



Universidad
Zaragoza

Laboratorio de Ingeniería Software

Grupo 01: Entrega parcial del
trabajo de la asignatura

David Mañas Vidorreta
Alberto Martínez Menéndez
Alejandro Navarro Castillo
Ian Paul Ávila Matos
Óscar Saboya Gómez
11-4-2018

Contenido

1	Introducción	2
1.1	El equipo	2
1.2	Estructura del documento	3
2	Producto.....	4
2.1	Objetivos y planificación de lanzamientos.....	4
2.2	Entradas de la pila de producto	4
2.3	Definición de hecho	6
2.4	Estado actual de la aplicación	7
2.5	Arquitectura y diseño dirigido por el dominio	7
2.6	Estrategia y herramientas utilizadas para el testeo de la aplicación.....	9
2.7	Construcción automática del software	10
3	Proceso.....	11
3.1	Estrategia de control de versiones.....	11
3.2	Estadísticas.....	12
3.3	Herramientas utilizadas	12
4	Conclusiones	13
4.1	Grado de cumplimiento de los objetivos	13
5	Enlaces y referencias.....	15
6	Anexos.....	16
6.1	Anexo 1	16

1 INTRODUCCIÓN

Este proyecto es parte del contenido de la asignatura Laboratorio de Ingeniería del Software, obligatoria dentro de la rama de Ingeniería del Software del Grado de Ingeniería Informática de la Universidad de Zaragoza.

La base de este radica en la creación de una plataforma de mantenimiento desde la cual los usuarios (alumnos, profesores, invitados, etc.) pueden indicar las averías que surjan en el campus de la EINA, facilitando su ubicación y reparación. Dichas incidencias serán controladas por un administrador para evitar un uso abusivo y así hacérselas llegar al equipo de mantenimiento, el cual podrá asignárselas para resolver en los horarios convenientes, donde no interfieran con la utilización del aula o el espacio en el que se haya producido la incidencia.

1.1 EL EQUIPO

El equipo del proyecto está formado por los cinco estudiantes que componen el grupo 1 de la asignatura Laboratorio de Ingeniería Software:

- David Mañas Vidorreta
- Alberto Martínez Menéndez
- Alejandro Navarro Castillo
- Ian Paul Ávila Matos
- Óscar Saboya Gómez

Se ha decidido seguir los principios de la metodología ágil Scrum debido a que varios miembros del equipo ya trabajaron con ella el cuatrimestre anterior en otra asignatura y porque se ha considerado que encaja bien en el tipo de proyecto a realizar. Dentro de los roles que Scrum propone se ha realizado el siguiente reparto:

- ScrumMaster: Alberto.
- Dueño de producto: Alejandro.
- Equipo de desarrollo: David, Alberto, Alejandro, Ian Paul, Óscar.

Los roles de ScrumMaster y dueño de producto se han asignado a los dos estudiantes que ya trabajaron con Scrum en el cuatrimestre previo.

1.2 ESTRUCTURA DEL DOCUMENTO

El documento consta de cuatro partes:

- **Introducción:** se hace una breve descripción del proyecto y del equipo.
- **Producto:** se tratan temas relacionados con los requisitos de la plataforma, el estado actual en el que se encuentra con respecto a la versión final prevista, la arquitectura, el diseño y aspectos sobre implementación y despliegue, aspectos relacionados con SIG y el Diseño Dirigido por el Dominio y, pruebas y tests automáticos implementados y, por último, posibles medidas utilizadas para mejorar la calidad del producto final.
- **Proceso:** Explica el sistema de control de versiones utilizado para gestionar la organización y realización del trabajo de forma colaborativa, los esfuerzos realizados por cada persona, así como las actividades realizadas, diagramas de reparto en el tiempo y todas las herramientas utilizadas para la realización de lo mencionado anteriormente.
- **Conclusiones:** Incluye un resumen final del documento, así como una evaluación del cumplimiento de los requisitos de la asignatura, indicando en qué grado están cubiertos, junto con una explicación, así como, en caso de ser necesario, indicar las herramientas que permiten comprobar que dicho porcentaje corresponde con la realidad.

2 PRODUCTO

2.1 OBJETIVOS Y PLANIFICACIÓN DE LANZAMIENTOS

El proyecto se ha dividido en dos sprints o iteraciones. Con el fin de conseguir que las dos iteraciones tuvieran la misma duración no ha sido posible acabar la primera iteración en la fecha establecida, por lo que posibles elementos de este documento estén incompletos. Por ello, si al profesor le parece correcto, se entregará el documento completamente actualizado cuando el primer sprint finalice. Las fechas de los dos sprints son las siguientes:

- Primer sprint: del 27 de marzo de 2018 hasta el 21 de abril de 2018.
- Segundo sprint: del 22 de abril de 2018 hasta el 17 de mayo de 2018.

Durante el primer sprint se abarcará la puesta a punto de la estructura del proyecto, así como de las herramientas necesarias para su desarrollo. Dichas herramientas se encuentran especificadas en el punto 3.3 de este mismo documento. Además, se han seleccionado para esta primera iteración las entradas de la pila que se han considerado como base para el desarrollo de tareas posteriores, como por ejemplo la gestión de usuarios, las pantallas principales, la visualización del mapa y de las capas geográficas o el diseño e implementación del modelo de dominio.

