

Informe final del proyecto

Septiembre

1. Introducción

El proyecto desarrollado es una versión reducida del trabajo solicitado en Junio.

Se ha implementado la versión “Mantenimiento del campus” aplicando el diseño dirigido por dominio y utilizando Sistemas de información Geográfica.

1.1. Equipo

El trabajo se ha desarrollado de forma individual por :

- González, Javier: estudiante del Grado de Ingeniería Informática.

2. Producto

- Planificación proyecto

El desarrollado del proyecto se ha desarrollado en el mes de agosto. Logrando desarrollar todos los requisitos requeridos.

2.2. Requisitos

Los requisitos para la versión “Mantenimiento del campus” son los siguientes:

Roles de Usuario

- Usuario **normal**: Sin registrarse puede acceder a la aplicación, consultando incidencias, consultando horarios de las clases y reportando incidencias cuando lo crea oportuno.
- Usuario **administrador**: Usuario con más privilegios que usa la aplicación para tratar las incidencias, aceptando o descartando, para su futura asignación y elaboración.

Usuario normal

- El usuario es capaz de insertar incidencias en la localización elegida.
- El usuario es capaz de visualizar los horarios de actividades en las distintas localizaciones.
- El usuario es capaz de visualizar las incidencias en proceso de resolución y las incidencias completadas.

Usuario administrador

- El administrador es capaz de ver el mapa para informarme de las incidencias de las distintas localizaciones.
- El administrador es capaz de ver incidencias pendientes para informarme de nuevas incidencias sin resolver.
- El administrador es capaz de cancelar las incidencias para su futura eliminación.
- El administrador es capaz de asignar las incidencias para su futura resolución.
- El administrador señala cuando las incidencias han sido completadas.

Dichas historias se han usado también para crear una pila de producto que se gestiona mediante issues y proyectos de GitHub.

2.3. Entradas de la pila de producto

1.1.1. Pila del proyecto

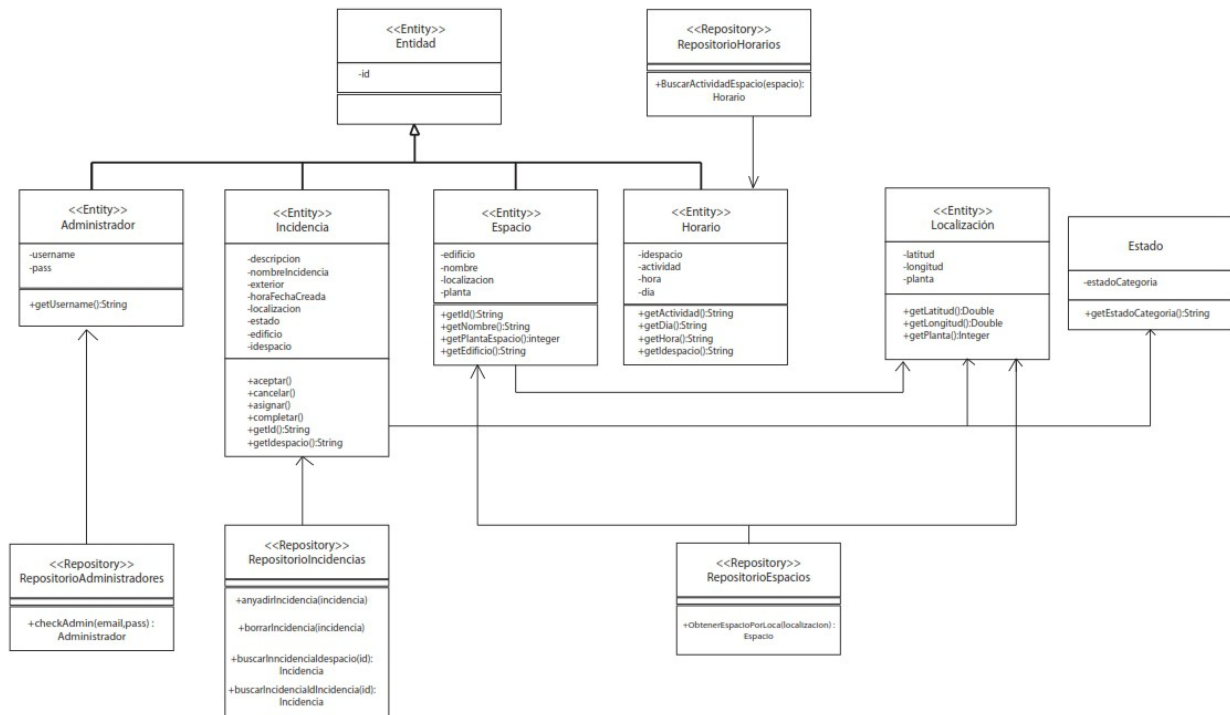
Entrada:Tarea	Horas	Estimación
Mapa de la aplicación	20	15
Preparar Geoserver	16	10
Dar estilo capas	7	5
Filtrar datos	6,5	6,5
Crear incidencias	6	4
Borrar incidencias	6	4
Aceptar incidencias	7	4
Asignar incidencias	3	4
Cancelar incidencias	3	4
Completar incidencias	3	4
Login Administrador	12	4
Logout Administrador	3	1
Interfaz Usuario	10	10
Interfaz Administrador	10	10
Buscar Horarios	6	3
Buscar información espacios	10	3
Documentación	15	6
Testing	12	6
Total	155,5	103,5

2.4. Definición de hecho

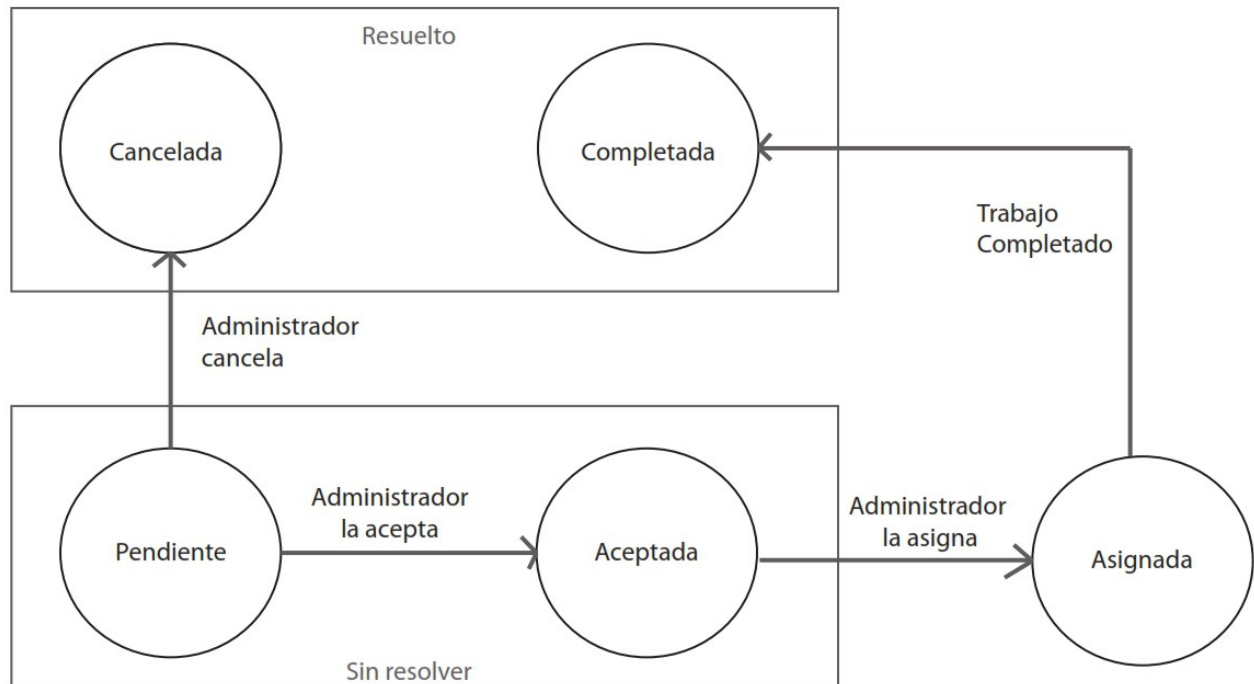
- La aplicación supera los tests automáticos especificados.
- Existe un manual de usuario que recoge las funcionalidades de la aplicación.
- Redacción de documentación técnica de la aplicación.

