeks ago
569fed1	conflictos solucionados	Daniel	4 weeks ago	Daniel	4 weeks ago
f2b35e9	Merge branch 'develop' of https://github.com/alvaro-avilac/ByteMasters_Library	Daniel	4 weeks ago	Daniel	4 weeks ago
eca4d4a	Update .gitignore	Álvaro Ávila Cabello	4 weeks ago	GitHub	4 weeks ago
5218c20	eliminados entidades y DAOS innecesarios hasta ahora	Daniel	4 weeks ago	Daniel	4 weeks ago
20548d6	DAO comenzados (falta cardinalidades), gestorTítulos reducido	Daniel	4 weeks ago	Daniel	4 weeks ago
08a387d	comienzo de formulario alta titulo	Alvaro Avila	4 weeks ago	Alvaro Avila	4 weeks ago
2f30cdc	Merge pull request #26 from alvaro-avilac/develop	Álvaro Ávila Cabello	4 weeks ago	GitHub	4 weeks ago
5c3c259	Merge branch 'feature-añadir-titulo' into develop	Álvaro Ávila Cabello	4 weeks ago	GitHub	4 weeks ago
8f287e2	Merge branch 'develop' of https://github.com/alvaro-avilac/ByteMasters_Library	Daniel	4 weeks ago	Daniel	4 weeks ago
5b4df09	cambios en los packages de los controllers y vaciarlos	Alvaro Avila	4 weeks ago	Alvaro Avila	4 weeks ago
b862712	cambios en los packages de las entidades	Alvaro Avila	4 weeks ago	Alvaro Avila	4 weeks ago
92d3c35	DAO corregido	Daniel	4 weeks ago	Daniel	4 weeks ago
7f078ae	capa servicio titulo, controlador titulos empezado	Daniel	4 weeks ago	Daniel	4 weeks ago
3d81254	Template, archivos html y application.properties	Daniel	4 weeks ago	Daniel	4 weeks ago
e7ea834	Esqueleto, archivos base visual paradigm	Daniel	4 weeks ago	Daniel	4 weeks ago
fe61bdb	origin/HEAD origin/hotfixes origin/main origin/releases README.md añadido	Alvaro Avila	5 weeks ago	Alvaro Avila	5 weeks ago
593c079	commit project	Alvaro Avila	5 weeks ago	Alvaro Avila	5 weeks ago
2a095fb	Initial commit	Daniel	5 weeks ago	Daniel	5 weeks ago

También se optó por tomar como servicio en la nube el alojar el repositorio con el proyecto, para ello utilizamos github, que es un servicio en la nube para repositorios remotos que utilizan git como sistema de control de versiones.


Github permite la colaboración en proyectos y poder compartir el código de una manera sencilla a través de github clonamos en los terminales la ubicación del repositorio , en el cual se realizara el proyecto , pero además github permite realizar muchas mas funciones a la hora de gestión del proyecto , en las que a continuación detallaremos.

En github también se puede ver la estructura del proyecto , como se aprecia en la imagen inferior que corresponde con la rama develop


ByteMasters_Library
Public
Watch 1

develop
15 branches
0 tags
Go to file
Add file
Code

This branch is **69 commits ahead** of main.
Contribute


DaniLopez23 Merge pull request #51 añadida ventana inicio y login ...
bbb9868 9 hours ago
72 commits

.mvn/wrapper	cambio en la jerarquía de archivos	last week
src	login hecho	yesterday
.gitignore	actualizando gitignore	last week
HELP.md	cambio en la jerarquía de archivos	last week
README.md	README.md añadido	last month
mvnw	cambio en la jerarquía de archivos	last week
mvnw.cmd	cambio en la jerarquía de archivos	last week
pom.xml	cambio en la jerarquía de archivos	last week

En github también permite poder establecer el backlog en su ventana Project en cada reunión de los sprint se definían los product backlogs, los cuales se iban añadiendo en el sprint de acuerdo con los principios de prioridad , como se puede apreciar en la imagen