En la segunda iteración se abarcarán el resto de las funcionalidades y características. Como creación, modificación, aceptación, rechazo y asignación de incidencias por parte de los respectivos usuarios a los que pertenezca cada una de las acciones nombradas.

2.2 ENTRADAS DE LA PILA DE PRODUCTO

A partir de la recolección de historias de usuario llevada a cabo durante varias reuniones con el profesor encargado de la asignatura se han generado las siguientes entradas de pila.

Product Backlog Items (PBIs)	Descripción	Estimación en PH	Sprint
Login general	Permitir que cualquiera de los tres tipos de usuarios pueda iniciar sesión en la aplicación.	5	1
Registro usuario básico	Permitir que el usuario básico se registre en la aplicación.	8	1
Preparar mapa base (Leaflet)	La aplicación mostrará un mapa en el que se podrán visualizar los tres edificios del campus EINA.	5	1

Definir modelo de dominio	Definir correctamente todos los elementos que constituyen el modelo de dominio de nuestra aplicación e implementarlo.	13	1
Añadir datos geográficos (espacios)	Añadir capas geográficas al mapa para visualizar los distintos espacios de los tres edificios del campus EINA.	13	1
Asignar horario para espacio (administrador)	Permitir que el administrador pueda modificar el horario de utilización de un espacio mostrado en el mapa.	8	1
Ver información extra en el mapa: horarios, uso del aula, etc.	Visualizar información extra sobre el espacio seleccionado como nombre, tipo, horario, etc.	8	1
Ver plan de asignación de trabajo (mantenimiento)	Permitir al personal de mantenimiento ver el plan de trabajo con las incidencias que se ha asignado.	5	2
Marcar incidencia como terminada (mantenimiento)	Marcar una incidencia como finalizada.	3	2
Marcar incidencia como empezada (mantenimiento)	Marcar una incidencia como empezada.	3	2
Asignar incidencia eligiendo horas de trabajo (mantenimiento)	Asignarse una incidencia eligiendo las horas en las que se resolverá.	5	2
Ver incidencias aceptadas (mantenimiento)	Ver lista de incidencias aceptadas y no asignadas.	5	2
Ver horario del espacio asignado a una incidencia (mantenimiento)	Ver el horario completo del espacio donde se ha aceptado una incidencia con el fin de poder seleccionar la hora a la que se resolverá.	5	2
Pedir clarificar incidencia (administrador)	Pedir al usuario que clarifique una incidencia cuya descripción sea insuficiente o confusa.	3	2
Rechazar incidencia (administrador)	Rechazar incidencia propuesta.	3	2
Aceptar incidencia (administrador)	Aceptar incidencia propuesta.	3	2

Filtrar incidencias (administrador)	Filtrar la lista de incidencias por aula, por estado de las mismas, etc.	5	2
Ver incidencias pendientes, admitidas, completadas, etc. (administrador)	Ver lista completa de todas las incidencias.	5	2
Notificar incidencias: admitida, rechazada, insuficiente (básico)	Notificar al usuario de cualquier cambio en el estado de una de sus incidencias.	5	2
Ver incidencias activas (básico)	Ver el total de incidencias aceptadas.	5	2
Ver incidencias creadas (básico)	Ver el total de incidencias creadas por el usuario.	5	2
Crear incidencia (básico)	Crear y proponer incidencia.	13	2
Modificar incidencia creada (básico)	Modificar alguno de los campos de la incidencia propuesta.	8	2

Los bocetos de la GUI realizados en una primera instancia para crear una visualización común sobre como sería a grandes rasgos la interfaz de la aplicación pueden verse en el anexo 1.

2.3 DEFINICIÓN DE HECHO

La definición de hecho recoge las condiciones que han de cumplirse para que una tarea pueda considerarse hecha o completada. A continuación, se expone la definición de hecho elegida para este proyecto.

Se considera que una entrada de la pila de producto está hecha si:

- Todas las tareas en las que se ha dividido la entrada están completadas.
- Se han escrito y ejecutado tests unitarios con al menos un 50% de cobertura de código.
- Se ha comprobado que, al terminar las tareas de una entrada, el resto de tests siguen dando resultados satisfactorios.
- Se ha realizado una documentación de mínimos del diseño e implementación del código de cara a entender el funcionamiento de este, mediante especificaciones en el código y diagramas de diseño e implementación.
- El correcto funcionamiento de la aplicación web se ha probado tanto mediante la ejecución de los tests automáticos desarrollados como mediante interacción manual.
- Se ha comprobado el correcto funcionamiento de la aplicación web en al menos los siguientes navegadores: Google Chrome, Mozilla Firefox.
- La GUI satisface al dueño del producto.

2.4 ESTADO ACTUAL DE LA APLICACIÓN

Actualmente la aplicación se encuentra en medio de la ejecución del primer sprint. Se han implementado las funcionalidades de gestión de usuarios como registro de usuarios básicos y el inicio de sesión con una cuenta de cualquiera de los tres tipos disponibles (básico, administrador y mantenimiento). Además, se ha comenzado con la implementación de la capa de dominio, con las entidades que representan los espacios, el repositorio de espacios y los objetos valor que representan los horarios de los espacios.