2.5. Diseño dirigido por el dominio

Se ha realizado un diagrama de clases para mostrar como está formado el la capa de dominio.



Estados en los que puede estar una incidencia:



2.6. Análisis de riesgos

Se realiza un análisis de riesgos con los siguientes resultados:

- Baja Seguridad en los datos de nuestra web.
- Baja experiencia en uso de mapas WMS
- Baja experiencia en el uso del diseño aplicado.

2.7. Arquitectura y Diseño

La aplicación consta de una arquitectura cliente/servidor de 3 niveles que consta de :

2 Servidores web: - Uno que proporciona la API de la aplicación.

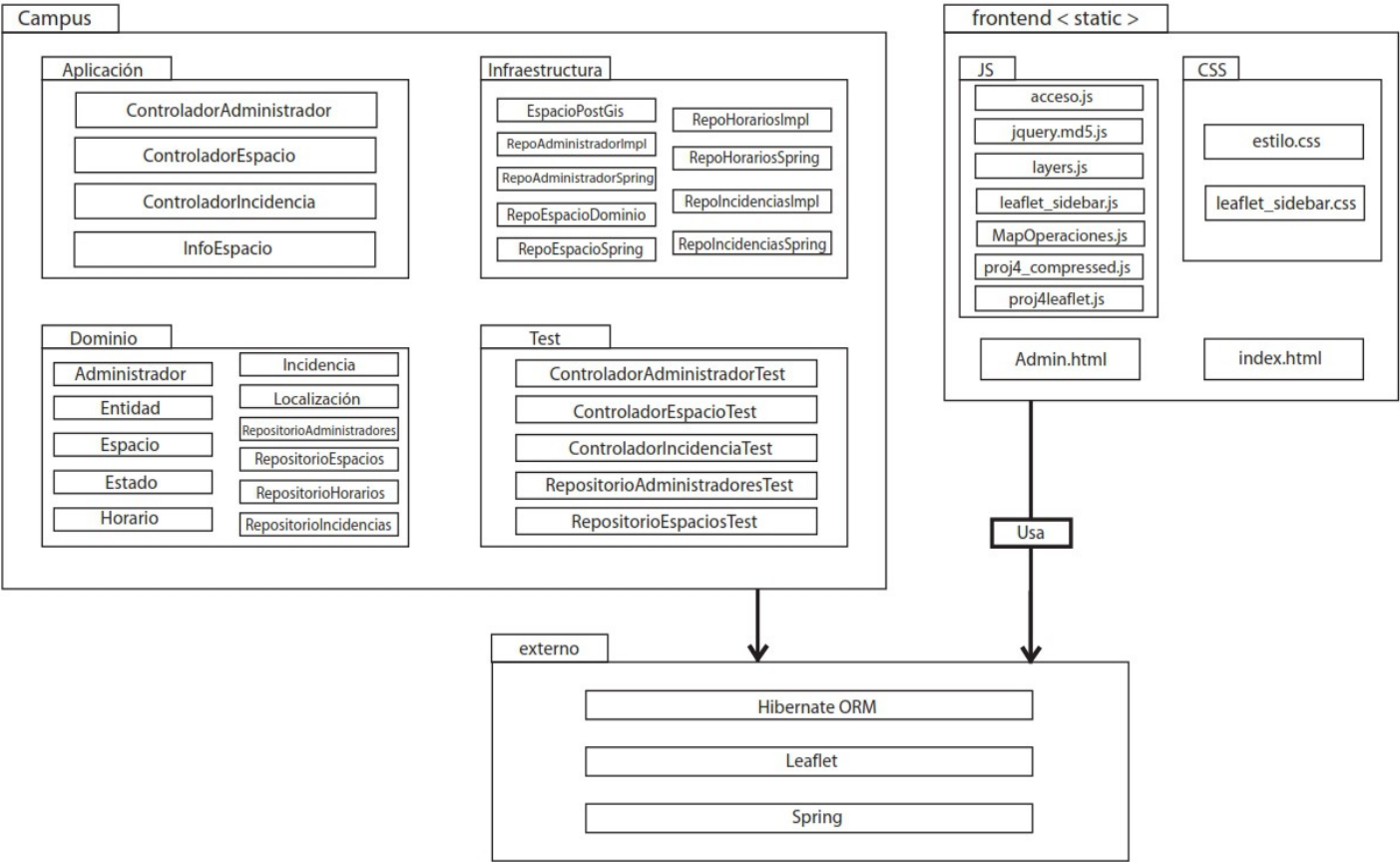
- Otro encargado de suministrar el mapa del campus con el servicio WMS.

La base de datos es la misma para ambos servidores. Es la que suministra los datos a ambos servidores.

- El cliente web utiliza librerías de Leaflet y JQuery Ajax para la comunicación con la capa de aplicación.

- Se utiliza el framework Springboot en las capas superiores de Aplicación y dominio
- En la capa inferior de infraestructura se utiliza Spring-JPA e Hibernate ORM
- El servicio WMS lo proporciona el servidor Apache con Geoserver instalado.

Diagrama de módulos:



Módulos

Backend:

Dominio:

- Entidad.java: Clase que representa una entidad.
- Administrador.java: Clase que representa un Administrador.
- Espacio.java: Clase que representa un espacio.
- Estado.java: Clase que representa los posibles estados de una incidencia.
- Horario.java: Clase que representa la entidad horario.
- Localizacion.java: Clase que representa una localización.
- Incidencia.java: Clase que representa la entidad incidencia.
- RepositorioEspacios.java: Interfaz del repositorio de espacios.
- RepositorioIncidencias.java: Interfaz del repositorio de incidencias.
- RepositorioHorario.java: Interfaz del repositorio de horarios.
- RepositorioAdministrador.java: Interfaz del repositorio de Administrador.