Library ByteMaster Inc.
Backlog
By priority
By size
New View

-sprint: Sprint #1 Backlog Sprint #2 Backlog Sprint #3 Backlog
36
Discard
Save

Product Backlog

- ByteMasters_Library #13 Aplicar penalización (Bibliotecario) Low historia de usuario
- ByteMasters_Library #6 Comprobar penalizaciones (Bibliotecario) Low historia de usuario
- ByteMasters_Library #12 Comprobar cupo de ejemplares Low historia de usuario
- ByteMasters_Library #11 Reservar un título (Bibliotecario) Low historia de usuario
- ByteMasters_Library #10 Reservar un título (Usuario) Low historia de usuario

Sprint #4 Backlog

- ByteMasters_Library #2 Borrar título (Admin) Medium historia de usuario
- ByteMasters_Library #3 Prestar ejemplar (Bibliotecario) Medium historia de usuario
- ByteMasters_Library #8 Gestionar devolución (Bibliotecario) Medium historia de usuario
- ByteMasters_Library #38 Eliminar título desde el formulario de detalle 4h
- ByteMasters_Library #42 Crear template para cada rol

In progress

- ByteMasters_Library #5 Prestar ejemplar (Usuario) Low historia de usuario
- ByteMasters_Library #40 Interfaz de usuario para pedir un prestamo 6h
- ByteMasters_Library #47 Añadir SonarCloud a develop 2h

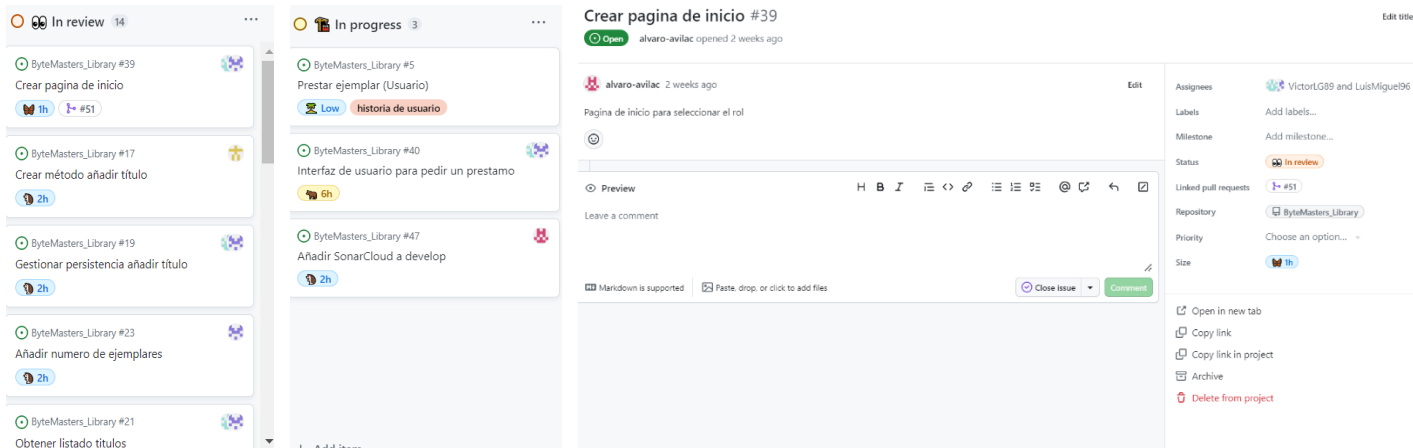
In review

- ByteMasters_Library #39 Crear pagina de inicio 1h #51
- ByteMasters_Library #17 Crear método añadir título 2h
- ByteMasters_Library #19 Gestionar persistencia añadir título 2h
- ByteMasters_Library #23 Añadir numero de ejemplares 2h
- ByteMasters_Library #21 Obtener listado títulos