Respecto a la GUI, se han implementado las pantallas de registro e inicio de sesión. También se ha conseguido incluir el mapa de OpenStreetMaps en la pantalla correspondiente. La lógica de añadido de capas e interacción entre las mismas está en su mayor parte implementada pero no ha sido enlazada con la vista del mapa. Además, la conexión del controlador del mapa con el servidor QGIS esta pendiente de ser terminada en breve.

2.5 ARQUITECTURA Y DISEÑO DIRIGIDO POR EL DOMINIO

Para el desarrollo de esta aplicación web se está siguiendo una arquitectura por capas. En la segunda iteración del proyecto, cuando el modelo de dominio este definido de una forma concreta y definitiva, el equipo se enfrentará al objetivo de seguir una arquitectura hexagonal.

Actualmente los elementos que componen cada una de las capas pueden verse en el diagrama de componentes y conectores de la Ilustración 1 [\[1\]](#).

En la capa de interfaz de usuario se encuentran los ficheros html que componen las páginas de la aplicación web, así como sus respectivos controladores. En la capa de aplicación podemos distinguir dos partes. Por un lado, en el lado del cliente, tenemos los servicios que se encargan de generar las peticiones HTTP necesarias para comunicarse con el servidor REST o directamente con el servidor QGIS. Por otro lado, tenemos la API del servidor, formada por varios controladores REST que se encargan de recibir y tratar las peticiones emitidas desde el cliente. En la capa de dominio tenemos los repositorios encargados de gestionar las operaciones de los objetos del dominio (como entidades, objetos valor, etc.). Dichos repositorios acceden a la base de datos PostgreSQL mediante JDBC. La base de datos esta situada en la capa de infraestructura de la aplicación.

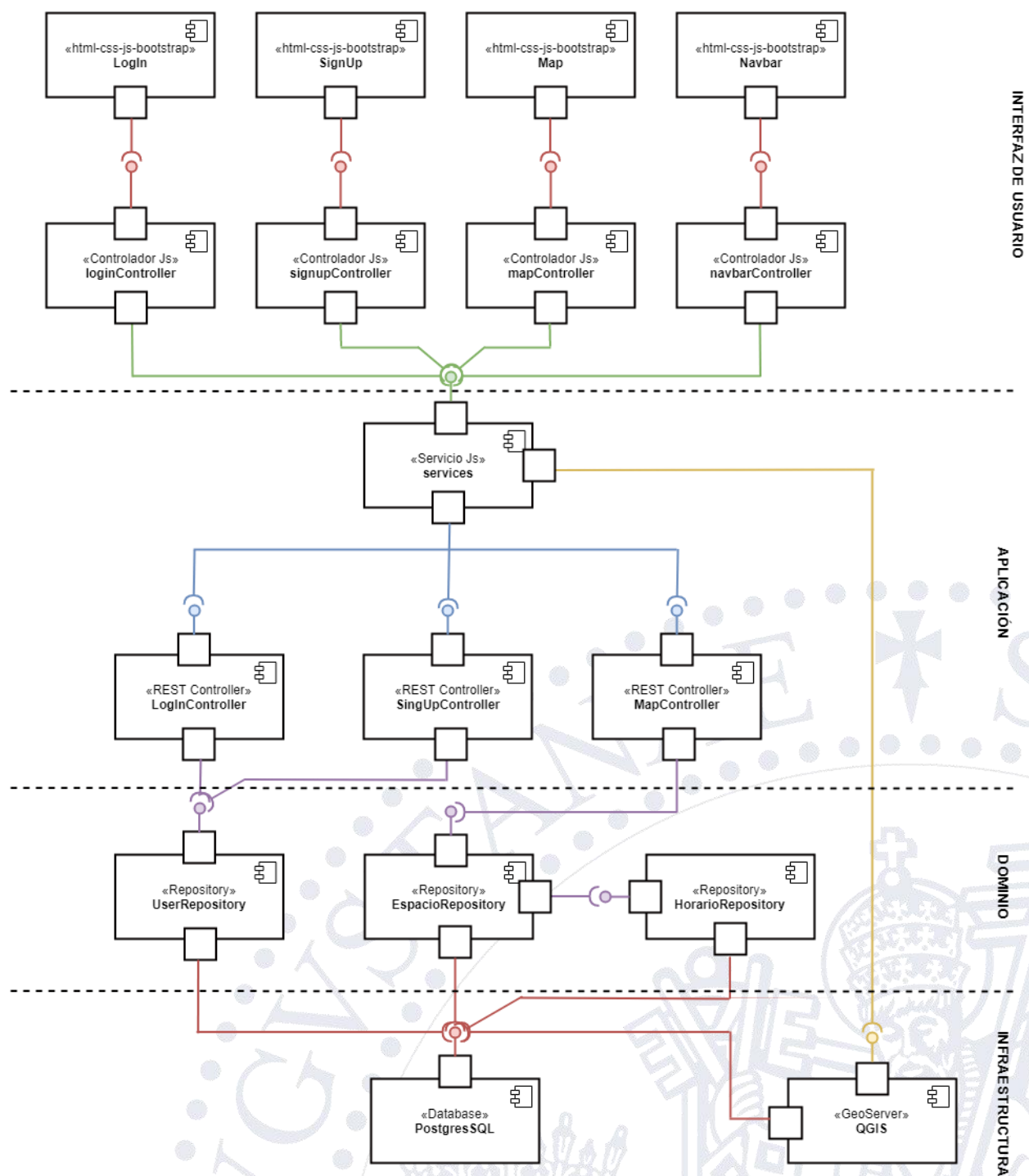


Ilustración 1 - Diagrama de componentes y conectores

El esquema del modelo de dominio está actualmente pendiente de ser concretado en su totalidad y será añadido posteriormente.