Infraestructura:

- EspacioPostGis.java: Clase que representa la entidad de EspacioPostGis
- RepoIncidenciasImpl.java: Clase que implementa el repositorio de Incidencias
- RepoIncidenciasSpring.java: Interfaz del repositorio de incidencias que usa Spring
- RepoEspacioDominio.java: Clase que implementa el repositorio de Espacios
- RepoEspacioSpring.java: Interfaz del repositorio de espacio que usa Spring
- RepoHorarioImpl.java: Clase que implementa el repositorio de Horarios
- RepoHorarioSpring.java: Interfaz del repositorio horario que usa Spring
- RepoAdministradorImpl.java: Clase que implementa el repositorio de Horarios
- RepoAdministradorSpring.java: Interfaz del repositorio administrador que usa Spring

Aplicación:

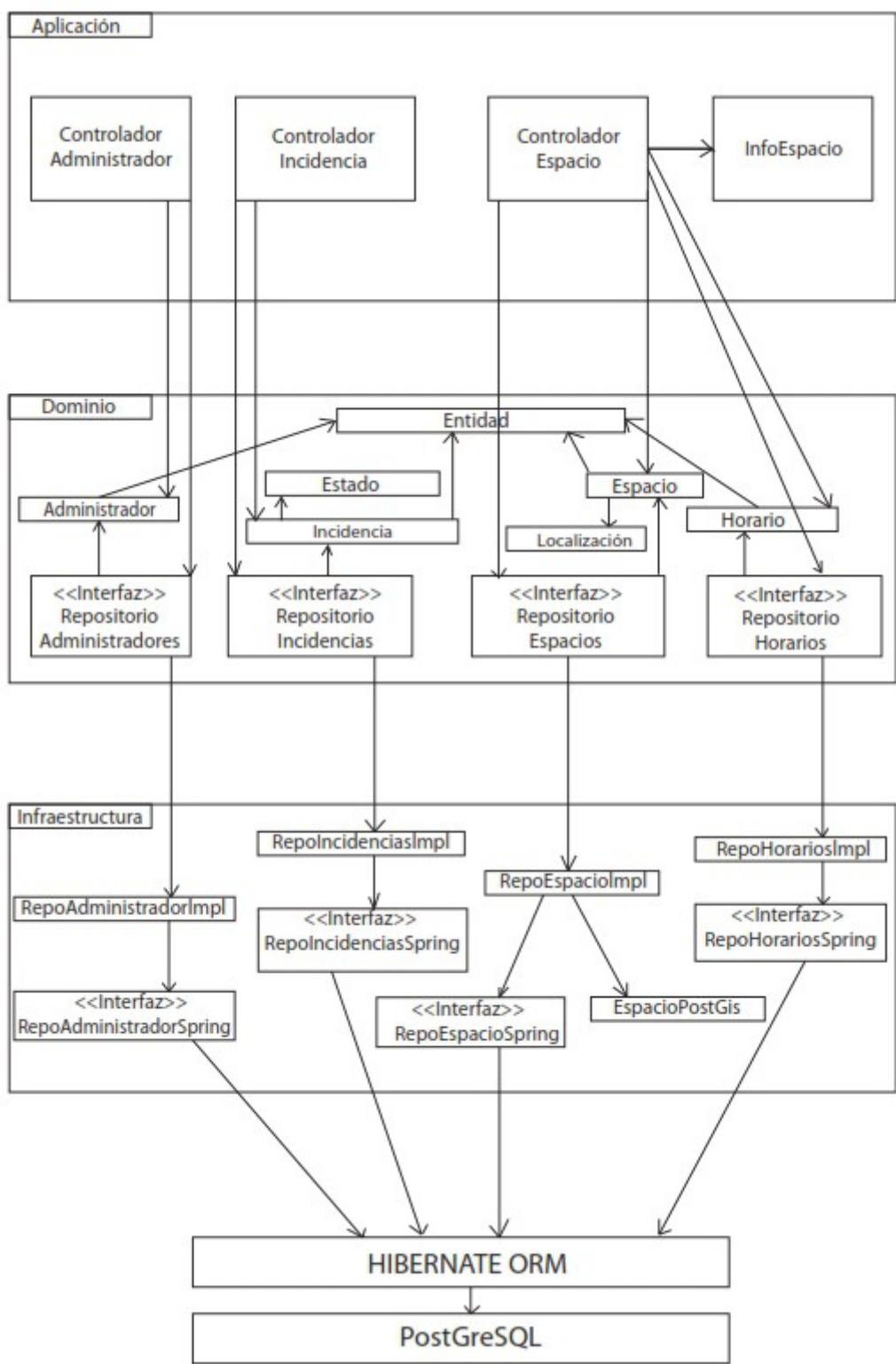
- ControladorEspacio.java: Clase que representa un controlador para las acciones relacionadas con los espacios
- ControladorAdminsitrador.java: Clase que representa un controlador de las acciones del administrador
- ControladorIncidencia.java: Clase que representa un controlador para acciones relacionadas con las incidencias
- InfoEspacio.java: Clase que representa la información de un espacio

Test:

- ControladorAdministradorTest.java: Implementa los casos de uso que se refieren al administrador
- ControladorEspacioTest.java: Implementa los casos de uso que se refieren al espacio
- ControladorIncidenciaTest.java: Implementa los casos de uso que se refieren a las incidencias