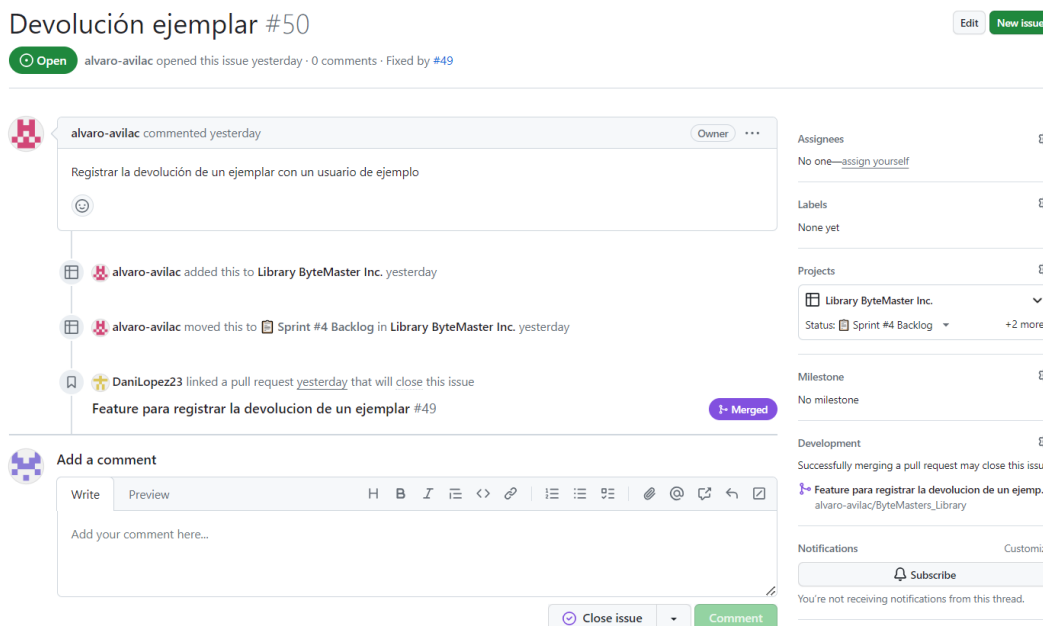
Done

- ByteMasters_Library #14 Actualizar título (Admin) Medium historia de usuario
- ByteMasters_Library #1 Añadir título (Admin) High historia de usuario
- ByteMasters_Library #4 Añadir ejemplar (Admin) High historia de usuario
- ByteMasters_Library #9 Borrar ejemplar (Admin) High historia de usuario
- ByteMasters_Library #33 Feature añadir título

En la imagen superior se muestra de manera general los diferentes estados en los que se encuentra los ítems del product backlog. Una vez asignado un ítem al backlog del sprint correspondiente, se identificaban los miembros del equipo en cargados, el tiempo estimado y su prioridad. A continuación se muestra en las imágenes los estados de in progress, in re wiew , así como una historia de usuario detallada



Cada historia de usuario, es asignada en una nueva issue donde se va a indicar en que rama se va a desarrollar esa nueva funcionalidad ejemplo devolución ejemplar , como se puede apreciar en el apartado developement, esta asignada a una rama de funcionalidad específica para ella.



6. Gestión de calidad

Para llevar a cabo un control de calidad de nuestro producto software utilizaremos SonarCloud. Esta herramienta nos provee un servicio de análisis estático de código basado en la nube.

Nuestros análisis los realizaremos en estas situaciones:

- Cada vez que se realice un commit en la rama **Develop**
- Cada vez que se intente hacer un **pull-request**
- Cada vez que se intente hacer algún commit en la rama **Main**

Sonar nos ofrece diferentes métricas que valoran diferentes aspectos del código que van desde buenas prácticas hasta vulnerabilidades de seguridad. Algunas de estas métricas son:

- Bugs: error que romperá tu código y debe corregirse de inmediato.
- Code Smells: problema de mantenibilidad que hace que tu código sea confuso y difícil de mantener.
- Vulnerabilities: punto en su código que está abierto a ataques.
- Security Hotspot: este es un fragmento de código que puede ser sensible en términos de seguridad y que un desarrollador debe revisar manualmente. Tras la revisión, se puede marcar si es o no una posible amenaza de seguridad.
- Coverage: esto le indica qué porcentaje de su código está cubierto por sus tests unitarios. Lo veremos más adelante en el apartado de Testing
- Duplications: número de bloques de líneas duplicados.

6.1 Quality Gates

Una “Quality Gate” es el indicador de que el código cumpla con un conjunto de requisitos de calidad y seguridad (determinados por nosotros) y por tanto es posible completar un “pull-request” o en cualquier momento que se lance el análisis.

