Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema de gestión de geofences

Geofences management system

Autor

Eduardo Ibáñez Vásquez

Director

Francisco Javier Zarazaga Soria

Escuela de Ingenieros y Arquitectos

2016

Sistema de gestión de geofences

RESUMEN

El proyecto consiste en implementar un sistema de gestión de geofences. Un geofence es un perímetro virtual de un área geográfica del mundo real. El sistema debe ser capaz de crear, modificar, obtener y borrar geofences. Del mismo modo, debe de poder realizar las mismas acciones para usuarios, reglas de comportamiento que tienen asociados los geofences y notificaciones generadas al cumplir las reglas.

Además, el sistema debe poder gestionar la posición enviada por el usuario, de forma que se encargue de la generación de las notificaciones en caso de que la localización recibida se encuentre dentro de un geofence, habiendo satisfecho alguna regla del mismo.

En el proyecto se han identificado cuatro recursos principales, usuarios, geofences, reglas y notificaciones, para los que se han desarrollado servicios web que siguen el estilo de arquitectura REST con el módulo Spring Web. Mientras que para el servicio relativo a la gestión de la posición, se emplea la tecnología WebSockets, que es una tecnología estándar que proporciona un canal de comunicación bidireccional y *full-duplex* sobre un único socket TCP.

Para el almacenamiento de los datos se emplea la base de datos PostgreSQL/PostGIS, que es una base de datos relacional con un módulo que ofrece soporte para objetos espaciales, necesario en el sistema para poder guardar la geometría de los geofences. Para la creación de las tablas se ha empleado Hibernate Spatial, que es una herramienta que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, siendo también capaz de reconocer estos objetos espaciales que deben seguir el estándar SFA, es decir, Simple Feature Access.

También se han diseñado un par de aplicaciones web desplegadas en distintos servidores de presentación para comprobar el correcto funcionamiento de los servicios web del sistema, es decir, que utilizan la API del sistema.

Toda la comunicación del sistema se encuentra bajo supervisión de un control de accesos, que se encarga de permitir o denegar los servicios web del sistema, dependiendo de los requerimientos de seguridad establecidos para cada funcionalidad. Además, se ha mejorado la configuración de la seguridad que implementa por defecto Spring Security, añadiendo la autenticación por medio de JSON Web Tokens, que sigue el estándar industrial RFC 7519 para representar peticiones seguras entre el cliente y el servidor.

Por último, se ha ejecutado una batería de pruebas unitarias, de sistema y de integración con el fin de depurar errores y mejorar el código del proyecto.

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Eduardo Ibáñez Vásquez,

con nº de DNI 73131561W en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Sistema de gestión de geofences,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 14 de mayo de 2016

Fdo: Eduardo Ibáñez Vásquez

Dedicar este trabajo de fin de grado

a mi hermano por su constante e inestimable apoyo,

a mis padres por su paciencia y solidaridad

durante estos intensos años de estudio en la Universidad y

a Francisco Javier López Pellicer y Francisco Javier Zarazaga Soria

por su entrega y dedicación al proyecto.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

1. INTRODUCCIÓN

Hoy en día estamos constantemente expuestos a la influencia de la publicidad en nuestra forma de vida. Esta publicidad nos llega a través de anuncios en la televisión, en las revistas, las paradas de autobús... Sin embargo, es genérica, no está orientada a un individuo en particular, sino a grandes colectivos de personas.

Este hecho no quiere decir que las empresas no busquen personalizar o adecuar su publicidad a cada cliente, sino que hasta hace pocos años no ha existido un medio o forma de hacer esto posible con gran precisión.

La tecnología del *geofencing* [1] permite realizar esta tarea con bastante éxito, dado que ofrece la posibilidad de enviar notificaciones a los usuarios en función de su ubicación exacta, proveniente del GPS o del rastreo de la IP. Esta tecnología tiene un gran potencial porque no es únicamente aplicable al marketing, sino que se puede utilizar en cualquier caso imaginable en el que se quiera enviar una notificación o anuncio en función de la ubicación de una persona. Además, el envío de la notificación puede condicionarse a una serie de reglas previamente diseñadas, lo cual posibilita un mayor control sobre cuándo y cómo una notificación será recibida por los usuarios.

1.1. Objetivo y alcance del proyecto

El objetivo propuesto para este trabajo consiste en crear un sistema de gestión de geofences, donde un geofence es un perímetro virtual sobre una zona geográfica real. Para ello, el sistema tiene que ser capaz de administrar usuarios, geofences, reglas que tienen asociados esos geofences, y notificaciones que se envían al cumplir alguna regla. Además, también tiene que saber gestionar las localizaciones enviadas por cada cliente.

Entre las funcionalidades que presenta el sistema se encuentra:

- La creación, lectura, actualización y borrado de usuarios.
- La creación, lectura, actualización y borrado de geofences.
- La creación, lectura, actualización y borrado de reglas.
- La creación, lectura, actualización y borrado de notificaciones.
- El guardado de la localización enviada de los clientes.
- La autenticación de los clientes para permitir utilizar la mayoría de las funcionalidades que ofrece el sistema.
- El procesamiento de las localizaciones para comprobar su ubicación y enviar una notificación determinada, en caso de que proceda.

1.2. Contexto

Actualmente existen varios servicios relacionados con el *geofencing*, aunque son poco conocidos por el momento. Sin embargo, las empresas empiezan a interesarse cada vez más en esta tecnología dado que puede resultar muy útil para sus planes estratégicos o de marketing.

1.2.1. Geofence

Un geofence es un perímetro virtual de un área geográfica del mundo real. Puede generarse de manera dinámica como un radio alrededor de la ubicación de un punto, como en la Figura 1, o puede ser un conjunto de coordenadas organizadas que forman un área con límites definidos.

Figura 1. Visualización de un geofence en un mapa

Un geofence puede tener asociados un conjunto de reglas que definen su comportamiento respecto a las acciones que pueda efectuar un usuario, generando eventos o notificaciones. Existen principalmente tres tipos de reglas: de entrada, de salida o de permanencia.

En conclusión, se le puede especificar a un geofence cuando va a originar los eventos a partir de las reglas que se le asignen; reglas de entrada si se quiere que cree el evento pasados *n* segundos desde la entrada del usuario en el geofence, reglas de salida si se quiere que se cree el evento pasados *n* segundos desde la salida del usuario del geofence o reglas de permanencia si se quiere que se cree el evento pasados *n* segundos desde que el usuario ha entrado en el geofence, manteniéndose dentro del mismo todo el tiempo.

1.2.2. Servicios comerciales de *geofencing*

Aplicaciones como las que ofrecen Plot Projects[1] o Google Maps[2] para la gestión de geofences resultan bastante limitadas en algunos aspectos. Por ejemplo, la aplicación de Plot Project presenta restricciones en las formas que pueden adoptar los geofences, dado que admite únicamente geofences circulares, como en la Figura 1, privando al cliente de la capacidad de amoldar un geofence según sus necesidades. Por otra parte, la aplicación de Google Maps tiene un conjunto de reglas insuficiente si se quiere administrar el comportamiento de un geofence optimizando al máximo el empleo de esta tecnología, aunque están trabajando en una nueva versión que seguro trae nuevas características.

Este proyecto, que es ligero, pequeño y abierto, pretende eliminar o reducir al máximo posible las restricciones anteriormente citadas que se han encontrado en aplicaciones del campo de *geofencing*.

1.3. Métodos y técnicas

El proyecto se ha organizado mediante el empleo de los conceptos de las metodologías ágiles de desarrollo del software [2], lo cual permite un desarrollo incremental e iterativo.

Cada iteración tiene una duración de dos semanas y a la finalización de cada una se lleva a cabo una reunión entre el autor y el director del proyecto, donde se discuten los problemas encontrados durante la iteración, así como el conjunto de tareas de la pila que se plantean realizar durante la siguiente iteración. Esta reunión sirve también para intercambiar opiniones sobre el proyecto y conocer el estado del mismo.

Las tareas que se van a realizar durante cada iteración se encuentran en una pila priorizada del producto, que se ha obtenido identificando qué se quiere construir y en qué orden al principio del proyecto, como se puede apreciar en la Figura 2. Las tareas no tienen que ser demasiado grandes, para que puedan ser abarcables durante una única iteración, dado que no es conveniente que una tarea de alargue más de una. En caso de necesitar hacer más cambios en una característica del proyecto comprendida en una tarea finalizada, se puede crear una nueva tarea que complemente esa característica añadiéndola a la pila. Estas tareas no se pueden dejar sin completar y tampoco es aconsejable añadir nuevas tareas durante la realización de las iteraciones, sino al principio de cada una.

[1] https://www.plotprojects.com/geofencing/
[2] https://developer.android.com/training/location/geofencing.html

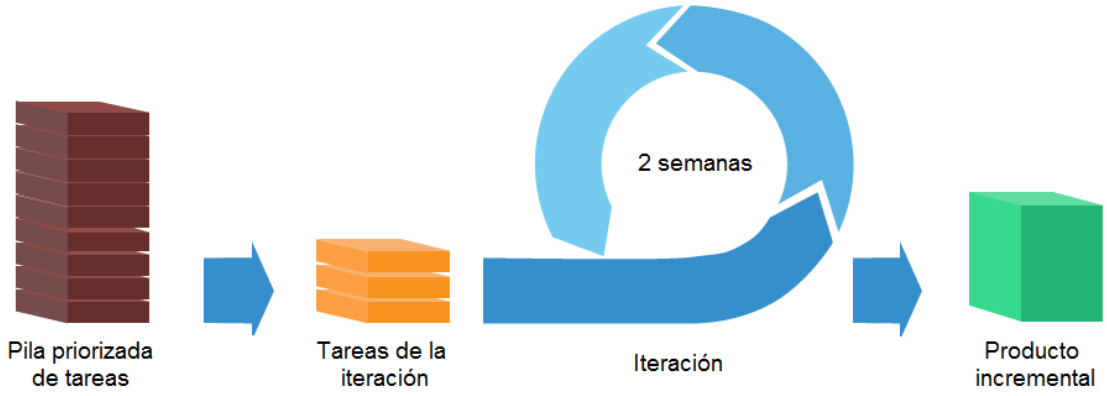

Figura 2. Diagrama de metodología ágil basada en pila de tareas e iteraciones

Cada una de las tareas se encuentra dentro de los siguientes rangos:

- **Análisis:** Análisis del estado del arte y búsqueda de requisitos.
- **Diseño:** Diseño de la arquitectura y módulos principales del proyecto.
- **Implementación:** Implementación de funcionalidades o características que presenta el proyecto, incluyendo sus pruebas unitarias y de sistema correspondientes.
- **Demostrador y evaluación del sistema:** Implementación de las aplicaciones de comprobación y utilización de las funcionalidades del sistema.

1.4. Tecnologías empleadas

Las principales tecnologías de las que se ha hecho uso para la elaboración del proyecto son las siguientes.