En el siguiente diagrama de despliegue, en la Ilustración 2 [2] puede observarse como se han dispuesto los diferentes elementos que componen la aplicación.

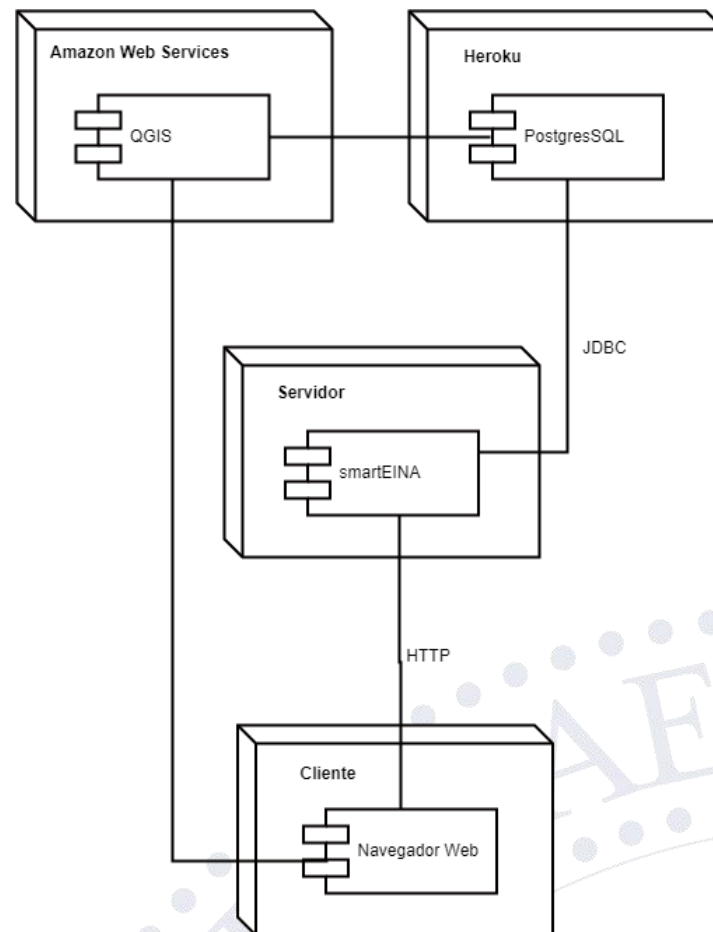


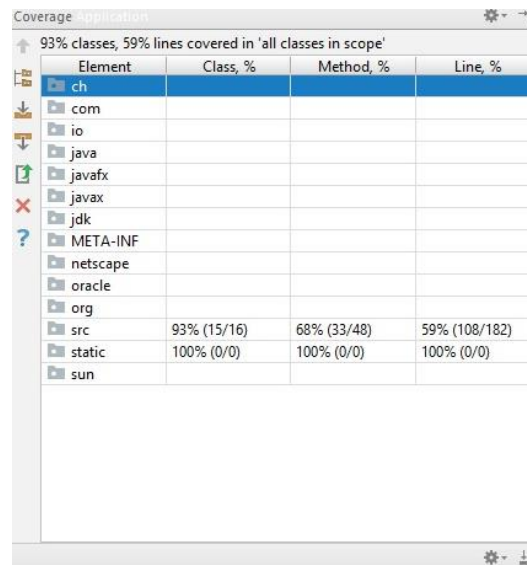
Ilustración 2 - Diagrama de despliegue

2.6 ESTRATEGIA Y HERRAMIENTAS UTILIZADAS PARA EL TESTEO DE LA APLICACIÓN

El testeo de la GUI será realizado de forma manual mediante una serie de tests de aceptación realizados por el equipo de desarrollo.

Los tests unitarios serán realizados de manera automática utilizando la herramienta de testeo Junit. El objetivo es que por cada función implementada exista un test unitario que compruebe su correcto funcionamiento. En el momento de la redacción de este documento, la cobertura de tests es del 59%. Este valor ha sido sacado de la opción de cálculo de *coverage* proporcionada por el IDE IntelliJ. Debido al estado actual de la aplicación las funciones testeadas son básicamente aquellas que interaccionan con la base de datos remota o crean objetos de dominio.

Actualmente estos tests unitarios son llamados manualmente por los desarrolladores. Durante la segunda iteración del proyecto se añadirá la herramienta Travis CI, que permite automatizar la ejecución de tests de manera que estos se lancen y se comprueben cada vez que hay un cambio en el código del repositorio de GitHub.