Esto lo hace determinando un umbral con valores de las distintas métricas descritas previamente.

Para nuestro proyecto hemos creado nuestra propia Quality Gate donde definimos los diferentes valores de varias métricas. Si no cumplimos los valores esperados por dichas métricas sonar nos advertirá que la acción que vamos a llevar a cabo no supera las quality gates definidas. Esta es la definición de nuestra quality gate:

ByteMasters

✔ This quality gate is configured for Clean as You Code. [Learn More](#)

Conditions ?

Conditions on New Code

Conditions on New Code apply to all branches and to Pull Requests.

Metric	Operator	Value
Coverage	is less than	85.0%
Duplicated Lines (%)	is greater than	2.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

Por ejemplo, si desde una rama donde desarrollamos una funcionalidad queremos llevar a cabo un pull-request y la cobertura de nuestro programa con los test realizados de dicha implementación no supera un 85.0% de coverage sonar nos advertirá que dicho pull-request no contiene las características necesarias para superar la quality gate.

7. Testing

Para llevar a cabo el “testing” de nuestro producto software y realizar distintas pruebas para lograr detectar el mayor número de errores posibles vamos a utilizar JUnit 4.

Nos centraremos en el desarrollo de pruebas unitarias. Para estas pruebas unitarias necesitaremos seleccionar una serie de valores de prueba para maximizar la cobertura. Además, trabajaremos en un nivel de cobertura de decisiones observando el flujo del código en función del valor de prueba y el resultado de la condición.

7.1 Pruebas unitarias

A continuación, vamos algunas de las principales funciones que van a ser probadas y sus respectivas tablas de valores y diferentes casos de prueba para llevar a cabo el testing de dicha función:

- Tests Gestor Penalizaciones:
 - ComprobarPenalizaciones.

```
public boolean comprobarPenalizaciones(Usuario user) {
    LocalDate fechaActual = LocalDate.now();

    if (user.getFechaFinPenalizacion() != null) {
        LocalDate fechaFinPenalizacion = user.getFechaFinPenalizacion().toInstant()
            .atZone(java.time.ZoneId.systemDefault()).toLocalDate();
        // Comprueba si tiene penalizaciones
        if (fechaActual.isBefore(fechaFinPenalizacion)) {
            return true;
        }
    }

    return false;
}
```

Parametros	Tipo	Clases de equivalencia	Valores clase de equivalencia	valores limite (ligero)	Conjetura de errores
user.getFechaFinPenalizacion	Date (MM/DD/YYYY)	(-inf, 00/00/0000)		00/00/0000	99/99/9999
		[00/00/0000, 12/18/2000)	05/12/1980	12/18/2000	""
		[12/18/2000, 12/18/2010)	04/07/2003	12/18/2010	
		[12/18/2010, 12/18/2023)	09/05/2019	12/18/2023	
		[12/18/2023, +inf)	11/23/2050		
fechaActual	Date (MM/DD/YYYY)	(-inf, 00/00/0000)		00/00/0000	99/99/9999
		[00/00/0000, 12/18/2000)	05/12/1980	12/18/2000	
		[12/18/2000, 12/18/2010)	04/07/2003	12/18/2010	
		[12/18/2010, 12/18/2023)	09/05/2019	12/18/2023	
		[12/18/2023, +inf)	11/23/2050		

D1: user.getFechaFinPenalizacion() != null

D2: fechaActual.isBefore(fechaFinPenalizacion)

COBERTURA DE DECISIONES				
	user.getFechaFinPenalizacion	fechaActual	D1	D2
CP1	05/12/1980	05/12/1982	True	False
CP2	null	05/12/1982	False	-
	05/12/1982	21/12/1982	True	True

- ComprobarCupo

```
public boolean comprobarCupo(Usuario user) {
    long prestamosActivos = user.getPrestamos().stream().filter(Prestamo::isActive).count();

    // Comprueba cupo de prestamos (si ya tiene 4 prestamos no puede pedir prestados
    // mas)
    if (prestamosActivos + 1 > CUPO_MAXIMO) {
        return true;
    }

    return false;
}
```