Con respecto a las tecnologías empleadas en el sistema:

- **Java 8**[3]**.** Es un lenguaje de programación de propósito general, concurrente y orientado a objetos. Se ha utilizado para implementar toda la lógica de la aplicación del sistema.
- **Spring framework**[4]**.** Es un *framework* para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Se hace uso de esta tecnología para la elaboración de la API del sistema con Spring Web, el acceso a la base de datos con Spring Data JPA y la autenticación para el empleo de determinados métodos con Spring Security.
- **OpenAPI Specification**[5]**.** Anteriormente conocida como Swagger Specification, es una especificación que permite describir, producir, consumir y visualizar servicios web de tipo REST, añadiendo una serie de etiquetas en las cabeceras de los métodos.

[3] https://www.java.com/es/download/faq/java8.xml
[4] https://projects.spring.io/spring-framework/
[5] https://openapis.org/specification

- **Hibernate Spatial[6].** Es una herramienta de mapeo objeto-relacional para la plataforma de Java, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación.
- **PostgreSQL/PostGIS[7].** Es un sistema de gestión de base de datos relacional orientado a objetos y libre, que se utiliza en el proyecto con el módulo PostGIS, que es un módulo que añade soporte de objetos espaciales a la base de datos.
- **JSON Web Tokens[8].** Es una metodología abierta que sigue el estándar industrial RFC 7519 para representar peticiones seguras entre dos partes. También se puede definir como un sistema de codificación que cifra un objeto JSON utilizando *JSON Web Signature* [3] y que es transferido entre dos partes para establecer una comunicación segura.
- **JUnit4[9].** Es un conjunto de librerías que son utilizadas en programación para hacer pruebas unitarias, de sistema y de integración en aplicaciones Java.

Con respecto a las tecnologías empleadas en las aplicaciones web:

- **JavaScript.** Es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
- **HTML5 y CSS3.** Son dos tecnologías utilizadas frecuentemente de manera conjunta, que sirven para establecer la estructura y contenido de una página web, así como para definir su estilo visual.
- **jQuery 1.9.1[10].** Es una librería de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, así como manejar eventos, entre otras cosas.
- **Bootstrap 3[11].** Es un *framework* que facilita el diseño de páginas y aplicaciones web.
- **WebSockets.** Es una tecnología estándar que proporciona un canal de comunicación bidireccional y *full-duplex* sobre un único socket TCP. Para iniciar la conexión primero el cliente tiene que pedir al servidor que desea iniciar la conexión, empleando peticiones HTTP. Una vez finalizada la etapa de *handshake*, es decir, de ponerse de acuerdo el cliente y el servidor, pasan a utilizar WebSockets.
- **SockJS.** Es una librería de JavaScript que proporciona un *socket* sobre el que establecer una conexión vía WebSockets utilizando el protocolo STOMP.

[6] http://www.hibernatespatial.org/
[7] http://postgis.net/
[8] https://jwt.io/introduction/
[9] http://junit.org/junit4/
[10] https://jquery.com/
[11] http://getbootstrap.com/

1.5. Herramientas utilizadas

Para el desarrollo del proyecto se han hecho uso de las siguientes herramientas de trabajo:

- **GitHub.** Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git.
- **IntelliJ IDEA.** Es un entorno de desarrollo integrado para el desarrollo de programas informáticos, realizado por JetBrains. Facilita la interacción con GitHub.
- **Gradle.** Es una herramienta que permite automatizar procesos de compilación, ejecución y despliegue de sistema en un servidor embebido en el propio script de Gradle. Incluye soporte de dependencias, permitiendo descargar aquellas librerías que sean necesarias para su correcta ejecución.

2. TRABAJO REALIZADO

A continuación se va a explicar el trabajo realizado en cada de una de las fases que componen en proyecto, como son análisis, diseño, implementación y demostrador y evaluación del sistema.

2.1. Análisis

La fase de análisis permite determinar el alcance del proyecto, las funcionalidades que va a presentar el sistema una vez esté acabado. Para alcanzar este objetivo, se redactan las historias de usuario, se realizan diagramas de casos de uso y se definen los requisitos funcionales y no funcionales.

2.1.1. Historias de usuario

Con respecto a las historias de usuario, en ellas podemos encontrar el comportamiento y las funcionalidades que se espera presente el proyecto.

El usuario quiere poder crear una cuenta de usuario, con la que sea capaz de identificarse en el sistema, así como poder modificar, eliminar u visualizar información de esa cuenta. Por otra parte, desea tener la posibilidad de crear, modificar, eliminar y visualizar geofences en el sistema.

En referencia a las reglas, el usuario busca tener la posibilidad de controlar la entrada, salida y permanencia de un usuario dentro de un geofence por un tiempo determinado. Por último, el usuario desea poder crear, modificar, eliminar y visualizar notificaciones, que estarán ligadas al cumplimiento de las reglas que se definan con anterioridad en el sistema.

Las historias de usuario completas se encuentran en el ANEXO A, donde se especifican cada una de las historias elaboradas por el usuario.

2.1.2. Casos de uso

Con el fin de representar las funcionalidades que va a ofrecer el sistema de gestión de geofences se han realizado los diagramas de casos de uso relativos a un usuario autenticado, en la Figura 3, y un usuario no autenticado, en la Figura 4.

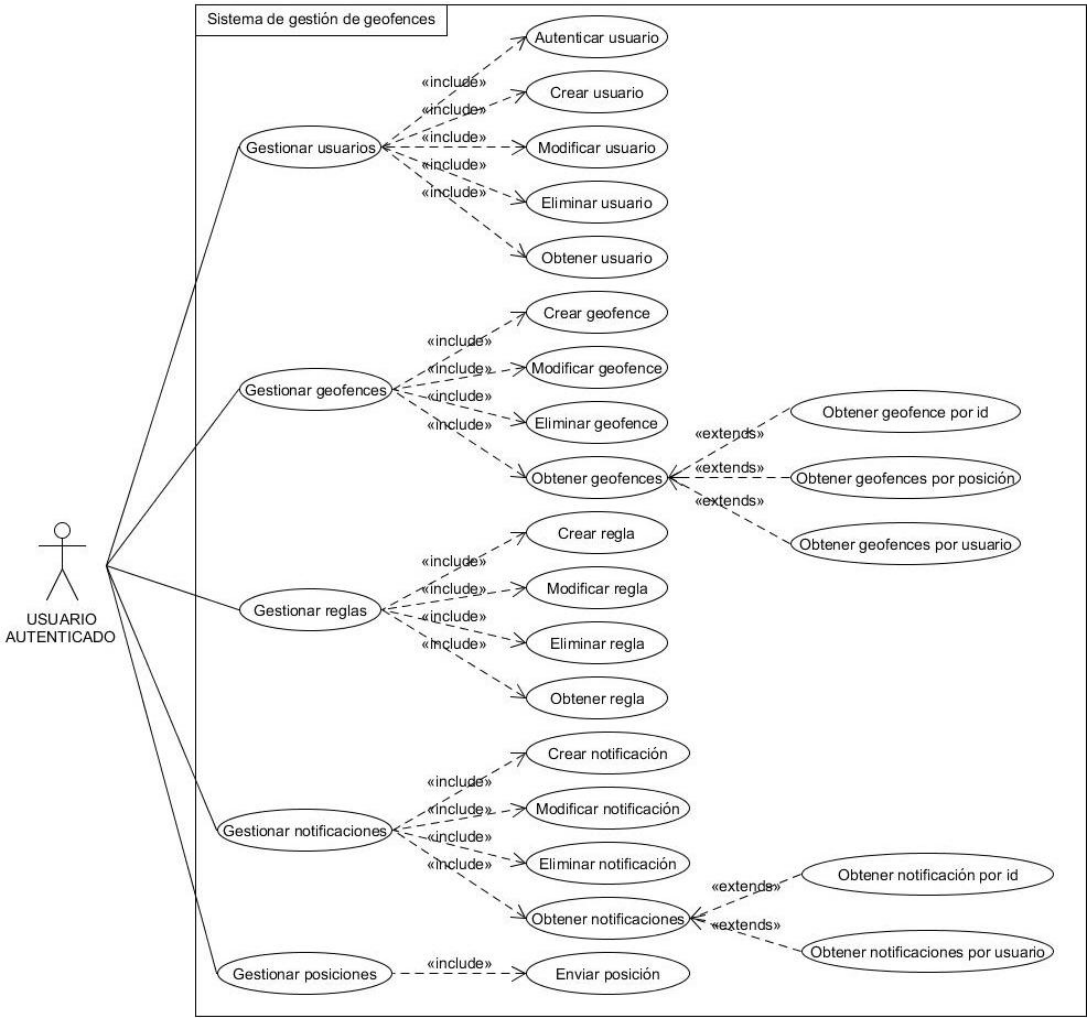

Figura 3. Diagrama de casos de uso para usuario autenticado

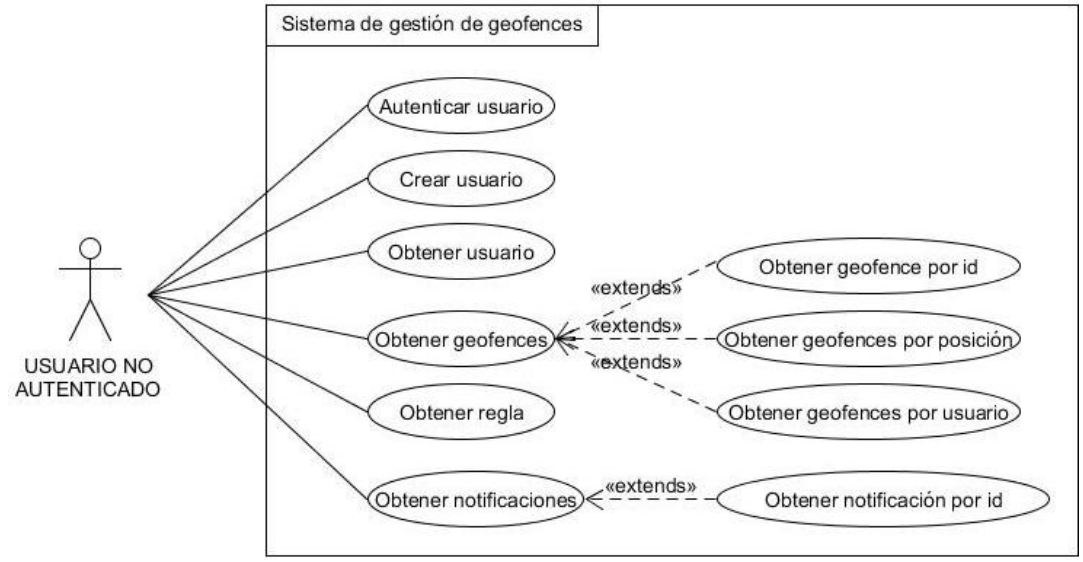

Figura 4. Diagrama de casos de uso para usuario no autenticado

Los casos de usos detallados se encuentran en el ANEXO B, donde se describen los principales casos de uso expuestos en los diagramas. Mientras que las relaciones entre los requisitos y los casos de uso se hallan en el ANEXO C.

2.1.3. Requisitos funcionales

A continuación se exponen los requisitos funcionales de los que consta el proyecto en la Tabla 1.