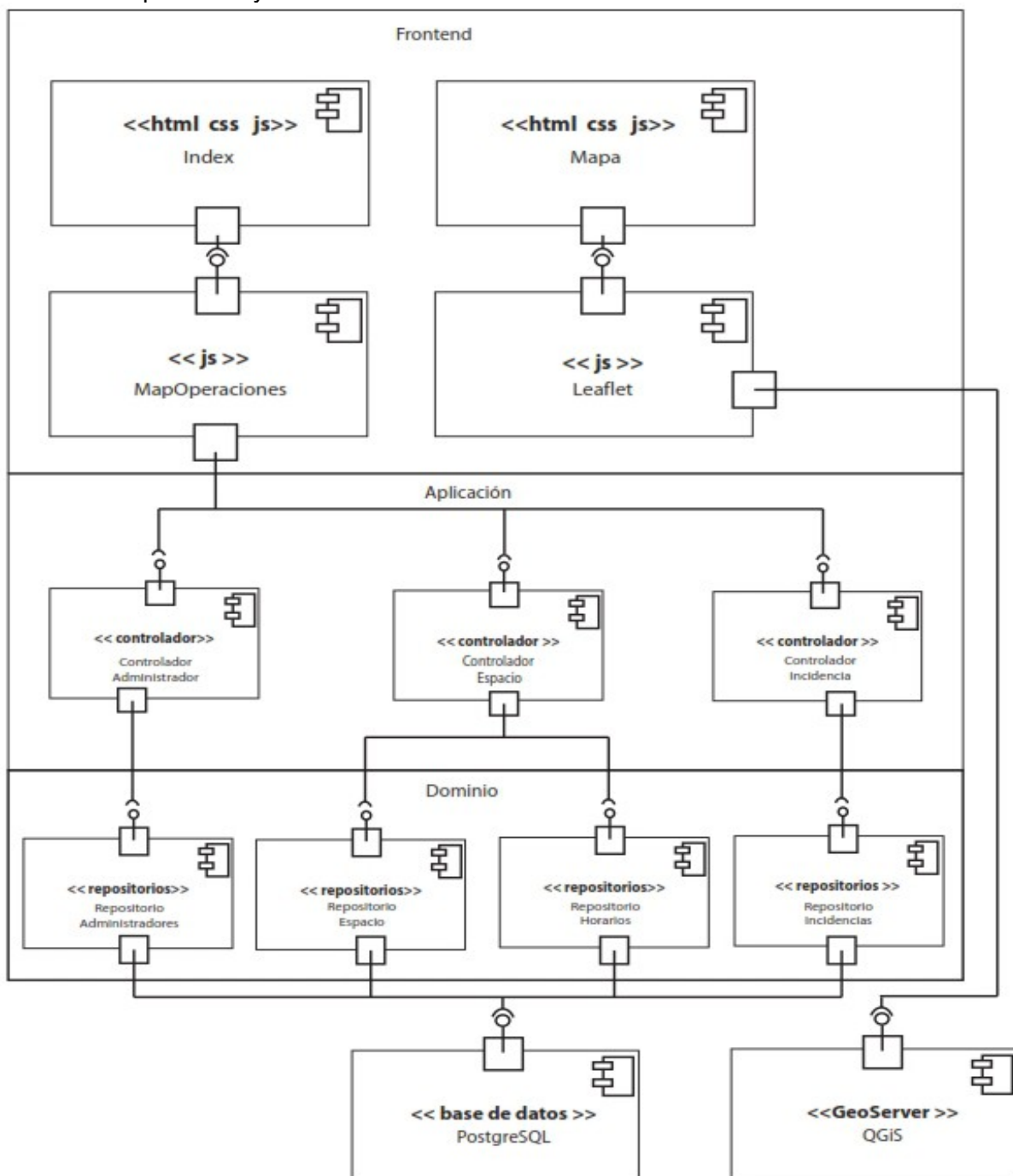
- RepositorioAdministradoresTest.java: Implementa las pruebas del repositorio Administradores
- RepositorioEspaciosTest.java: Implementa las pruebas del repositorio Espacio

Diagrama de módulos dominio:



En el diagrama se observa como están compuestas las diferentes capas y las dependencias entre ellas.

Vista de componentes y conectores



3. Conclusiones

3.1. Resumen

Se ha realizado el trabajo “Mantenimiento del campus” aplicando diseño guiado por el dominio. Y se ha conseguido los requisitos esperados desarrollando la destreza y comprensión del tipo de diseño aplicado.

3.2. Evaluación del proyecto

Suficiente

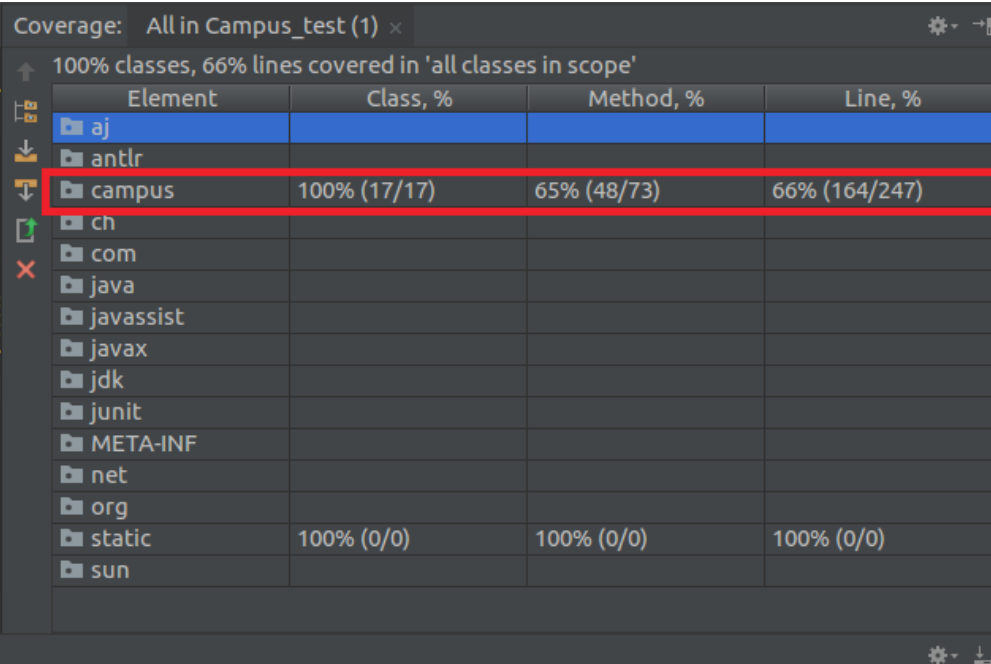
- **El código y la documentación del proyecto se alojan en GitHub. Se trabaja de forma habitual contra Git:**
 - <https://github.com/Gonzakick/Campus.git>
- **Compilación y gestión de dependencias basadas en scripts (Gradle, Maven, npm...)**
 - Se utiliza Gradle.
- **Se llevará un control de esfuerzos con las horas dedicadas por persona. Se trabaja un número de horas en el entorno de lo requerido para la asignatura (unas 90 horas por persona). Se entregará un resumen al mes.**
 - Se han entregado todos los resúmenes en las fechas indicadas
- **La aplicación cumple sus requisitos**
 - Se ha desarrollado la versión “Mantenimiento del campus” propuesta para la segunda convocatoria.
- **La documentación arquitectural es la adecuada al momento del proyecto, refleja fielmente el sistema e incluye al menos una vista de módulos, otra de componentes-y-conectores, y otra de despliegue**
 - La documentación arquitectural es la adecuada
- **La arquitectura del sistema es por capas**

La arquitectura del sistema es por capas (Interfaz, Aplicación, Dominio, Infraestructura)

- **Se usan adecuadamente estos conceptos de diseño dirigido por el dominio: entidades, objetos valor, agregados, factorías y repositorios**
 - En los diagramas se puede apreciar.
- **La aplicación permite hacer modificaciones de datos concurrentes**
 - Se usa la etiqueta *@Transactional* en todas aquellas operaciones que pueden tener un comportamiento concurrente
- **Se ha puesto en marcha y se usa un servicio de mapas tipo WMS con los edificios disponibles del campus Río Ebro. Los mapas de este servicio se superponen en el cliente sobre otro servicio externo (p.ej. Open Street Map) que proporcione un mapa de la zona**
 - A través de Leaflet se superpone un mapa tipo WMS al mapa Open street Map el cual se encuentra en un servidor Apache Tomcat en el cual esta instalado Geoserver.