Parametros	Tipo	Clases de equivalencia	Valores clase de equivalencia	valores limite (ligero)	Conjetura de errores
user.getPrestamos.activos.count()	int	(-inf, 0)	-4	0	9999999
		[0, 4)	2	4	
		[4, +inf)	12		
CUPO_MAXIMO	int	(-inf, 0)	-12		9999999
		[0, 4)	2		
		[4, +inf)	32		

COBERTURA DE DECISIONES			
	user.getPrestamos.activos.count()	CUPO_MAXIMO	D1: prestamosActivos + 1 > CUPO_MAXIMO
CP1	1	4	False
CP2	3	3	True

- Gestor títulos
 - SeleccionarEndpointPorRol

```
public String seleccionarEndpointPorRol(String rol) {
    String endpoint = "";

    if (rol.equals("admin")) {
        endpoint = "redirect:/admin";
    } else if (rol.equals("bibliotecario")) {
        endpoint = "redirect:/bibliotecario";
    } else if (rol.equals("usuario")) {
        endpoint = "redirect:/user";
    } else {
        endpoint = "redirect:/";
    }
    return endpoint;
}
```

Parametros	Tipo	Clases de equivalencia	Valores clase de equivalencia	valores limite (ligero)	Conjetura de errores
rol	String	cadena de caracteres	usuario admin bibliotecario cliente		conjunto mayor a 100 caracteres

COBERTURA DE DECISIONES				
	Rol	D1: rol.equals("admin")	D2: rol.equals("bibliotecario")	D3: rol.equals("usuario")
CP1	usuario	false	false	true
CP2	admin	true	-	-
CP3	bibliotecario	false	true	-
CP4	default	false	false	false

- tituloTieneReservasPrestamos

```
private boolean tituloTieneReservasPrestamos(Titulo titulo) {
    List<Prestamo> listadoPrestamos = prestamoService.listarPrestamos();
    for(Prestamo p: listadoPrestamos) {
        if(p.getEjemplar().getTitulo().equals(titulo)) {
            if (p.isActivo() && p.getEjemplar().getTitulo().equals(titulo)) {
                return true;
            }
        }
    }
    return false;
}
```

Parametros	Tipo	Clases de equivalencia	Valores clase de equivalencia	valores limite (ligero)	Conjetura de errores
Titulo	Titulo	Titulo existente InoExistente	"titulo"		""
p.isActivo()	boolean	true,false	true false		cadena de mas de 255 caracteres
p	Prestamo	null, Prestamo			
p.getEjemplar().getTitulo()	Titulo	active, no active	titulo 1, NULL	titulo Max_CARACTER	

D1 p.getEjemplar().getTitulo().equals (titulo
D2 p.isActivo && p.getEjemplar().getTitulo().equals(titulo)

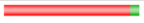
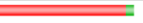






p.getEjemplar().getTitulo()	Título	p.isActivo	p	D1	D2
Título no existe	Título no existe	false	null	false	-
Título existente	Título existente	true	prestamo	true	true
Título no existe	Título no existe	false	active	false	-
Título existente	Título no existe	true	no active	false	-

7.2 Informes de análisis

Una vez hechos los tests unitarios podremos observar fácilmente los resultados gracias al uso de los informes de pruebas que obtendremos con el uso de la herramienta Surefire junto con JaCoCo. JaCoCo es una herramienta que analiza la cobertura en pruebas que se tienen y te genera un reporte en formato HTML en una manera similar a como funciona Javadoc. Mientras que surefire es un plugin se utiliza durante la fase de testing del ciclo de vida de compilación para ejecutar las pruebas unitarias de una aplicación. Genera informes en dos formatos de archivo diferentes: archivos de texto sin formato (*.txt), archivos XML (*.xml).