Coverage application

93% classes, 59% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
ch			
com			
io			
java			
javafx			
javax			
jdk			
META-INF			
netscape			
oracle			
org			
src	93% (15/16)	68% (33/48)	59% (108/182)
static	100% (0/0)	100% (0/0)	100% (0/0)
sun			

Ilustración 3 - Cobertura de código en el momento de la redacción del documento

2.7 CONSTRUCCIÓN AUTOMÁTICA DEL SOFTWARE

Para la construcción automática del software se han utilizado dos herramientas. Maven para la gestión de dependencias del proyecto, así como para la construcción del propio proyecto. Spring, con su modulo *Spring-boot* para el despliegue de la aplicación web de forma rápida y simple durante la fase de desarrollo. En la segunda iteración del proyecto, como ya se ha comentado en el punto anterior, se añadirá la herramienta Travis CI para la ejecución automática de tests.

3 PROCESO

3.1 ESTRATEGIA DE CONTROL DE VERSIONES

Para el control de versiones se ha decidido utilizar Github (git) ya que es la plataforma de desarrollo colaborativo con la que los miembros del equipo están más familiarizados. Se ha creado una organización llamada UNIZAR-30249-2018-Labis en la que se encuentran todos los integrantes de este equipo tal y como se pedía en las instrucciones del trabajo. En un primer momento la idea era tener tres repositorios dentro de la organización:

- labis-documents: para los documentos del proyecto.
- labis-client: para el código de la parte del cliente de la aplicación web.
- labis-server: para el código de la parte del servidor de la aplicación web.

Esta separación entre parte del cliente y parte del servidor fue descartada al poco tiempo por las innecesarias complicaciones que creaba a la hora de preparar el proyecto en el entorno de desarrollo, así como a la hora de probar la aplicación.

Actualmente, los repositorios existentes son los siguientes:

- labis-documents: para los documentos del proyecto.
- smartEina: para el código de la aplicación web (cliente y servidor).

Además, se ha utilizado la herramienta de tablero de tareas (o tablero de proyecto) proporcionada por Github para que los miembros del equipo se sincronicen y puedan llevar un control de las tareas pendientes, de las tareas que están siendo desarrolladas y de las tareas completadas, tanto a nivel global de proyecto como a nivel de sprint.

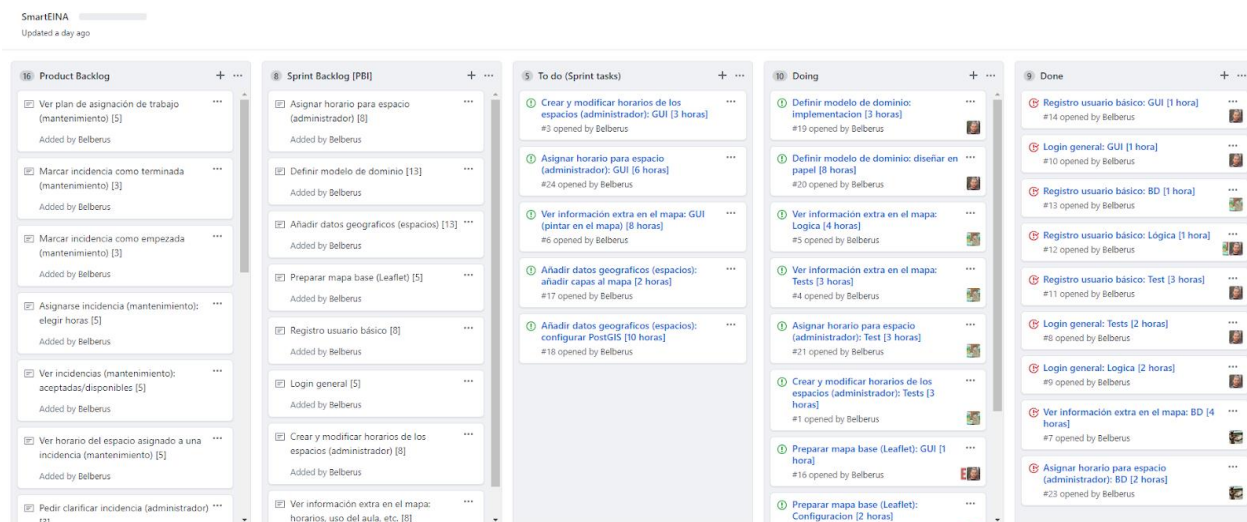


Ilustración 4 - Tablero de trabajo del proyecto smartEina en GitHub.

Respecto al tipo de *workflow* utilizado, se ha optado por un flujo de trabajo simple. Cada miembro del equipo se clona directamente los repositorios de la organización, trabaja en local sobre ellos y “pushea” cualquier cambio realizado, actualizando el repositorio maestro.

Se descartó la opción de utilizar un flujo de trabajo basado en *forks* y *pull-requests* ya que añaden innecesaria complejidad al proyecto y des-agilizan la colaboración entre los integrantes del equipo.