Notable

- **Cobertura de tests automáticos de al menos el 30% del código (unitarios y/o de integración)**



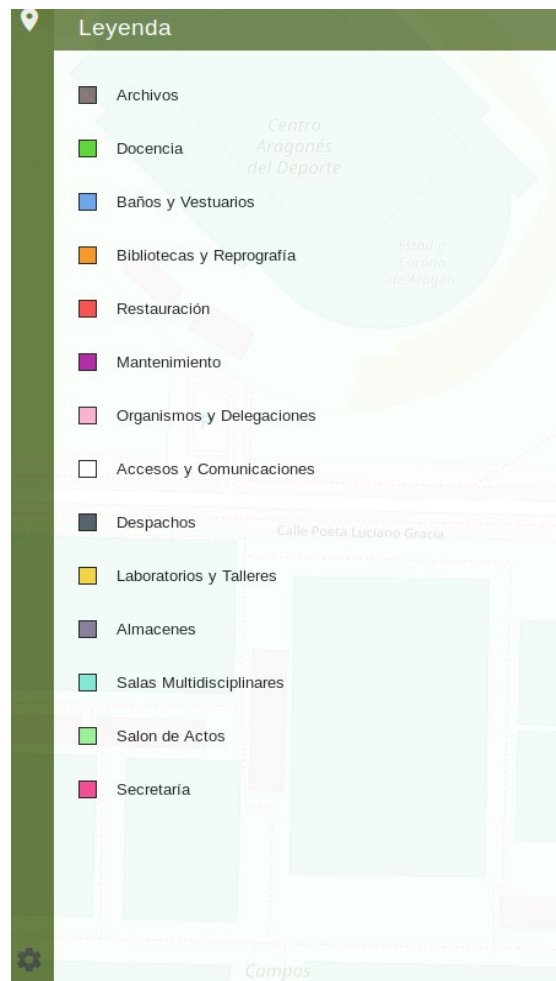
Coverage: All in Campus_test (1) ×			
↑ 100% classes, 66% lines covered in 'all classes in scope'			
Element	Class, %	Method, %	Line, %
aj			
antlr			
campus	100% (17/17)	65% (48/73)	66% (164/247)
ch			
com			
java			
javassist			
javax			
jdk			
junit			
META-INF			
net			
org			
static	100% (0/0)	100% (0/0)	100% (0/0)
sun			

- **El modelo de dominio utiliza adecuadamente estos conceptos del diseño dirigido por el dominio: servicios, paquetes, interfaces reveladoras, aserciones, funciones libres de efectos secundarios**

- Si.

- **El estilo cartográfico de los edificios en el servicio de tipo WMS refleja el tipo de uso de cada espacio (por ejemplo, los laboratorios de un color, los despachos de otro etc.)**

- Se usa un estilo cartográfico que refleja cada espacio en un color según su uso:



Sobresaliente

- Cobertura de tests automáticos de al menos el 50% del código (unitarios y/o de integración)

Coverage: All in Campus_test (1) ×

100% classes, 66% lines covered in 'all classes in scope'

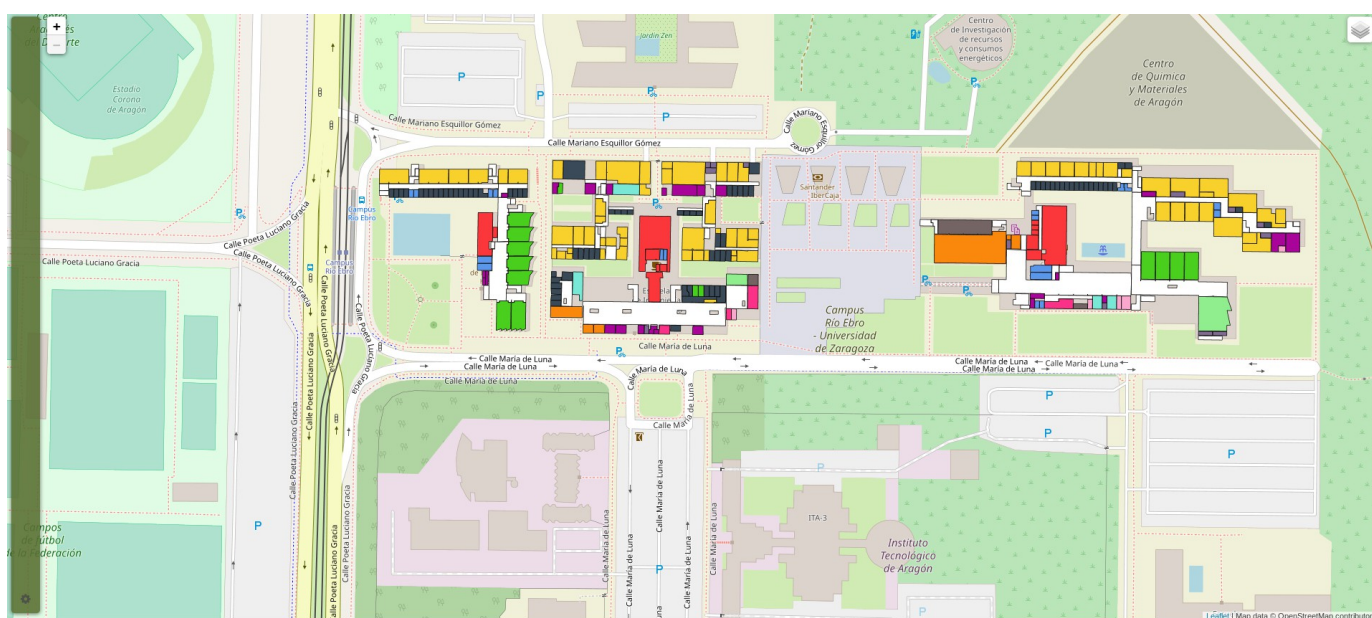
Element	Class, %	Method, %	Line, %
aj			
antlr			
campus	100% (17/17)	65% (48/73)	66% (164/247)
ch			
com			
java			
javassist			
javax			
jdk			
junit			
META-INF			
net			
org			
static	100% (0/0)	100% (0/0)	100% (0/0)
sun			

○

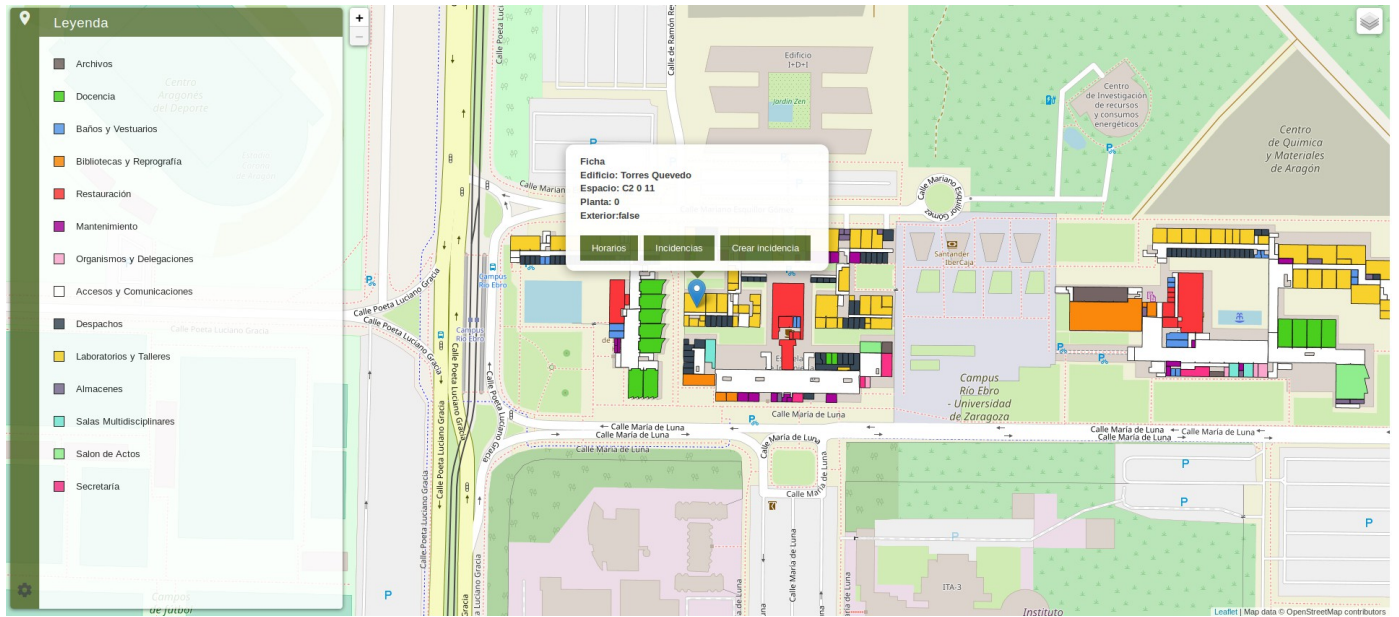
- La arquitectura del sistema es hexagonal
 - La arquitectura del sistema no es hexagonal