Código	Requisito funcional
RF1	El usuario podrá crear, visualizar, modificar o eliminar una cuenta de usuario. Una cuenta de usuario está formada por el nombre, el apellido, la fecha de nacimiento, el nick, la contraseña, el IMEI del dispositivo, la fecha de la última vez que se ha modificado la contraseña, el rol de usuario, un booleano de habilitación del rol y el identificador público.
RF2	El usuario podrá iniciar sesión en su cuenta de usuario introduciendo su nick y contraseña en la aplicación.
RF3	El usuario podrá crear, visualizar, modificar o eliminar geofences. Un geofence está formado por el identificador, el tipo, las propiedades, la geometría del geofence y el usuario que lo ha creado.
RF4	El usuario podrá crear, visualizar, modificar o eliminar reglas. Una regla está formada por el identificador, el booleano de habilitación de la regla, el tipo, el tiempo asignado a la regla, el mensaje, el conjunto de días en los que aplicar la regla y el geofence al que está asociado a la regla.
RF5	El usuario podrá crear, visualizar, modificar o eliminar notificaciones. Una notificación está formada por el identificador, la regla sobre la que está asociada la notificación, el usuario que ha recibido la notificación, el estado de la misma y la fecha de envío de la notificación.
RF6	El usuario podrá enviar su posición para que el sistema compruebe su ubicación con respecto a las geofences.
RF7	El usuario podrá definir mediante una regla si un geofence tiene que enviar una notificación cuando el usuario entre en el área de cobertura, cuando el usuario salga del área de cobertura o en ambos casos.
RF8	El usuario podrá asignarle a una regla un tiempo de espera antes de que envíe alguna notificación a los usuarios, siempre que éstos se hayan mantenido en todo momento dentro de su área de cobertura.
RF9	El usuario podrá asignarle a una regla un tiempo de espera antes de que envíe alguna notificación a los usuarios, a partir del momento de entrada o de salida del geofence.

Tabla 1. Requisitos funcionales

2.1.4. Requisitos no funcionales

De igual manera, se muestran los requisitos no funcionales que posee el proyecto en la Tabla 2.

Código	Requisito funcional
RNF1	El servidor debe estar codificado en el lenguaje Java.
RNF2	El servidor debe ofrecer sus servicios web a través de una API de tipo RESTful.
RNF3	El servidor debe utilizar WebSockets para comunicaciones periódicas y frecuentes.
RNF4	La base de datos que emplee el servidor tiene que ser relacional y tenga soporte para datos espaciales.
RNF5	Tienen que realizarse dos aplicaciones que verifiquen el correcto funcionamiento del servidor, una aplicación de gestión y una aplicación de demostración.
RNF6	Las aplicaciones que se diseñen para comprobar la funcionalidades que ofrece el servidor tienen que presentar un diseño *responsive*, que se adapta al tamaño del dispositivo.

Tabla 2. Requisitos no funcionales

2.2. Diseño

Una vez establecidas las funcionalidades que tiene que abarcar el proyecto y las limitaciones que se presentan para llevarlo a cabo, se prepara un primer esbozo de los componentes que se van a necesitar, así como su organización. Además, se establece un modelo de datos que se adecue lo mejor posible a las necesidades del sistema.

2.2.1. Arquitectura del sistema

Con el fin de conseguir una arquitectura lo más adecuada posible para las funcionalidades que debe proporcionar el sistema, se realiza una primera aproximación, como en la Figura 5, donde se pueden distinguir tres bloques diferenciados, el almacenamiento, el servidor y el cliente. En el almacenamiento se localiza la base de datos donde se guardan todos los datos que maneja el servidor, es decir, se encarga de la persistencia del sistema. El servidor tiene la función de recibir las peticiones web y procesarlas mediante la lógica del sistema. El cliente, con las dos aplicaciones que cumplen la función de verificar las principales funcionalidades del servidor, es el responsable de realizar las peticiones web al servidor, dependiendo de los servicios que requiera en cada momento.

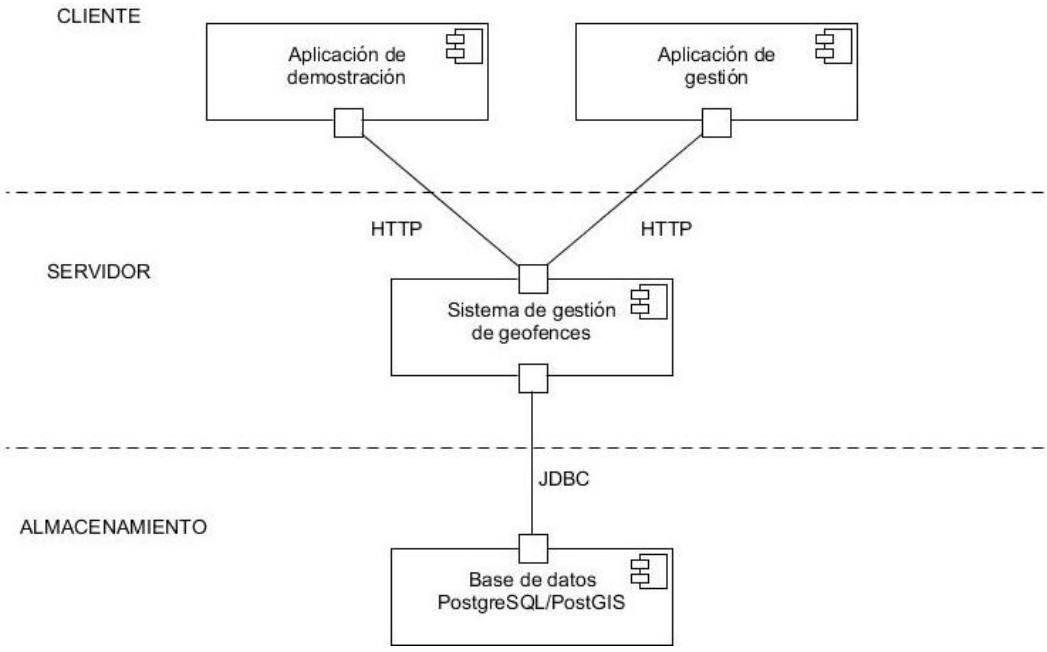

Figura 5. Diagrama de componentes y conectores – Alto nivel

Sin embargo, aunque el nivel de precisión de la Figura 5 es bastante útil para empezar a hacerse una idea de la estructura del proyecto, no proporciona la suficiente información como para conocer la organización interna del código del sistema.

Por ello se ha procedido a diseñar un segundo diagrama, Figura 6, de más bajo nivel, en la que se muestra la arquitectura en la parte del servidor y el almacenamiento. En este diagrama se abstraen gran parte de los componentes de configuración que se van a necesitar en el sistema, así como las clases que representan el modelo de datos del proyecto.

En primer lugar, la Figura 6 presenta un componente de Spring que se encarga de proporcionar la API del sistema, para que sea accesible por el cliente mediante peticiones HTTP, siguiendo el estilo de arquitectura REST. Los controladores son los encargados de gestionar las peticiones recibidas por el componente de Spring, accediendo si es necesario a la base de datos por medio de unos repositorios. Por último, existe otro componente de Spring que hace de intermediario para gestionar los accesos de los repositorios a la base de datos, encargándose también de todo el mapeo del modelo de datos del sistema en la base de datos.

Hay un servicio que se va a ofrecer mediante el empleo de WebSockets, es decir, mediante una conexión persistente entre el cliente y el servidor, y en la que ambas partes se pueden enviar datos. WebSocket [4] es una tecnología que proporciona un canal de comunicación bidireccional y *full-duplex* sobre un único socket TCP y se utiliza en el proyecto porque permite ahorrar recursos al sistema, dado que el servicio se va a solicitar de manera periódica y frecuente.

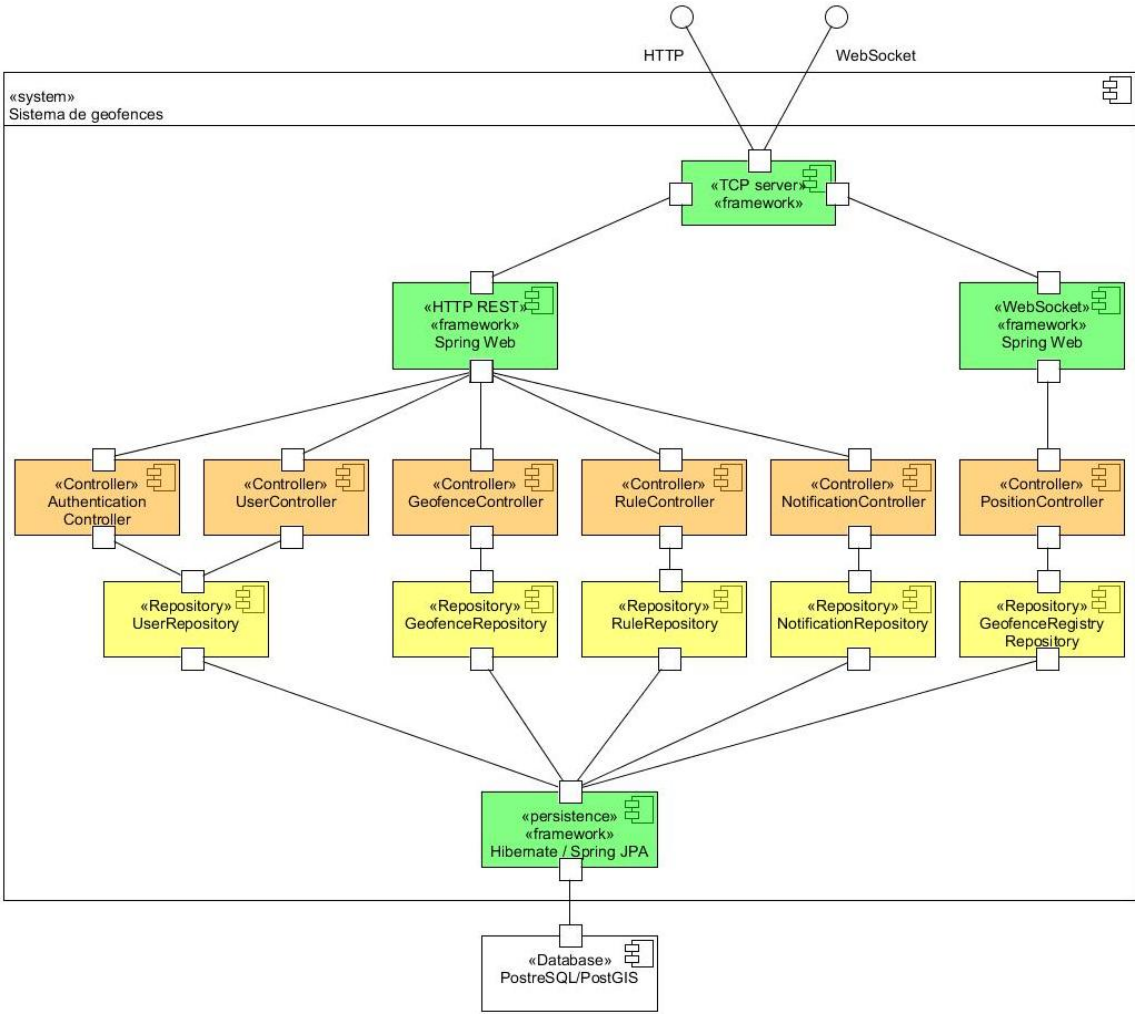

Figura 6. Diagrama de componentes y conectores – Bajo nivel

En último lugar, se ha realizado un diagrama de despliegue, Figura 7, donde se expone el despliegue físico de cada una de las partes que componen el proyecto en distintos nodos.