3.2 ESTADÍSTICAS

Como ya se ha comentado en la introducción, el equipo sigue los principios de la metodología ágil Scrum. Se ha dividido el proyecto en dos iteraciones (sprints) con las siguientes fechas para que sean de igual duración, como ya se ha visto en el punto 2.1:

- Primer sprint: del 27 de marzo de 2018 hasta el 21 de abril de 2018.
- Segundo sprint: del 22 de abril de 2018 hasta el 17 de mayo de 2018.

Como el día de entrega de este documento es diez días antes del elegido para la finalización del primer sprint, ni las horas ni el trabajo pendiente o completado reflejan realmente el desarrollo de esta primera iteración y por tanto se ha decidido no incluir los diagramas que representan estos elementos (horas trabajadas, burnup, burndown), al no proporcionar información útil.

No obstante, si al profesor le parece correcto, el equipo le hará llegar estos diagramas una vez finalizado el primer sprint, aunque no haya una entrega de material asignada a dicha fecha.

3.3 HERRAMIENTAS UTILIZADAS

Hasta el momento de redacción de este documento, las herramientas utilizadas para el proyecto han sido las siguientes:

Herramienta	Descripción
IntelliJ IDEA	Entorno de desarrollo utilizado por ciertos miembros del equipo.
Netbeans	Entorno de desarrollo utilizado por ciertos miembros del equipo.
Heroku	Plataforma Cloud que almacena la base de datos PostgreSQL del proyecto.
PGAdmin	Herramienta de acceso a bases de datos con interfaz de usuario.
SQLPro for Postgres	Herramienta de acceso a bases de datos con interfaz de usuario.
Amazon Web Services	Plataforma Cloud que aloja al servidor de QGIS encargado de proporcionar datos geográficos a la aplicación.
GitHub	Sistema de desarrollo colaborativo utilizado para el control de versiones tanto de código como de documentos.
Junit	Librería para creación y ejecución de tests unitarios automáticos en el ámbito del desarrollo.
Travis CI	Complemento para GitHub que permite la ejecución automática de los tests de la aplicación cada vez que se actualice el código del repositorio.

QGIS Server	Herramienta encargada de crear servicios de mapas web (WMS) a partir de los datos proporcionados por la base de datos PostgreSQL.
Postman	Herramienta utilizada para generar peticiones a la API de nuestro servidor con el fin de probarla.
Spring	Framework para el desarrollo de aplicaciones que facilita, entre otras cosas, la inyección de dependencias y el despliegue de aplicaciones durante la fase de desarrollo.
AngularJS	Framework para el desarrollo de Single Page Applications. Utilizado en el lado del cliente para facilitar el control de la GUI.
VNC Viewer	Herramienta utilizada para acceder de forma remota a la máquina desplegada en Amazon Web Services mediante una interfaz gráfica.
Hibernate	Herramienta ORM que facilita el desarrollo de las funcionalidades de persistencia de datos.

4 CONCLUSIONES

En el momento de redacción de este documento el equipo se encuentra a 10 días de la finalización del primer sprint del proyecto. Es por esto por lo que los datos mostrados a lo largo del documento no reflejan actualmente los resultados esperados tras la primera iteración. Como se ha comentado anteriormente, en caso de ser necesario, se realizará una actualización del presente documento una vez terminado el sprint con el fin de mostrar los datos de una manera más precisa y útil.

4.1 GRADO DE CUMPLIMIENTO DE LOS OBJETIVOS

En este apartado se muestra el grado actual de cumplimiento de los objetivos para la evaluación del proyecto.

Nivel aprobado			
Objetivo	Estado actual	Comentarios	Enlaces
Código y documentación en GitHub.	Completado.	Todo el código y documentación del proyecto se encuentra alojado en los repositorios de la organización en GitHub desde el comienzo.	[3]
Compilación y gestión de dependencias basado en Scripts.	Completado.	Se está utilizando la herramienta Maven para la compilación automática y la gestión de dependencias del proyecto.	[4]

Control de esfuerzos.	Completado.	Hasta la fecha se lleva al día la cuenta de horas dedicadas al proyecto y se han realizado las entregas de estos cuando han sido solicitadas.	[5]
Cumplimiento de requisitos.	En proceso.	En este momento se están desarrollando los requisitos asignados al primer sprint.	-
Documentación adecuada.	Completado.	Toda documentación necesaria como actas de reuniones, tareas, diagramas, etc. está disponible en el repositorio de documentos de GitHub.	[5]
Arquitectura por capas.	Completado.	Tal y como se explica en el apartado 2.5 la arquitectura sigue una división por capas.	(2.5)
Diseño dirigido por el dominio (entidades, objetos valor, agregados, factorías y repositorios).	En proceso.	En este punto del desarrollo todavía no se han utilizado algunos de los elementos descritos en el objetivo, sin embargo, a medida que avance el proyecto y se aclare el modelo de dominio serán incluidos.	(2.5)
Modificación de datos concurrentes.	No realizado.	En el estado actual del proyecto no ha sido necesario gestionar la modificación de datos concurrentes. En el segundo sprint se abarcará este objetivo.	-
En marcha y con servicio tipo WMS con mapas superpuestos.	En proceso.	Algunos elementos de la aplicación como la gestión de usuarios ya están en funcionamiento, pero otros como la visualización de las capas sobre el mapa serán acabados en breve.	[4]