Anexo. Manual de usuario

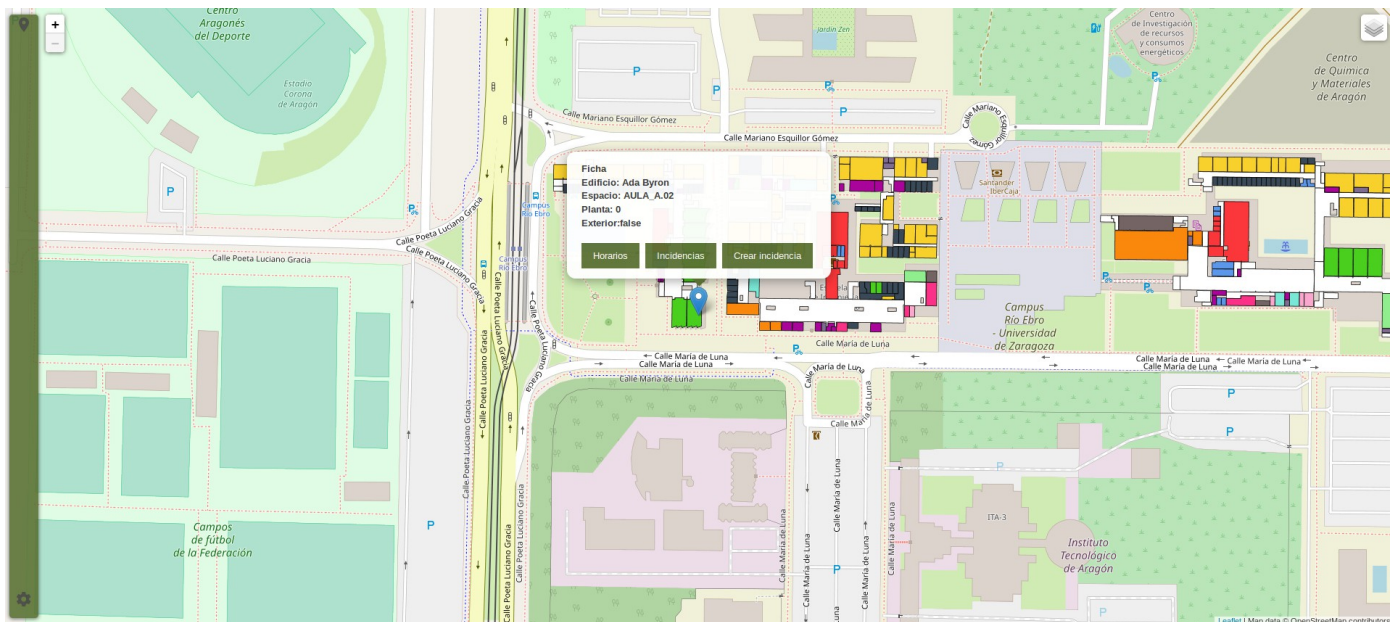
Mantenimiento del campus



Visión general del campus, con los colores señalizados en la leyenda para consultarla en cualquier momento.