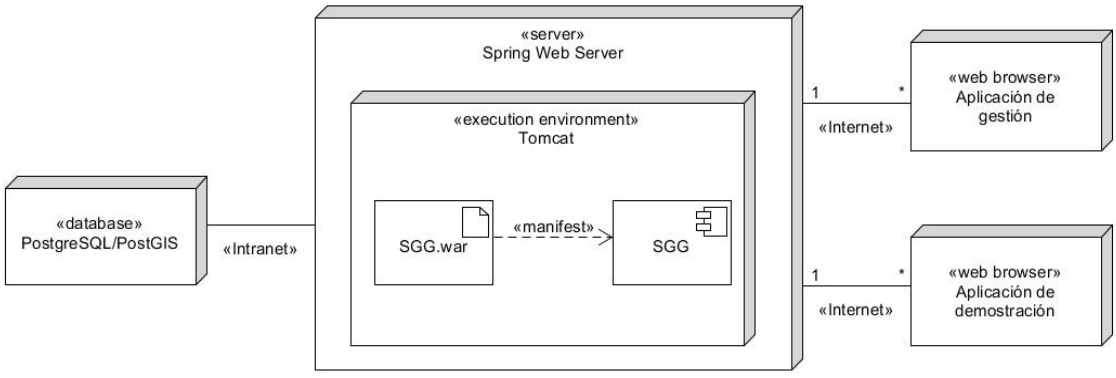

Figura 7. Diagrama de despliegue

2.2.2. Modelo de datos

A continuación se ha planteado el modelo de datos que va a seguir el sistema. Este modelo de datos, visible en la Figura 8, presenta seis entidades: usuario, geofence, regla, día, notificación y registro de geofences.

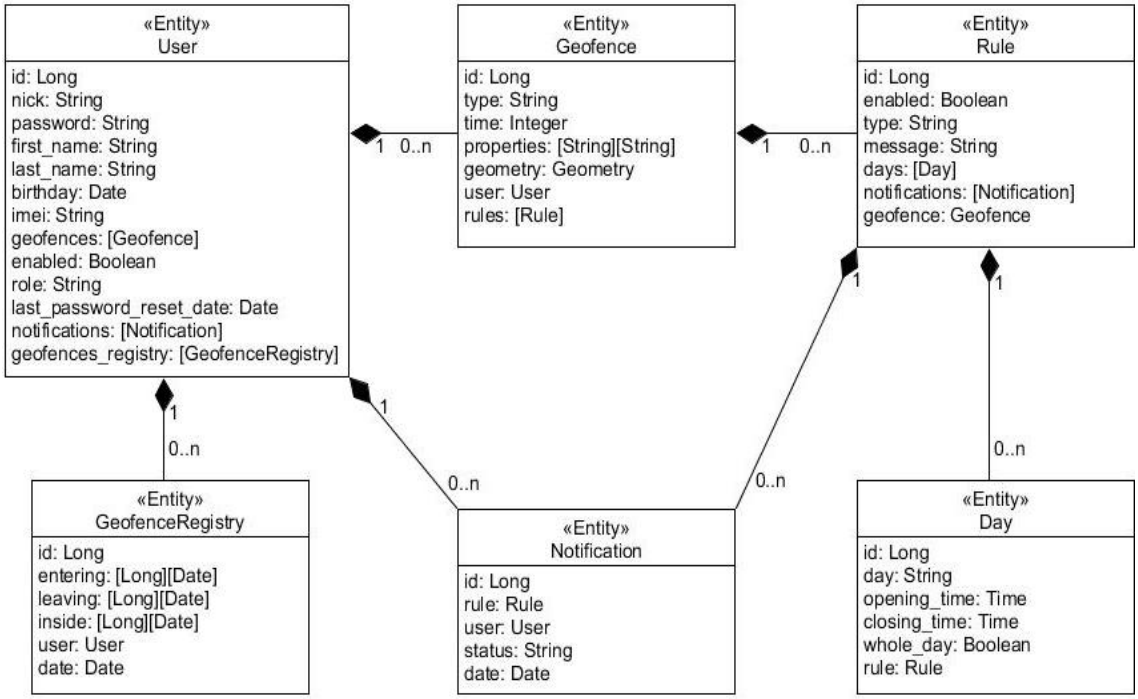

Figura 8. Modelo de datos

La entidad Usuario es la representación del cliente en el sistema, contiene datos personales del mismo como nick y contraseña, y posee un conjunto de geofences creados y notificaciones recibidas, además de un registro de geofences. La entidad Geofence es un perímetro virtual que contiene una geometría que denota su área, está ligado al usuario que lo ha creado y contiene un conjunto de reglas que especifican su comportamiento. La entidad Regla especifica un comportamiento a los geofences, tiene establecido un tipo, ya sea de entrada, de salida o de permanencia, así como el conjunto de días en los que se va a estar habilitada, el geofence al que está relacionado y las notificaciones que ha creado. La entidad Día representa un día de la semana, junto con las horas en las que está activado y está vinculado a una regla. La entidad Notificación se refiere al evento generado al cumplirse una regla, está relacionado con el usuario que lo ha recibido y la regla que la ha originado. Por último, la entidad Registro de geofences, que es un registro de los geofences en los que ha entrado, salido o permanecido dentro un usuario, contiene los datos del usuario del que anota su actividad, así como la fecha en la que se ha efectuado esa actividad.

La OGC (Open Geospatial Consortium) y la ISO (International Organization for Standardadization) definen normas respecto a las geometrías,

de las cuales va a ser empleada en el proyecto el estándar SFA (Simple Feature Access). Por esta razón, se va a utilizar la tecnología PostGIS para la base de datos y el módulo JTS para el procesamiento de las geometrías, dado que ambas cumplen con este estándar, el SFA, y hace compatible la geometría usada en el proyecto con todos aquellos que empleen este mismo estándar.

A continuación se va a describir con mayor detalle, en la Tabla 3, las reglas que se han definido en el modelo de datos del sistema, especificando los tipos que cubren, una breve descripción de cada una de ellas y su correspondiente parametrización.

Regla	Descripción	Parametrización	
Entrada	Se activa cuando se entra en un geofence	**Id**	Identificador único.
		Activado	Booleano que indica si la regla está activada o no.
		Tipo	Tipo de regla.
		Tiempo	Tiempo desde que se entra en el geofence.
		Mensaje	Mensaje asociado al evento o notificación.
		Días	Días y horas en las que la regla se ejecuta.
		Notificaciones	Notificaciones que se asocian a la regla.
		Geofences	Geofences que se asocian a la regla.
Permanencia	Se activa cuando se pasa un determinado tiempo dentro de un geofence	**Id**	Identificador único.
		Activado	Booleano que indica si la regla está activada o no.
		Tipo	Tipo de regla.
		Tiempo	Tiempo que se permanece en el geofence.
		Mensaje	Mensaje asociado al evento o notificación.
		Días	Días y horas en las que la regla se ejecuta.
		Notificaciones	Notificaciones que se asocian a la regla.
		Geofences	Geofences que se asocian a la regla.

Regla	Descripción	Parametrización	
Salida	Se activa cuando se sale de un geofence	**Id**	Identificador único.
		Activado	Booleano que indica si la regla está activada o no.
		Tipo	Tipo de regla.
		Tiempo	Tiempo desde que se sale del geofence.
		Mensaje	Mensaje asociado al evento o notificación.
		Días	Días y horas en las que la regla se ejecuta.
		Notificaciones	Notificaciones que se asocian a la regla.
		Geofences	Geofences que se asocian a la regla.

Tabla 3. Tipos de reglas asociadas a un geofence

2.2.3. Diseño de los servicios web

Los servicios web del sistema se van a organizar de forma que puedan recibir peticiones HTTP que sigan el estilo de arquitectura RESTful [5]. Esto quiere decir principalmente que tienen que cumplir con los siguientes principios.

- **Proporcionar a todos los recursos un identificador.** Establecer los principales recursos del problema y utilizar HTTP URIs para identificar todo aquello que necesite estar identificado.
- **Vincular los recursos.** Utilizar enlaces para hacer referencia a los recursos, siempre que sea posible.
- **Usar operaciones estándar.** Para interactuar con los recursos, se emplean los principales métodos estándar que presenta HTTP: GET, PUT, POST y DELETE, descritos en la Tabla 4.
- **Comunicación sin estado.** Cada mensaje HTTP contiene toda la información necesaria para comprender la petición.

Método	Descripción
GET	Obtiene una representación de un recurso especificado.
POST	Crea un nuevo recurso a partir de los datos enviados.
PUT	Actualiza un recurso especificado a partir de los datos enviados.
DELETE	Borra un recurso especificado.

Tabla 4. Operaciones HTTP

Estas peticiones HTTP pueden albergar información en el *payload*, si se requiere. Se ha decidido que toda esta información que se intercambie en los mensajes siga el formato JSON. A continuación se muestra la estructura en JSON de un geofence en la Figura 9. Las estructuras en JSON de un usuario, de una regla y de una notificación se encuentran en el ANEXO D.


```
{
    "id" : 2,
    "type" : "Feature",
    "properties" : {
        "name" : "Prueba"
    },
    "geometry" : {
        "type" : "Point",
        "coordinates" : [41.618618, -0.847992]
    },
    "user" : {
        "id" : 1,
        "nick" : "example.gmail.com",
        "password" : "$2a$10$FiigyLASJZICbTAaLZ2JCu3Cv7lPc6y2Zz25rVctpUoQoA92H81/G",
        "birthday" : "1992-08-07",
        "imei" : "356938035643809",
        "geofences" : [],
        "enabled" : true,
        "role" : "ROLE_USER",
        "notifications" : [],
        "firstName" : "First",
        "lastName" : "Last",
        "lastPasswordResetDate" : 1463608800000,
        "geofencesRegistry" : []
    },
    "rules" : []
}
```


Figura 9. Geofence en formato JSON

Los servicios web del sistema siguen el estilo de arquitectura REST, como se ha explicado previamente. A continuación se van a exponer cada uno de los servicios disponibles en la API del sistema en base a los recursos a los que conciernen. Ciertos servicios requieren autenticación para ser utilizados, otros no necesitan autenticación y algunos devuelven información diferente dependiendo de si presentan token de autenticación o no.

Los siguientes servicios web permiten la creación, modificación, obtención y borrado de geofences. También hay un servicio web que devuelve una lista de los geofences que se encuentren dentro de un determinado área.

- **GET /api/geofences.** Este método devuelve una lista de geofences de un usuario autenticado. Si el usuario no está autenticado, devuelve una lista vacía. No recibe ningún parámetro.