Nivel notable			
Objetivo	Estado actual	Comentarios	Enlaces
Cobertura de tests automáticos de mínimo 30%	Completado.	Actualmente la cobertura de nuestro código es del 59% tal y como puede verse en el apartado 2.6.	(2.6)
Diseño dirigido por el dominio (servicios, paquetes, interfaces reveladoras, aserciones, funciones libres de	En proceso.	En este punto del desarrollo no se han utilizado todos los componentes descritos en el objetivo, sin embargo, en la segunda iteración, conforme el modelo de dominio se aclare, aparecerán.	(2.5)

efectos secundarios).			
Estilo cartográfico refleja cada tipo de uso de un espacio.	En proceso.	Actualmente se están preparando las capas de los diferentes pisos de los tres edificios para ser incluidos en el mapa.	-

Nivel sobresaliente			
Objetivo	Estado actual	Comentarios	Enlaces
Cobertura de tests automáticos de mínimo 50%	Completado.	Actualmente la cobertura de nuestro código es del 59% tal y como puede verse en el apartado 2.6.	(2.6)
Arquitectura hexagonal	No realizado.	Este objetivo se deja para la segunda iteración, cuando el modelo de dominio este mejor definido. El equipo intentará completar este objetivo.	-

5 ENLACES Y REFERENCIAS

[3] Página de GitHub de la organización con los repositorios de código y documentación:

<https://github.com/UNIZAR-30249-2018-Labis>

[4] Repositorio de código con fichero de Maven, pom.xml, y el resto de los elementos:

<https://github.com/UNIZAR-30249-2018-Labis/smartEina>

[5] Repositorio de documentación: <https://github.com/UNIZAR-30249-2018-Labis/labis-documents>

6 ANEXOS

6.1 ANEXO 1

Bocetos de la GUI.

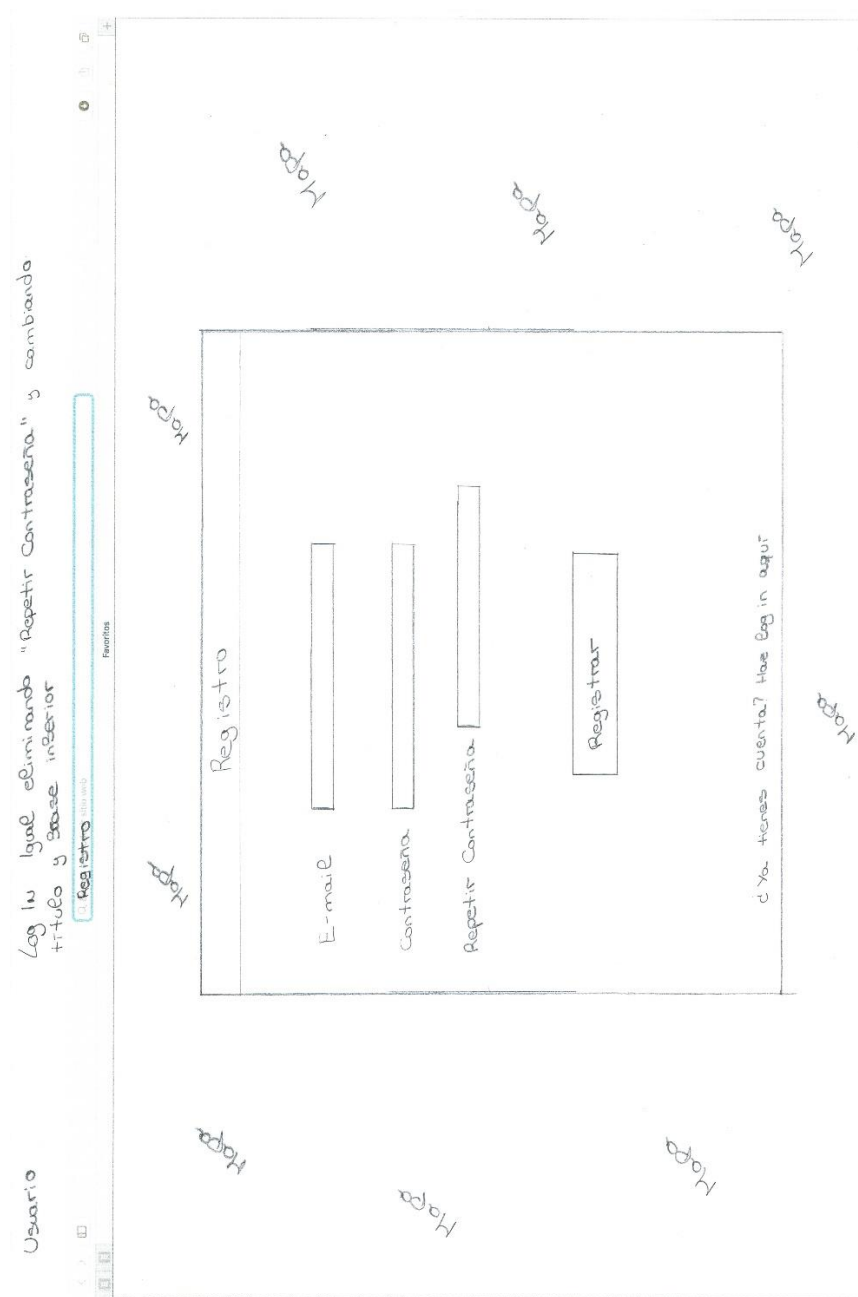


Ilustración 5 - Boceto GUI pantalla de registro

Usuario

Atadir Incidencia

Atadir Incidencia Aula 001

Controles de Haza

Enviar

Descripción

Ubicación

Título

Incidencias A.O.01	
1.	Let. Sordida
2.	
3.	
4.	
5.	
6.	
7.	