Este sería un ejemplo de obtención de informes de análisis con dicha herramienta:

Library

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
com.library.Library.controller		7%		6%	132 143	404 436	50 57	1 3
com.library.Library.entity		14%		0%	64 79	126 151	63 78	4 7
com.library.Library.service.impl		0%		0%	46 46	64 64	38 38	6 6
com.library.Library		0%		n/a	2 2	3 3	2 2	1 1
Total	2,436 of 2,637	7%	179 of 190	5%	244 270	597 654	153 175	12 17

8. Mantenibilidad

En este apartado vamos a hablar de como una vez desarrollado nuestro producto software conseguimos que este sea mantenible con el objetivo de detectar y eliminar posibles defectos que puedan producir diferentes tipos de fallos en nuestro sistema. Para elaborar esto llevaremos a cabo un plan de mantenimiento el cual establecerá una serie de pasos y herramientas a utilizar para lograr nuestro objetivo.

8.1 Plan de mantenimiento

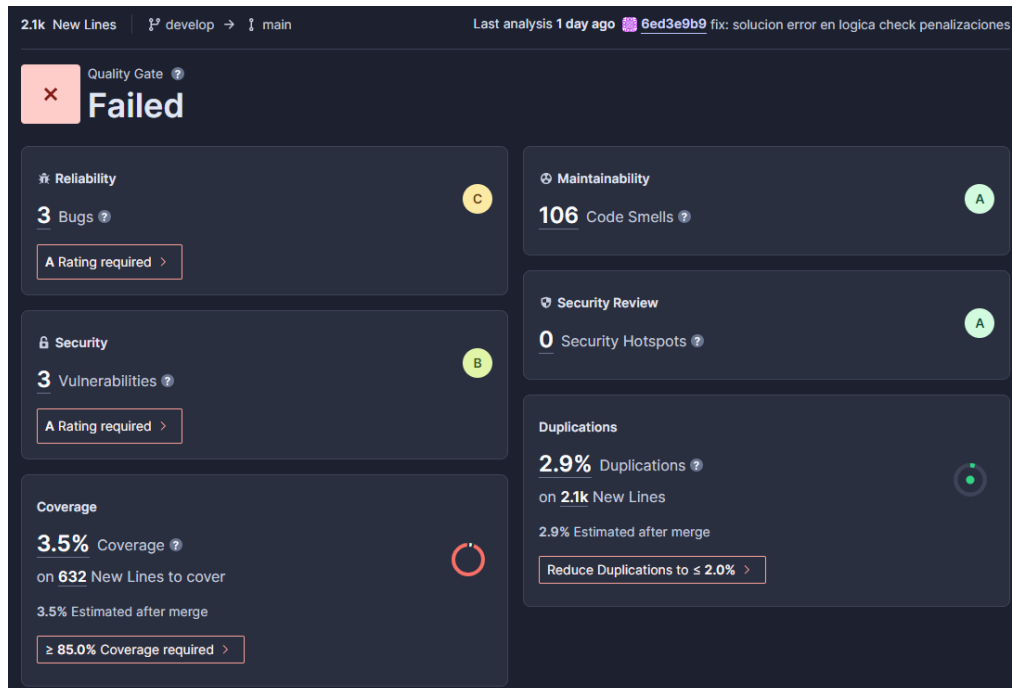
Objetivos del plan de Mantenimiento:

- Asegurar la estabilidad y rendimiento del sistema
- Mejorar y mantener la calidad del código
- Identificar y corregir posibles bugs o vulnerabilidades
- Adaptar el sistema a cambios en los requisitos o entorno tecnológico
- Garantizar la documentación actualizada y completa