- **POST /api/geofences.** Este método crea un nuevo geofence. Recibe como parámetro un geofence en formato JSON y devuelve el geofence creado en el mismo formato.
- **GET /api/geofences/area.** Este método devuelve una lista de aquellos geofences que estén dentro de un área, formado por un punto específico y un radio. Recibe como parámetro la longitud del punto específico, la latitud del punto específico, el radio para formar el área a partir del punto y el máximo de geofences a devolver, siendo este último parámetro opcional.
- **DELETE /api/geofences/{id}.** Este método borra los datos de un geofence previamente creado. Recibe como parámetro el identificador del geofence y devuelve el geofence borrado.
- **GET /api/geofences/{id}.** Este método devuelve un geofence por id. Recibe como parámetro el identificador del geofence.
- **PUT /api/geofences/{id}.** Este método modifica los datos de un geofence previamente creado. Recibe como parámetro el identificador del geofence y el geofence en formato JSON, devolviendo el geofence ya modificado.

Los siguientes servicios web permiten la creación, modificación, obtención y borrado de notificaciones.

- **GET /api/notifications.** Este método devuelve una lista de notificaciones de un usuario autenticado. Si el usuario no está autenticado, devuelve una lista vacía. No recibe ningún parámetro.
- **POST /api/notifications.** Este método crea una nueva notificación. Recibe como parámetro la notificación en JSON y devuelve la notificación creada en el mismo formato.
- **DELETE /api/notifications/{id}.** Este método borra los datos de una notificación previamente creada. Recibe como parámetro el identificador de la notificación y devuelve la notificación borrada.
- **GET /api/notifications/{id}.** Este método devuelve una notificación por id. Recibe como parámetro el identificador de la notificación.
- **PUT /api/notifications/{id}.** Este método modifica los datos de una notificación previamente creada. Recibe como parámetros el identificador y la notificación a modificar en formato JSON, devolviendo la notificación ya modificada.

Los siguientes servicios web permiten la creación, modificación, obtención y borrado de reglas.

- **POST /api/rules.** Este método crea una nueva regla. Recibe como parámetro la regla en JSON y devuelve la regla creada.
- **DELETE /api/rules/{id}.** Este método borra los datos de una regla previamente creada. Recibe como parámetro el identificador de la regla y devuelve la regla borrada.
- **GET /api/rules/{id}.** Este método devuelve una regla por id. Recibe como parámetro el identificador de la regla.

- **PUT /api/rules/{id}.** Este método modifica los datos de una regla previamente creada. Recibe como parámetros el identificador y la regla a modificar en formato JSON, devolviendo la regla ya modificada.

Los siguientes servicios web permiten la creación, modificación, obtención y borrado de usuarios. Además hay un servicio web que genera tokens de autenticación y otro que refresca esos tokens.

- **POST /api/users.** Este método crea un nuevo usuario. Recibe como parámetro el usuario en JSON y devuelve el usuario creado.
- **DELETE /api/users/{nick}.** Este método borra los datos de un usuario previamente creado. Recibe como parámetro el nick del usuario y devuelve el usuario borrado.
- **GET /api/users/{nick}.** Este método devuelve un usuario por nick. Recibe como parámetro el nick del usuario.
- **PUT /api/users/{nick}.** Este método modifica los datos de un usuario creado previamente. Recibe como parámetro el nick del usuario y el usuario a modificar en formato JSON, devolviendo el usuario ya modificado.
- **POST /api/users/auth.** Este método autentica a un usuario dentro del sistema. Recibe como parámetro el usuario y la contraseña dentro un objeto y el dispositivo desde el que se envía la petición, devolviendo el token de autenticación.
- **GET /api/users/refresh.** Este método refresca la autenticación de la cabecera de una petición y autentica a un usuario dentro del sistema de nuevo. Recibe la petición y devuelve un nuevo token de autenticación.

2.2.4. Diseño de los servicios de mensajería

Respecto a los servicios de mensajería del sistema, se va a emplear la tecnología WebSockets para ofrecerlos. Como se ha explicado con anterioridad, WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y *full-duplex* sobre un único socket TCP. El protocolo de comunicación que se va a usar en los WebSockets es STOMP [6], que es un protocolo de mensajería orientado a texto. Este protocolo proporciona un formato de conexión interoperable que permite a los clientes comunicarse con casi todos los *brokers* de mensajería existentes.

Para proveer los servicios de mensajería del sistema, se ha de configurar el bróker que se encargue de gestionar la conexión y los mensajes, por lo que se deben de especificar los *endpoints* por los que se va a establecer la conexión WebSocket, los destinos o direcciones que va emplear el cliente para enviar los mensajes, así como los *topics* o temas por los que el servidor va a responder. Los destinos y los *topics* son efectivos únicamente de manera interna en los WebSockets, es decir, estas direcciones son virtuales, están vinculadas al protocolo STOMP y no existen para el resto del sistema.

Por lo tanto, el *endpoint* que se ha decidido definir en la configuración del bróker del sistema es el siguiente.

- **/api/locations.** Es el *endpoint* por el que se crea la conexión WebSocket y por donde se envían las peticiones entre el servidor y el cliente cuando emplean esta tecnología.

A continuación se exponen las direcciones o destinos que se van a especificar en el bróker del sistema encargado de la mensajería y que van a emplear el cliente y el servidor para comunicarse, como se puede ver en la Figura 10.

- **/api/locations.** Es la dirección de mensajería del protocolo STOMP por la que el cliente va a enviar los mensajes al servidor (no confundir con el *endpoint /api/locations* anteriormente descrito).
- **/topic/positions.** Es la dirección de mensajería del protocolo STOMP por la que el servidor va a responder al cliente.

Figura 10. Traza de mensajes que se intercambian el servidor y el cliente vía WebSockets

2.3. Implementación

En este apartado se van a tratar todos los temas relacionados con la implementación del servidor, de cómo se han utilizado distintas tecnologías con el objetivo de conseguir las funcionalidades deseadas por el cliente, así como la lógica interna del sistema para obtener el comportamiento esperado en cada uno de los posibles escenarios a los que se enfrente el sistema.

2.3.1. Uso del *framework* Spring

Se ha empleado el *framework* Spring para facilitar la implementación del sistema, dado que ofrece una serie de módulos que simplifican la codificación de ciertas funcionalidades. Las áreas en las que se ha utilizado son:

- **Servicios web.** El *framework* Spring ofrece un módulo llamado Spring Web, el cual permite crear controladores que puedan recibir peticiones HTTP siguiendo el estilo de arquitectura REST, y transformar de manera automática los objetos que se reciben o se envían a JSON mediante el uso de la clase *Jackson2ObjectMapperBuilder*, que configura la librería Jackson[12] que es el sistema responsable de convertir objetos Java a JSON y viceversa. Además, se puede configurar añadiéndole módulos, con el fin de que reconozca ciertas clases que en un principio no sería capaz de mapear. En el proyecto se ha añadido el módulo JTS de Vivid Solutions a la configuración, para que identifique en forma canónica la clase *Geometry*, que es la que establece la geometría de un geofence.
- **Seguridad.** El módulo de Spring encargado de la seguridad, Spring Security, se encarga de proporcionar un control de acceso mediante roles a la API del sistema, comprobando que cada petición tenga los permisos requeridos en cada caso.
- **Acceso a datos.** Spring presenta un módulo llamado Spring Data JPA, que se encarga del acceso a la base de datos, en este caso una base de datos relacional como es PostgreSQL.
- **WebSockets.** Este *framework* presenta varios módulos como Spring Messaging y Spring WebSocket que facilitan la implementación de los WebSockets en el sistema.

2.3.2. Mapeo del modelo de datos

La base de datos que se ha empleado para el almacenamiento de los datos es PostgreSQL/PostGIS. Esto se debe a que se necesita soporte para el almacenamiento de datos espaciales (geometría de un geofence), soporte que no presenta por defecto la base de datos PostgreSQL, ni otras bases de datos relacionales convencionales. Con respecto a las no relacionales como MongoDB, sí ofrecen soporte para datos espaciales, pero como era requisito indispensable emplear una base de datos relacional en el proyecto, se decidió que la mejor opción era añadir el módulo PostGIS a la base de datos PostgreSQL, ya que este módulo provee soporte para la gestión de objetos espaciales, que es lo que se quiere. De no haber solucionado el problema de esta manera, no se habrían podido guardar las geometrías de los geofences.

Para generar las tablas en las que se van a almacenar los datos, se ha optado por utilizar la tecnología Hibernate, que es una herramienta de mapeo objeto-relacional (ORM) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación. De igual

[12] https://github.com/FasterXML/jackson

manera, para que Hibernate pueda reconocer los datos espaciales se ha empleado la extensión llamada Hibernate Spatial, que utiliza el módulo JTS que se ha comentado antes. Por lo tanto, esta extensión sabe interpretar la clase *Geometry* de un geofence y para que pueda reconocerla en el código hay que señalar esta clase con la etiqueta *org.hibernate.spatial.GeometryType.* De esta forma sabrá transformarla adecuadamente para su guardado en la base de datos.

El módulo JTS de Vivid Solutions que emplea también Hibernate Spatial se encarga de proporcionar soporte a las geometrías, que son objetos espaciales definidos por un conjunto de puntos. Estas geometrías que establecen el área de un geofence son convertidas por el módulo JTS en forma canónica para facilitar su tratamiento.

Para que Hibernate reconozca correctamente cada una de los modelos de datos que se desea que se mapeen en la base de datos, en este caso, la base de datos relacional PostgreSQL/PostGIS, se deben de colocar etiquetas que identifiquen cada una de las entidades (*Entity*), así como etiquetas que describan cada uno de los atributos de las entidades (*Column*), además de las etiquetas que denoten las relaciones de dependencia que existen entre los distintos objetos (*OneToMany* o *ManyToOne*).

Una vez hecho esto, al lanzar el servidor se generan de manera automática las tablas de los objetos que Hibernate mapea, manteniendo las relaciones dentro de la base de datos y siendo accesibles desde el servidor a través de repositorios, que usan el módulo Spring Data JPA, como se puede apreciar en la Figura 11.

```
Hibernate: create table geofences (id int8 not null, geometry GEOMETRY not
null, type varchar(20) not null, user_ID int8, primary key (id))
Hibernate: create table notifications (id int8 not null, date date not null,
status varchar(15) not null, rule_ID int8, user_ID int8, primary key (id))
Hibernate: create table rules (id int8 not null, enabled boolean not null,
message varchar(130) not null, time int4 not null, type int4 not null,
geofence_ID int8, primary key (id))
```

Figura 11. Traza de creación de tablas de Hibernate

Los repositorios del sistema encargados de acceder a la base de datos extienden la interfaz *CrudRepository*, que presenta una serie de métodos básicos para la interacción con una base de datos, como puede ser la búsqueda de un objeto por identificador, el borrado de un objeto por identificador o el guardado de un objeto. En caso de que se necesite el empleo de alguna operación que no proporcione la interfaz *CrudRepository* para acceder a la base de datos, el módulo Spring Data JPA facilita un conjunto de etiquetas para implementarla utilizando una especie de pseudocódigo propio, bastante parecido al lenguaje SQL y que evita tener que codificar la funcionalidad dentro de esa operación, dado que el pseudocódigo que se escriba es el que se va a ejecutar. Para realizar consultas que no modifican los

datos de la base de datos se emplea la etiqueta *Query*, mientras que si se quiere actualizar o borrar datos de la base de datos se tiene que colocar la etiqueta *Modifying* al método, para indicar que la función va a cambiar datos, acompañada de la etiqueta *Transactional*, que señala que la función se ejecuta dentro de una transacción, es decir, que la operación se realiza o no, pero nunca se queda a medias.