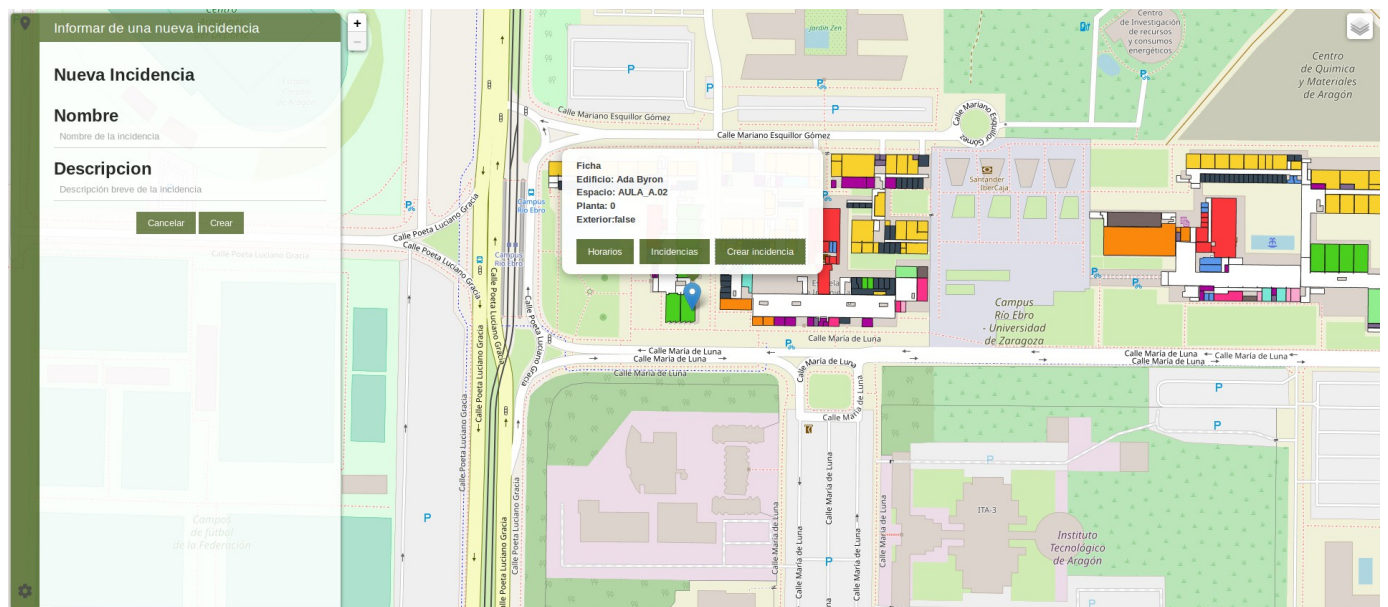
Información de un espacio



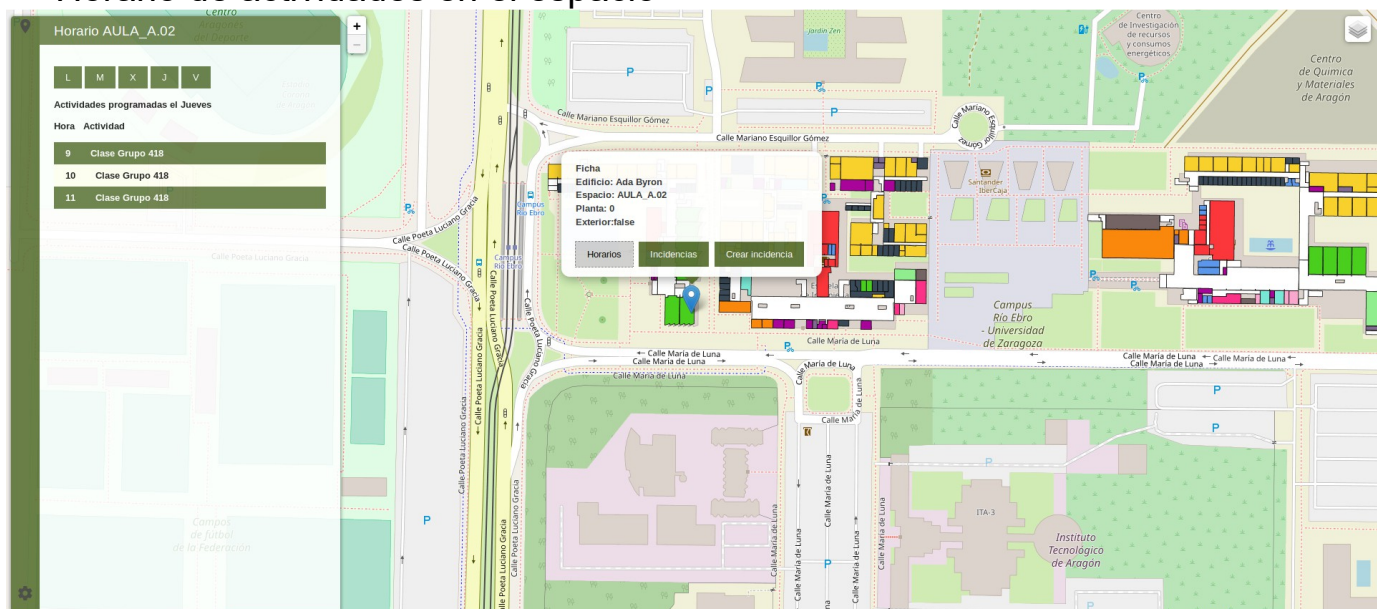
El usuario al seleccionar una localización se le despliega el siguiente menú para que seleccione la acción que quiere realizar: Horario , incidencias o crear una incidencia.

Creación de una incidencia

El usuario para registrar una nueva incidencia solo tiene que rellenar los campos y aceptar.

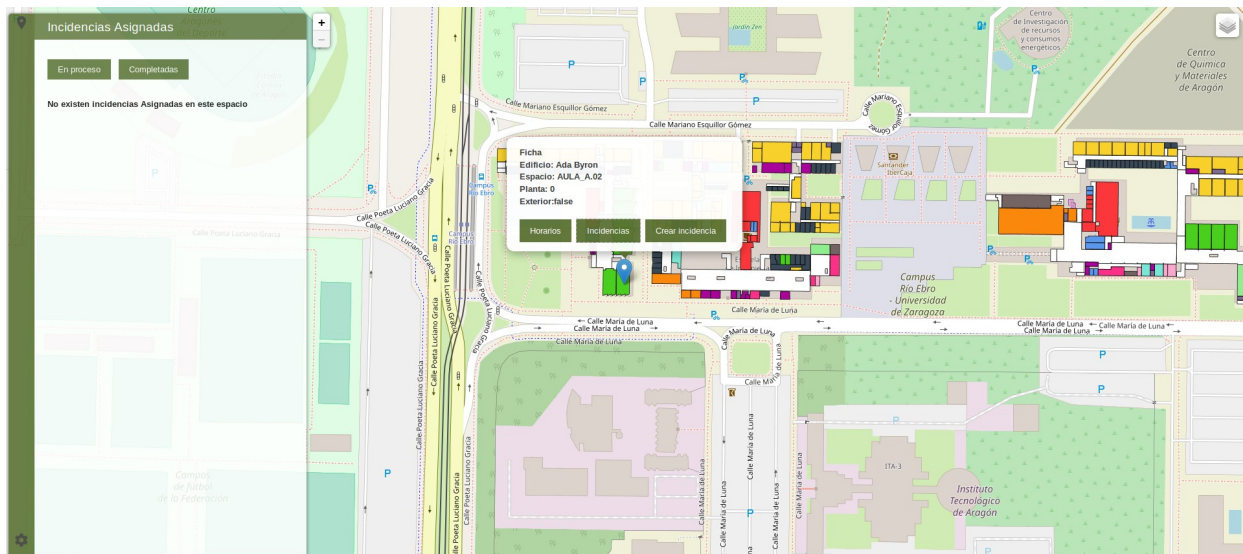


Horario de actividades en el espacio



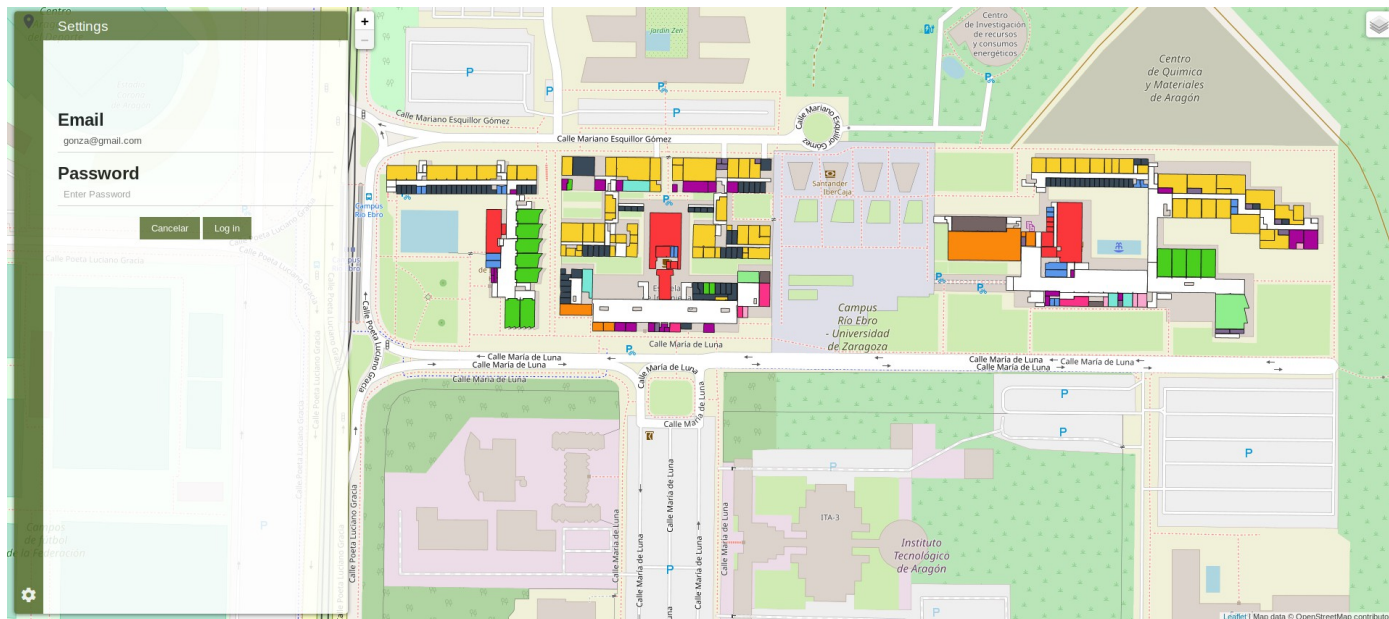
En cambio, si selecciona la pestaña horario se le desplegara la barra mostrando el horario actual de la localización , dando la posibilidad de navegar entre el resto de días de la semana.