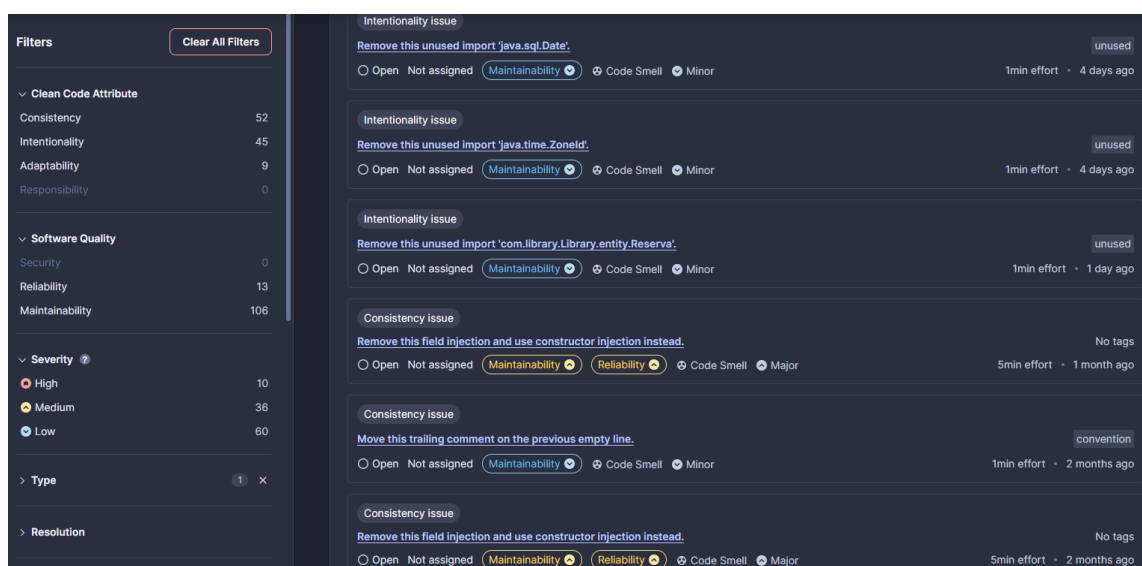
TIPOS DE MANTENIMIENTO UTILIZADOS

Mantenimiento correctivo:

Este tipo de mantenimiento lo abordaremos utilizando la ayuda de la herramienta Sonarcloud para detectar “bugs“, “code smells”, líneas duplicadas... y posteriormente corregirlos priorizando la corrección de bugs. Antes de tratar con el mantenimiento del programa esta fue la salida que nos otorgó sonar tras realizar un análisis en la rama develop:

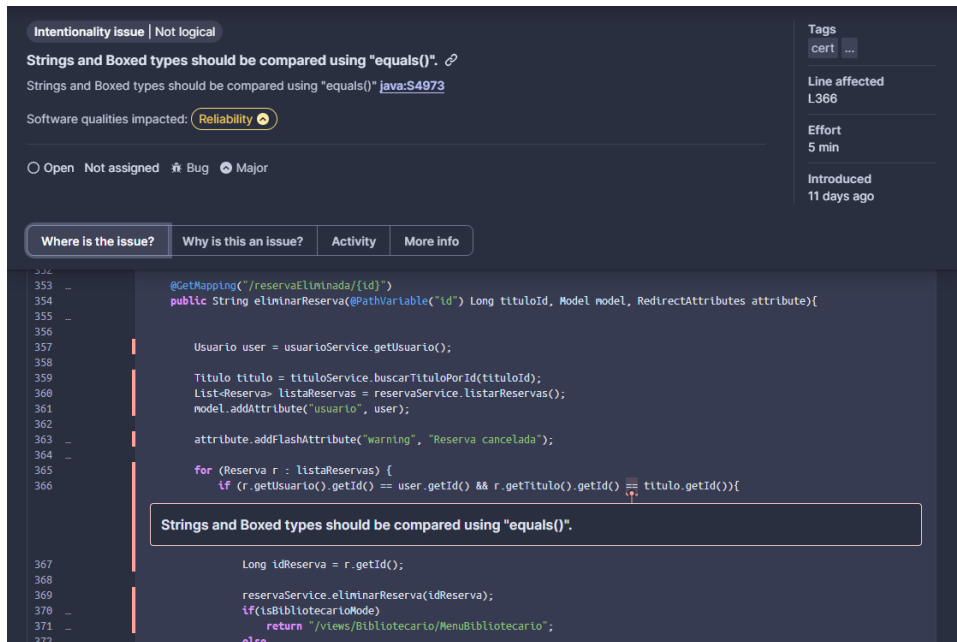


Para reducir estos problemas crearemos una rama en base a la analizada anteriormente llamada **mantenimiento**. En esta rama vamos a tratar de corregir y mejorar nuestro código gracias a los “tips” y consejos que nos ofrece sonar.