Por otra parte, como se puede apreciar en la Figura 8, el modelo de datos es complejo y contiene muchas relaciones entre las entidades. Esto genera un problema a la hora de interpretar los datos en JSON, que es el formato que se va a emplear para enviar la información en las peticiones que realice tanto el servidor como el cliente. Al representar estos datos en formato JSON se generaban bucles, debido a las dependencias señaladas entre las entidades. Esto se ha solucionado delimitando la información que cada entidad va a devolver cuando se transforme a JSON sus datos con el uso de vistas. Añadiendo la etiqueta *JsonView,* en la que se ha de colocar la interfaz que se va a manejar para indicar a la librería Jackson los atributos que se quiere que se devuelvan al formar el JSON, se evitan estos bucles indefinidos.

2.3.3. Servicios web

El *framework* Spring proporciona una configuración predeterminada para la implementación de la seguridad en un sistema por medio del módulo Spring Security. Sin embargo, esta configuración no está pensada para su utilización en sistemas que provean servicios web de tipo REST, sino para los de tipo MVC (Modelo-Vista-Controlador), por lo que hay que reconfigurar partes del sistema, como por ejemplo los permisos de acceso a servicios web, para que funcionen de la manera deseada. El módulo de Spring es bastante estricto en cuanto a los cambios sobre la configuración predeterminada, por lo que no ha sido una tarea trivial.

Otra opción hubiera sido emplear el MVC en el sistema para proporcionar los servicios aprovechando la implementación de Spring por defecto, pero esto significaría que para hacer uso de los servicios del sistema habría que interactuar previamente con la capa de presentación o vista del mismo, en vez de hacer peticiones directamente a los servicios web. El sistema debe ofrecer sus servicios web sin intermediarios, por lo que este hecho, unido a la intención de crear aplicaciones verificadoras en distintos servidores y no en el mismo sistema, hizo desestimar la idea.

Los servicios web devuelven un tipo de información diferente en algunos casos dependiendo de si el usuario presenta o no autenticación. Esta diferenciación se realiza aprovechando la etiqueta *JsonView* que aparece en las entidades, que permite especificar los atributos a devolver al formar el JSON. Esta etiqueta es utilizada por la librería Jackson. Por ello, se han creado dos vistas para las entidades Usuario, Geofence, Regla y Notificación, una

vista básica con datos no comprometidos ni sensibles y una vista completa, con toda la información de la que se dispone en la base de datos. De esta manera, dentro de cada servicio web se ha comprobado si la petición adjunta token de autenticación, para que en caso afirmativo devuelva la vista completa y en caso negativo, la vista básica.

En la Tabla 5 se muestra el contenido devuelto por aquellos servicios web que no requieren necesariamente autenticación y que, por lo tanto, varían la información que devuelven en función de si la petición presenta token de autenticación o no cuando se realiza. Todos aquellos servicios web que no se encuentren en la tabla requieren siempre autenticación y devuelven la vista completa de su correspondiente recurso.

API	Token	Configuración
GET /api/geofences/area	Sin token	Vista de geofence básica
	Con token	Vista de geofence completa
GET /api/geofences/{id}	Sin token	Vista de geofence básica
	Con token	Vista de geofence completa
GET /api/notifications/{id}	Sin token	Vista de notificación básica
	Con token	Vista de notificación completa
GET /api/rules/{id}	Sin token	Vista de regla básica
	Con token	Vista de regla completa
POST /api/users	Sin token	Vista de usuario completa
	Con token	Vista de usuario completa
GET /api/users/{nick}	Sin token	Vista de usuario básica
	Con token	Vista de usuario completa

Tabla 5. Vistas devueltas en función de token de autenticación

Por otra parte, se busca que la información que envíe y reciba el servidor referente a geofences siga un estándar de codificación de elementos geográficos. Por esta razón se ha decidido que la representación de un geofence adopte el formato GeoJSON[13], que es un estándar de codificación de datos geográficos entendido por todos los clientes de visualización de mapas

La transformación a JSON se realiza mediante el uso de la librería Jackson, que es el mapeador que emplea por defecto Spring. Sin embargo, este mapeador no sabe tratar objetos espaciales, por lo que hay que añadirle el módulo JTS de Vivid Solutions a la configuración, para que traduzca a forma canónica la clase *Geometry*, que es la que establece la geometría de un geofence. Una vez hecho esto, la representación resultante del mapeador ya es capaz de generar un GeoJSON para los geofences y JSON para los usuarios, reglas y notificaciones.

[13] http://geojson.org/

Por último, el servidor ha presentado problemas con la especificación CORS[14], que es el mecanismo que permite que una página web pueda solicitar recursos restringidos (en particular peticiones AJAX y WebSockets) a un dominio distinto del que procede la página web. La configuración por defecto de Spring no permite realizar peticiones a un dominio distinto. Este inconveniente se ha solucionado modificando la configuración del CORS, aceptando peticiones de cualquier origen, así como de los siguientes tipos: GET, POST, PUT, DELETE y OPTIONS.

2.3.4. Servicios de mensajería

Aparte de la configuración especificada anteriormente respecto a la configuración del bróker de mensajería, indicando los *endpoints* en los que se crean las conexiones WebSockets, los destinos o direcciones por los que los clientes envían los mensajes y los *topics* o temas por los que responde el servidor, la comunicación vía WebSocket se enfrenta a otro problema, la configuración de control de accesos.

En un principio, se creía que el control de accesos afectaba únicamente a los servicios web del sistema. Sin embargo, se ha comprobado que era una deducción errónea. Por defecto, se ha establecido que todas las direcciones no especificadas en la configuración de accesos requieran autenticación, por lo que la conexión WebSocket también se encuentra influenciada por esta medida. Cuando el cliente se intenta comunicar con el servidor por medio de WebSockets, el sistema le pide autenticación para acceder al servicio. Esto se ha solucionado permitiendo el uso de las direcciones empleadas en la comunicación WebSocket a todo el mundo, sin necesidad de un token de autenticación. Sin embargo, esto significa que cualquiera pueda utilizar este servicio, aunque no esté registrado ni autenticado en el sistema. Como tampoco se encuentra deseable esto, se ha optado por mantener el acceso a los servicios de mensajería sin restricciones de autenticación, pero comprobando de manera interna dentro de cada servicio WebSocket si el usuario envía el token de autenticación o no, y en caso de no enviarlo, no ejecutar la lógica del método, como se puede ver en la Figura 10.

2.3.5. Autenticación

Este sistema requiere de un control de accesos para determinados métodos que proporciona la API. Para ello, se ha hecho uso de Spring Security, el cual permite establecer un control de accesos basado en roles.

En primer lugar hay que configurar el sistema para que el módulo de seguridad pueda acceder a la base de datos y comprobar los permisos de un usuario registrado, es decir, se debe de indicar al *authentication manager builder* o constructor del gestor de autenticación de Spring Security el método

[14] https://www.w3.org/TR/cors/

que tiene que utilizar para acceder al usuario en la base de datos y así poder comparar la contraseña. Posteriormente se han de señalar los diferentes permisos que se requieren para el empleo de cada uno de los métodos que presenta el sistema, ya sea permitir el acceso a usuarios autenticados y no autenticados, solo a los autenticados en el sistema o únicamente a los usuarios que pertenezcan a un determinado rol.

La configuración elaborada en el control de accesos del sistema se representa en la Tabla 6, donde se puede apreciar si se necesita el token de autenticación o no para acceder a determinados servicios. Se ha establecido que los métodos que no aparezcan especificados en la configuración, en caso de que se amplíe la API, requieran el token de autenticación para ser usados.

API	Configuración
GET /api/geofences/area	Autenticación no requerida
GET /api/geofences/{id}	
GET /api/notifications/{id}	
GET /api/rules/{id}	
POST /api/users	
GET /api/users/{nick}	
POST /api/users/auth	
GET /api/geofences	Autenticación requerida
POST /api/geofences	
DELETE /api/geofences/{id}	
PUT /api/geofences/{id}	
GET /api/notifications	
POST /api/notifications	
DELETE /api/notifications/{id}	
PUT /api/notifications/{id}	
POST /api/rules	
DELETE /api/rules/{id}	
PUT /api/rules/{id}	
DELETE /api/users/{nick}	
PUT /api/users/{nick}	
GET /api/users/refresh	

Tabla 6. Requisitos de autenticación para la API del sistema

Una vez establecida y configurada la seguridad básica proporcionada por el módulo Spring Security, se ha decidido mejorarla introduciendo el empleo de JWT, cuya estructura se puede apreciar en la Figura 12, para la autenticación en el sistema. Esto quiere decir que ahora la autenticación se basa en tokens, dado que JWT es una codificación de token de seguridad basado en JSON que permite que la información de identidad y seguridad se comparta entre los dominios de seguridad.

Cada una de las peticiones que se hagan al sistema debe de ir acompañada de un token de autenticación generado por el sistema cuando te identificas, si se quiere realizar estando autenticado. JWT es más seguro que HTTP Basic, que es el sistema de autenticación que presentaba el sistema previamente, entre otras cosas, porque evita tener que estar pasando en cada petición el nombre de usuario y la contraseña, como en HTTP Basic, al sustituirlo por un token. Además, el token expira con regularidad, no sucediendo lo mismo con las credenciales. Por otra parte, se ha tenido que deshabilitar la autenticación CSRF que contiene Spring Security por defecto porque este tipo de autenticación está orientada a aplicaciones web y el sistema no va a ofrecer aplicaciones web, sino servicios web, por lo que no se necesita.

Figura 12. Estructura de un JSON Web Token

Por otra parte, la contraseña de todos los usuarios del sistema se almacena encriptada, aportando un mayor grado de seguridad al usuario. La encriptación se realiza con la ayuda de la clase *BCryptPasswordEncoder*, que utiliza la función de encriptado *hashing* de passwords *bcrypt* [7] para llevar a cabo ese cometido. Esta clase se debe de especificar en el constructor del gestor de autenticación de Spring Security para que a la contraseña enviada por el usuario se le aplique una función hash previamente antes de compararse con la guardada en la base de datos y, en caso de coincidir, generar el token de autenticación.

2.3.6. Gestión de localización mediante reglas

Una parte fundamental de las funcionalidades del sistema es el envío de notificaciones cuando se cumple una regla asociada a un geofence. En el sistema se han definido tres tipos de regla: de entrada, de salida y de permanencia.

Las reglas de entrada se utilizan para que se envíen notificaciones pasados *n* segundos desde la entrada del usuario en el geofence, las reglas de salida se utilizan para que se envíen notificaciones pasados *n* segundos desde la salida del usuario del geofence y las reglas de permanencia se utilizan para que se envíen notificaciones pasados *n* segundos desde que el usuario ha entrado en el geofence, manteniéndose dentro del mismo todo el tiempo.