Ilustración 7- Boceto GUI añadir incidencia

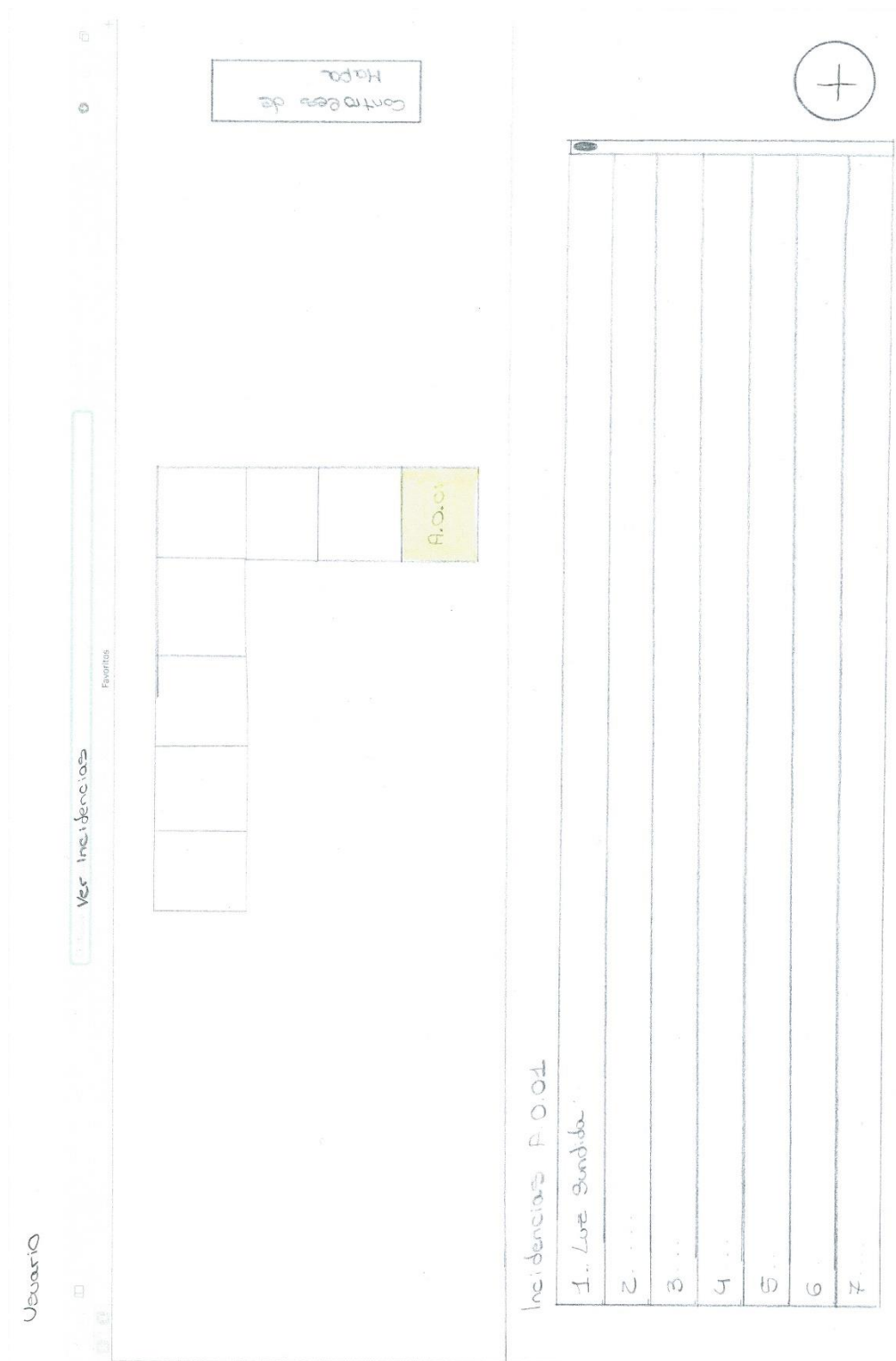


Ilustración 8 - Boceto GUI de ver incidencias

[illegible]

Ilustración 9 - Boceto GUI del mapa del administrador

Mantenimiento

Ver incidencias aceptadas

Favoritos

Controles de Mapa

A.O.01

Incidencias

1. Título, Ubicación, Descripción	<input checked="" type="checkbox"/>
2. ...	<input type="checkbox"/>
3. ...	<input type="checkbox"/>
4. ...	<input type="checkbox"/>
5. ...	<input type="checkbox"/>
6. ...	<input type="checkbox"/>
7. ...	<input type="checkbox"/>

Ilustración 11 - Boceto GUI de ver incidencias aceptadas