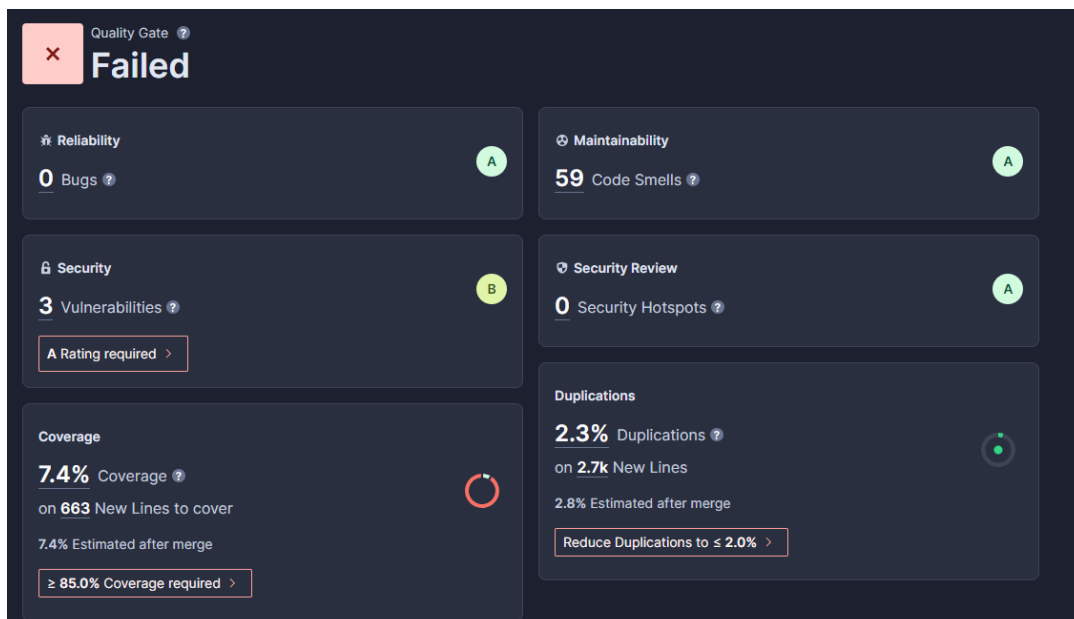
Este ejemplo de arriba nos ofrece una vista de cómo sería una lista con una serie de “code smells” (un code smells es un fragmento de código el cual muy probablemente de problemas a

futuro). Estos problemas los podemos escoger en función del tipo (problemas de fiabilidad, mantenibilidad o seguridad) o en relación a lo grave que sea este problema.



En este otro ejemplo podemos ver lo que nos ofrece sonar cuando seleccionamos un problema, en este caso un bug. En nuestra rama de **mantenimiento** podemos acceder directamente a la línea donde está el problema y sonar nos explica por qué lo que nos señala es un problema y en algunas situaciones como podríamos abordarlo.

A continuación vamos a mostrar un análisis tras abordar los problemas que nos ha avisado sonar que tenemos en nuestro proyecto:



Podemos observar como la quality gate ha fallado debido a que sigue habiendo algunas fallas de seguridad. No obstante, hemos conseguido reducir en gran parte los code smells así como reducir a 0 el número de bugs e incrementar la cobertura con los tests realizados.

Mantenimiento preventivo:

Mediante revisiones periódicas (semanales) para identificar y corregir posibles problemas mejorando la calidad del código. Los criterios utilizados para identificar posibles problemas fueron, buscar entradas de texto que tiene que escribir el usuario, variables... Una vez identificados los problemas proponemos diversas posibles soluciones para elegir posteriormente la óptima.

Mantenimiento adaptativo:

Durante el transcurso del desarrollo del proyecto se han propuesto múltiples herramientas nuevas las cuales fueron investigadas e implementadas en el proyecto.

Respecto al mantenimiento continuamos con la mejora del coverage y revisar el código para perfeccionarlo lo máximo posible. Hemos aprendido que las tareas de mantenimiento son muy importantes para tener un código de calidad al igual que también hay que destacar el saber priorizar estas tareas. Por último, descubrimos la importancia de organizar al equipo y tener una comunicación estrecha y precisa para garantizar esta calidad.