Para facilitar la comprensión de la lógica implementada, se muestra un caso práctico en la Figura 13, en la que un usuario entra en un geofence que contiene una regla de permanencia dentro del geofence antes del envío de la notificación.

En primer lugar, es necesario que el usuario esté autenticado en el sistema para poder enviar su posición al sistema, por lo que en caso de no estarlo, se requiere solicitar la autenticación previamente al sistema.

Posteriormente, la aplicación de demostración envía la posición del usuario al sistema por medio del WebSocket, donde se comprueba si se encuentra dentro de algún geofence o no. En caso de que no se halle dentro de ningún geofence, no se realiza ninguna acción. Sin embargo, si se encuentra dentro de uno o varios geofences, se obtienen las reglas que tienen asociadas cada uno de esos geofences, comprobando la consecución de estas mismas. De cumplirse la regla, se genera su notificación correspondiente y se guarda asociándola al usuario que ha enviado su posición al sistema.

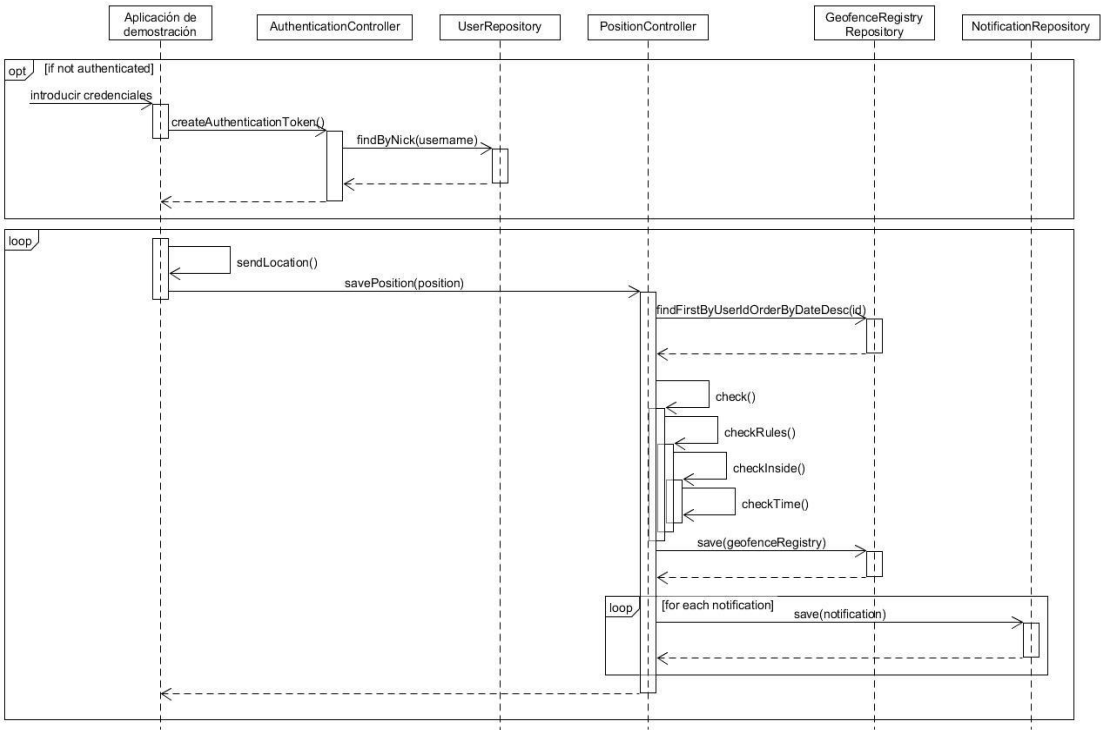

Figura 13. Diagrama de secuencia

2.3.7. Pruebas funcionales

Con el objetivo de comprobar el correcto funcionamiento de todas las funcionalidades contenidas en el proyecto, se ha procedido a realizar pruebas unitarias, de sistema y de integración de todos los principales métodos. A continuación se va a explicar con mayor detalle en que ha consistido cada una de estas pruebas [8].

- **Pruebas unitarias.** Consisten en la ejecución de actividades que permitan verificar que los componentes unitarios están codificados bajo condiciones de robustez, es decir, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Con este tipo de prueba se ha comprobado la lógica de la gestión de localización mediante reglas.
- **Pruebas de integración.** Consisten en la comprobación de que elementos del software que interactúan entre sí, funcionan de manera correcta. Se han realizado pruebas de integración para cada una de los métodos que ofrece la API del sistema.
- **Pruebas de sistema.** Consisten en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos en el proyecto. Es importante que los tipos de pruebas ejecutados en este nivel se desplieguen en un ambiente de pruebas cuya infraestructura y arquitectura sea similar al ambiente de producción. Se ha hecho una prueba de sistema, por ejemplo, para comprobar el método de autenticación del sistema y, posteriormente, utilizar el token obtenido en un método que lo requiere.

2.4. Demostrador y evaluación del sistema

La realización de un demostrador en el proyecto está motivada básicamente en el deseo y necesidad de probar en un entorno real las funcionalidades implementadas en el sistema. Por ello, se ha decidido desarrollar dos aplicaciones web desplegadas en servidores distintos al del sistema principal, es decir, en servidores de presentación; una centrada en la gestión de los recursos identificados en el sistema y otra más orientada en la demostración de la lógica del sistema en cuanto al envío de localización respecto de los geofences y la obtención de las subsiguientes notificaciones.

2.4.1. Prototipos

Para poder empezar a implementar las aplicaciones, primero hay que tener una idea bastante clara del alcance y la apariencia. Por ello, hay que realizar unos prototipos en los que se defina cómo va a ser la GUI. Algo que tienen en común las dos aplicaciones es que ambas van a presentar una pantalla de inicio de sesión, la Pantalla A de las siguientes figuras.

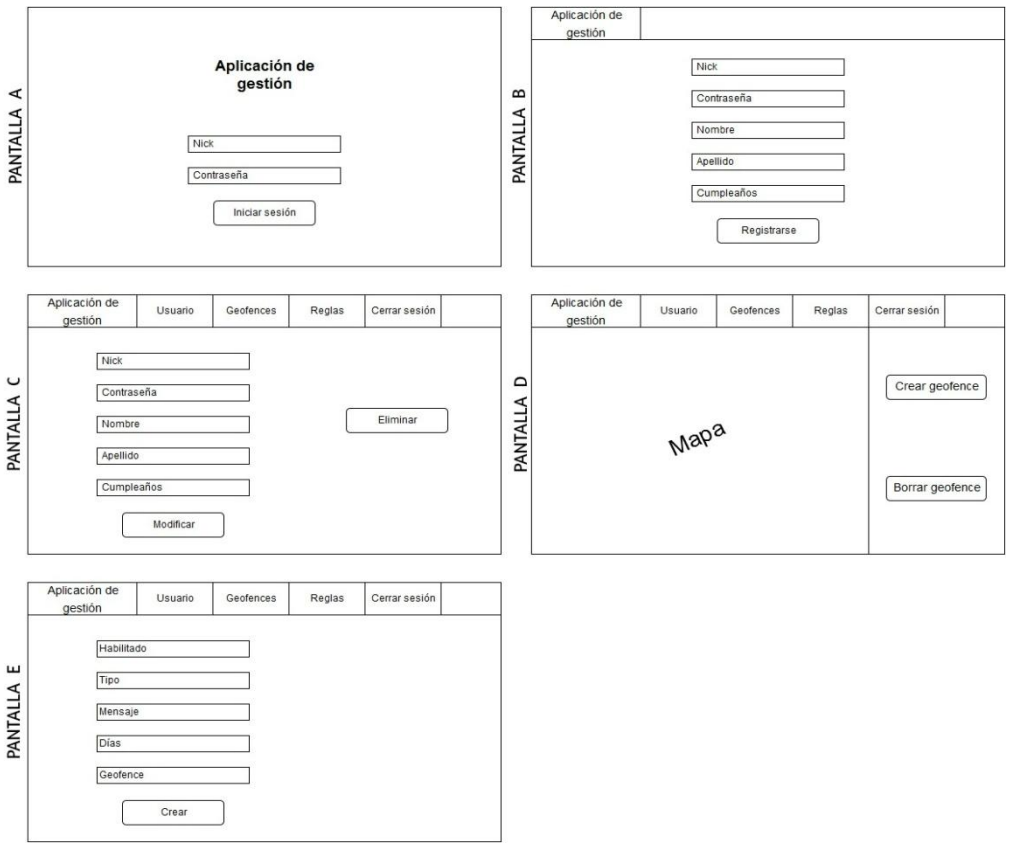

Figura 14. Prototipos de la aplicación de gestión

Para la aplicación de gestión se han diseñado un par de interfaces gráficas que permitan la creación, modificación y borrado de usuarios, como se puede apreciar en la Figura 14, Pantallas B y C. A su vez, también se ha planteado una interfaz que permita la creación y borrado de geofences, visible en la Pantalla D, así como otra que facilite la creación de reglas para los geofences existentes en el sistema, como se muestra en la pantalla E.

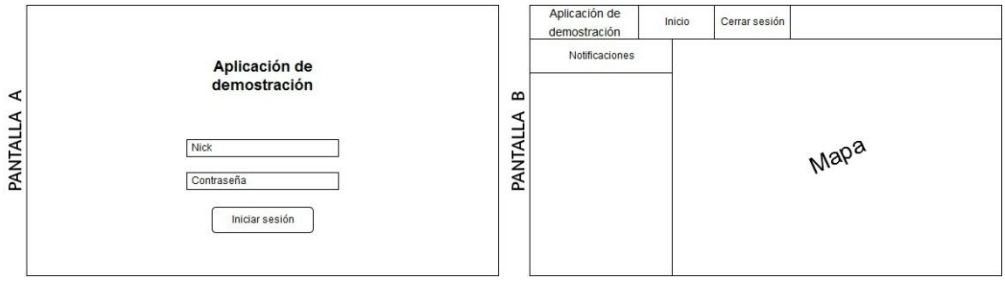

Figura 15. Prototipos de la aplicación de demostración

Para la aplicación de demostración se ha dibujado una única pantalla que muestra las notificaciones obtenidas en una lista, así como un mapa en el que dibujar los geofences almacenados en el sistema, como se puede ver en la Figura 15, Pantalla B.

2.4.2. Mapa de navegabilidad

Con respecto a los mapas de navegabilidad de cada una de las aplicaciones, en la Figura 16 se muestra el mapa de la aplicación de gestión, manteniendo cada pantalla su correspondiente numeración alfabéticamente como en el apartado anterior. El mapa de navegación de la aplicación de demostración se encuentra en la Figura 17.