Registro de incidencias



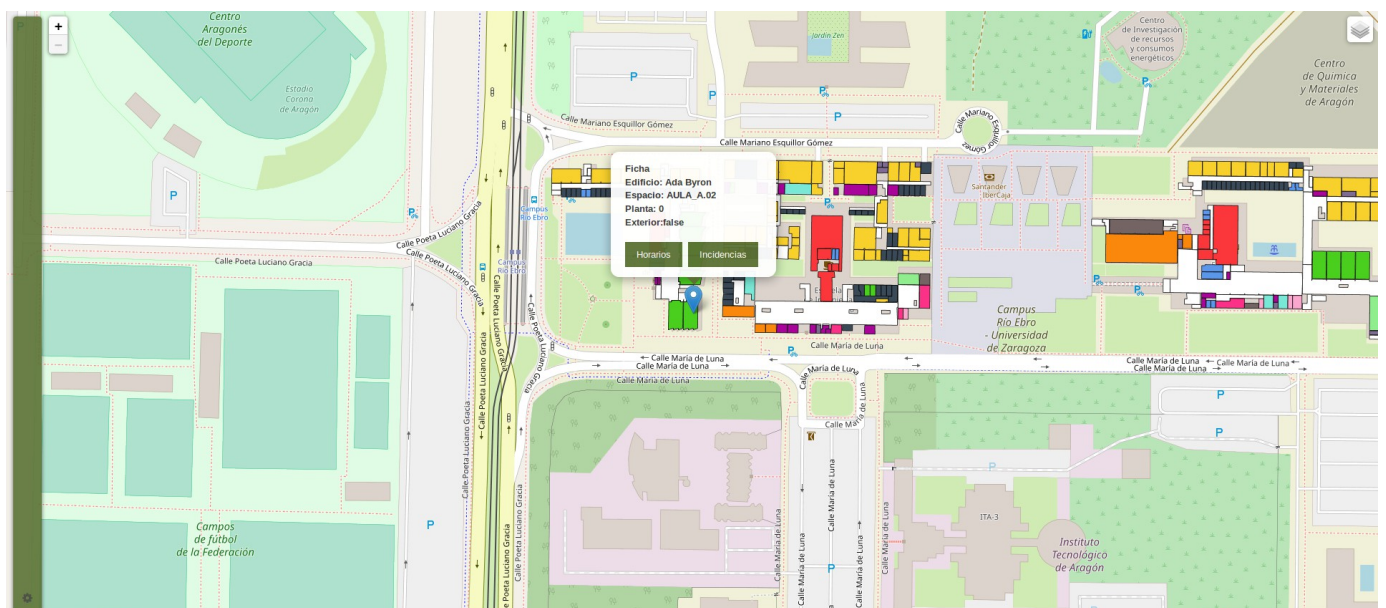
El usuario sólo tiene posibilidad de acceder a las incidencias asignadas o completadas y visualizarlas con sus datos.

Log in del administrador



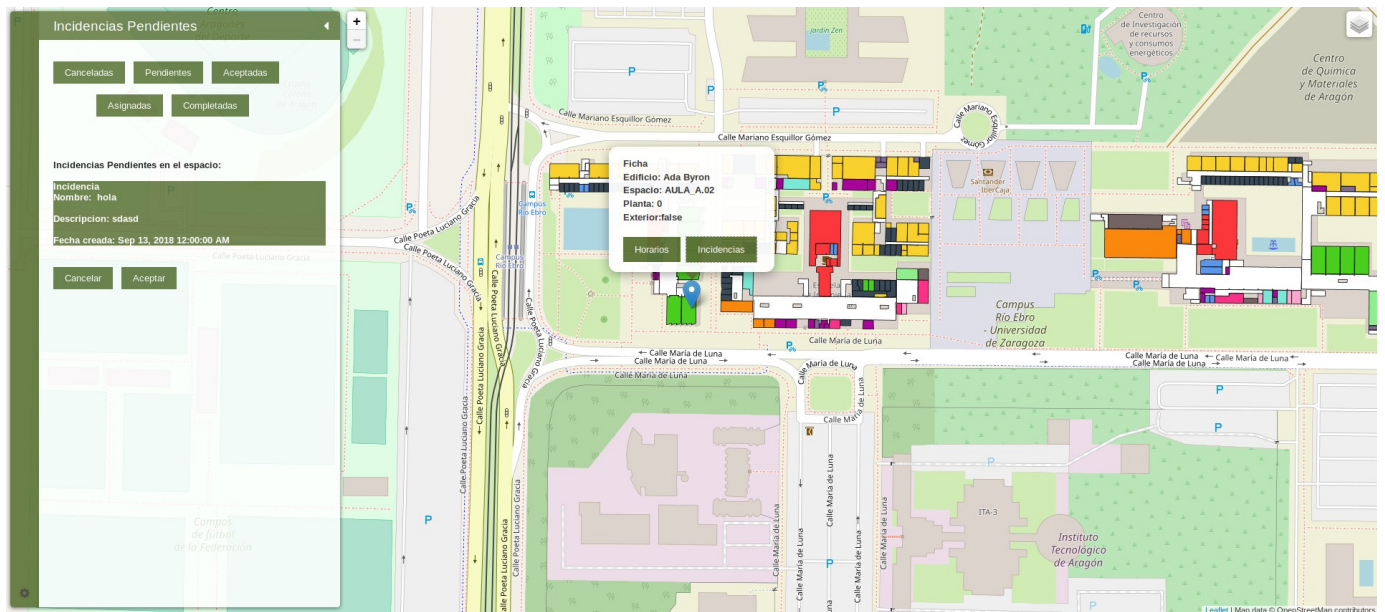
Si el usuario tiene permisos de administrador, solo tiene que introducir sus credenciales para acceder a la gerencia.

Menú del administrador



El menú del administrador es más simple, solo muestra horarios de las localizaciones y las incidencias que hay en ellas.

Gestión de las incidencias del administrador



A diferencia del usuario normal, el administrador tiene permiso para gestionar todo tipo de incidencias, realizando las acciones pertinentes.