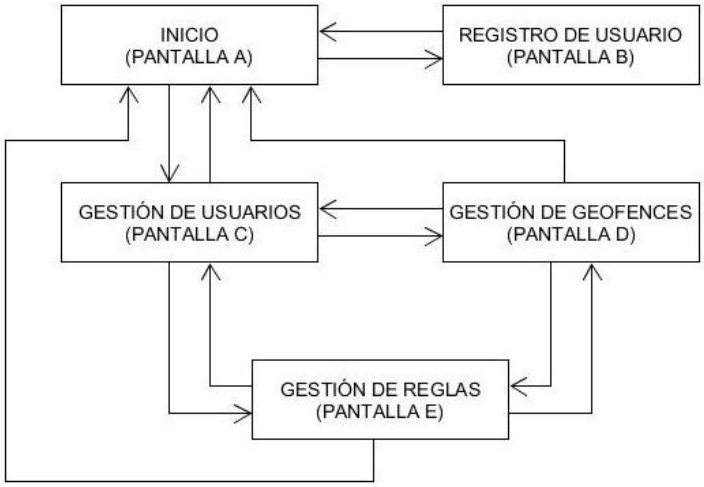

Figura 16. Mapa de navegabilidad de la aplicación de gestión

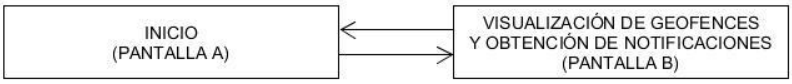

Figura 17. Mapa de navegabilidad de la aplicación de demostración

A continuación se expone el resultado final de alguna de las interfaces gráficas realizadas para la aplicación de gestión, como en la Figura 18, y para la aplicación de demostración, como en la Figura 19.

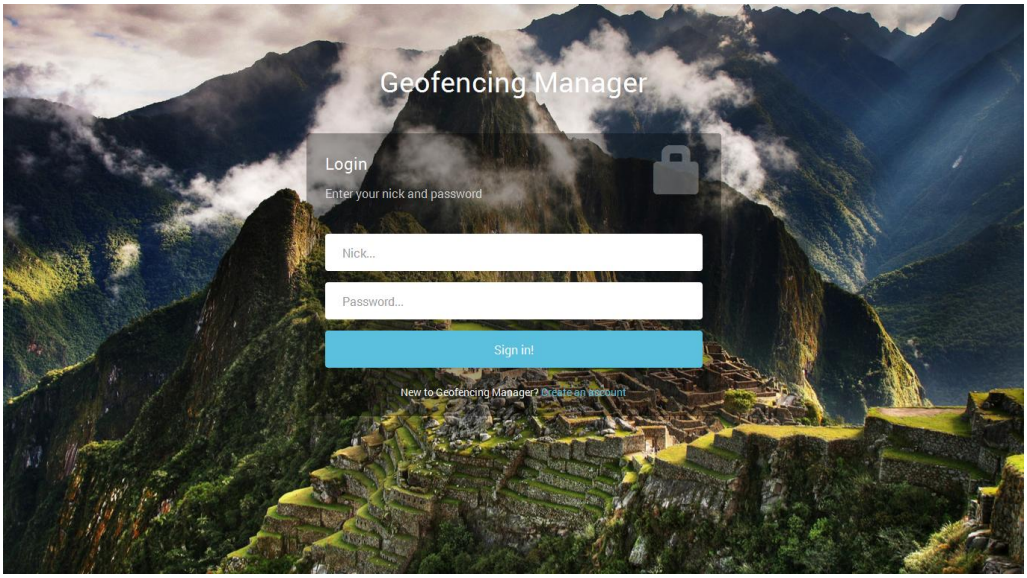

Figura 18. Interfaz gráfica de inicio de sesión de aplicación de gestión

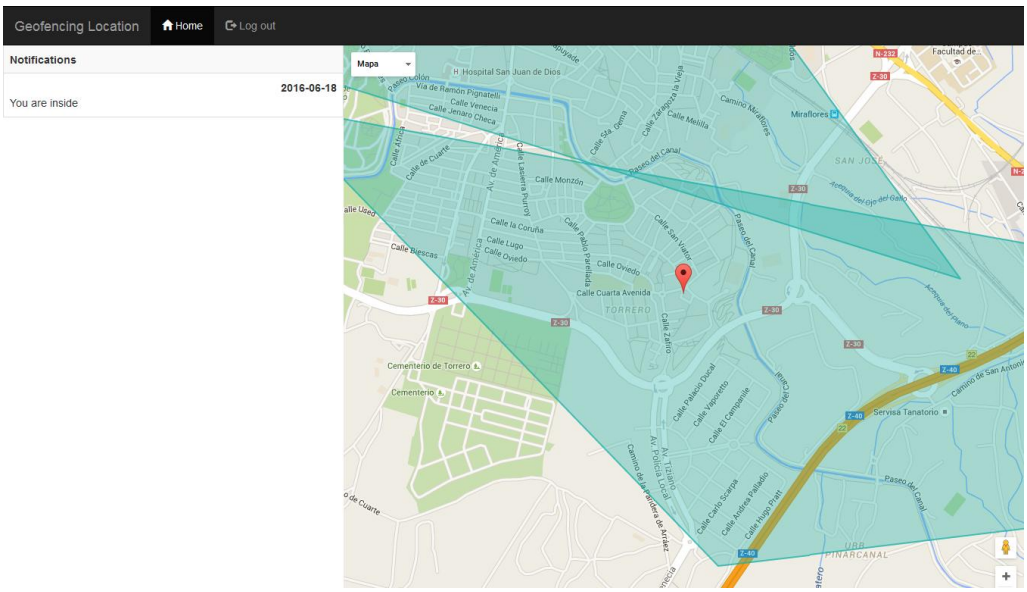

Figura 19. Interfaz gráfica de menú de aplicación de demostración

2.4.3. Aplicación de gestión

Respecto a la implementación de la aplicación de gestión, se ha hecho uso del *framework* Spring para que se encargue del montaje del servidor de presentación en el que se despliega la aplicación web, así como para la gestión de las peticiones.

Por otra parte, también se utiliza Thymeleaf [9] en la aplicación, que es una librería que permite construir y visualizar la capa de presentación usando tecnología XHTML o HTML5 sin necesidad de usar un servidor de aplicaciones web, únicamente necesitando una máquina virtual JVM para compilar, depurar, probar y desplegar. Lo cual resulta muy útil en la etapa de desarrollo de la aplicación, dado que permite evitar estar continuamente lanzando el servidor para comprobar el resultado final de la interfaz gráfica. Para elaborar la interfaz gráfica de la aplicación se ha empleado Bootstrap, que es un *framework* que contiene plantillas de diseño *responsive*.

Esta aplicación hace llamadas al sistema para realizar operaciones sobre los usuarios, los geofences y las reglas, con el fin de verificar su correcto funcionamiento. Respecto a las geofences, para su creación y visualización se ha empleado la API de Google Maps. La lógica de la aplicación está escrita en JavaScript, utilizando también la librería jQuery, implementada en el mismo lenguaje y que permite, entre otras cosas, agregar interacción con la técnica AJAX a la aplicación. Por tanto, para efectuar las peticiones al servidor se ha empleado la tecnología AJAX, una técnica de desarrollo web para crear aplicaciones interactivas que se ejecutan en el cliente manteniendo la comunicación asíncrona con el servidor en segundo plano.

Por último, al desarrollar esta aplicación se ha descubierto un problema en la parte del servidor del sistema con el CORS, el mecanismo de intercambio de recursos de origen cruzado, dado que no permitía la entrada de las peticiones de la aplicación porque el origen de las mismas no estaba configurado y, por tanto, permitido. Este inconveniente se ha solucionado modificando la configuración por defecto del CORS del sistema, permitiendo aceptar todas las peticiones, independientemente del origen de las mismas.

2.4.4. Aplicación de demostración

Respecto a la implementación de la aplicación de demostración, también se ha utilizado Spring para que se monte de forma automática el servidor y para gestionar las peticiones enviadas por el usuario. De igual manera se ha añadido Thymeleaf y Bootstrap a la aplicación, en la que la lógica está implementada en JavaScript junto con jQuery para realizar peticiones asíncronas con AJAX.

Esta aplicación hace peticiones al sistema relacionadas con la obtención de los geofences y de las notificaciones de un usuario determinado, previamente autenticado. Para que se produzcan las notificaciones el servidor necesita saber la posición del usuario y de esta manera poder averiguar si el usuario está dentro de un geofence con unas determinadas reglas, que en caso de cumplirse, generarán estas notificaciones.

La aplicación tiene que enviar la posición del usuario para que el servidor pueda gestionarla y para ello se necesita establecer la conexión mediante WebSockets que ofrece el sistema para este cometido. Para establecer la conexión vía WebSocket en la parte del cliente se va a hacer uso de SockJS, así como del protocolo STOMP para la comunicación. Con SockJS básicamente se crea el *endpoint* del WebSocket por el que se van a intercambiar mensajes el servidor y el cliente, en este caso se ha decidido llamarlo /api/locations. Una vez establecido el *endpoint* por el que se van a comunicar, las direcciones y *topics* o temas establecidos en la configuración del servidor son las que van a tener que utilizar para la transmisión de mensajes entre ellos.

Junto con la posición que envía el cliente, se adjunta el token de autenticación que verifica posteriormente el servidor, dado que en caso que el usuario no esté autenticado, no se gestionará la posición por parte del sistema.

Por último, los geofences se enseñan en un mapa proporcionado por Google Maps, donde a partir de la geometría de los geofences obtenidos del sistema se dibujan por medio de la API que proporciona este servicio. La posición del usuario se consigue utilizando la API del servicio Geolocation [10].

3. GESTIÓN DEL PROYECTO

En relación a la planificación que se ha seguido para la realización del proyecto, como ya se ha expuesto con antelación, se ha empleado una metodología ágil basada en iteraciones. A continuación se va a mostrar con más detalle la planificación del proyecto, así como las horas dedicadas y las herramientas utilizadas.

3.1. Planificación

Como se puede apreciar en la Figura 20, la fecha de inicio del proyecto fue el día 10 de Febrero de 2016, con una fase preliminar, cuya misión era aclarar tanto el alcance como las funcionalidades a cubrir por el proyecto, que finaliza el 21 de Febrero de 2016 y deriva en el inicio de las iteraciones, que tienen 2 semanas de duración y conllevan una reunión entre el autor y el director del proyecto al término de cada una.

Estas iteraciones, en las que realizan tareas de análisis, diseño, implementación, pruebas y documentación, tanto del sistema como de las aplicaciones verificadoras, concluyen el día 12 de Junio de 2016. Posteriormente comienza una breve semana dedicada a terminar de perfeccionar la memoria hasta el 19 de Junio de 2016, que es cuando la memoria tiene que estar finalizada y lista para su entrega y depósito, según las previsiones realizadas en el proyecto.

Nombre	Fecha de inicio	Fecha de fin
Análisis y Diseño	10/02/16	6/03/16
Fase preliminar	10/02/16	21/02/16
Iteración 1	22/02/16	6/03/16
Implementación y pruebas	7/03/16	12/06/16
Iteración 2	7/03/16	20/03/16
Iteración 3	21/03/16	3/04/16
Iteración 4	4/04/16	17/04/16
Iteración 5	18/04/16	1/05/16
Iteración 6	2/05/16	15/05/16
Iteración 7	16/05/16	29/05/16
Iteración 8	30/05/16	12/06/16
Documentación	13/05/16	19/06/16
Entrega del proyecto	20/06/16	24/06/16
Preparación de presentación	25/06/16	3/07/16
Defensa del proyecto	4/07/16	8/07/16

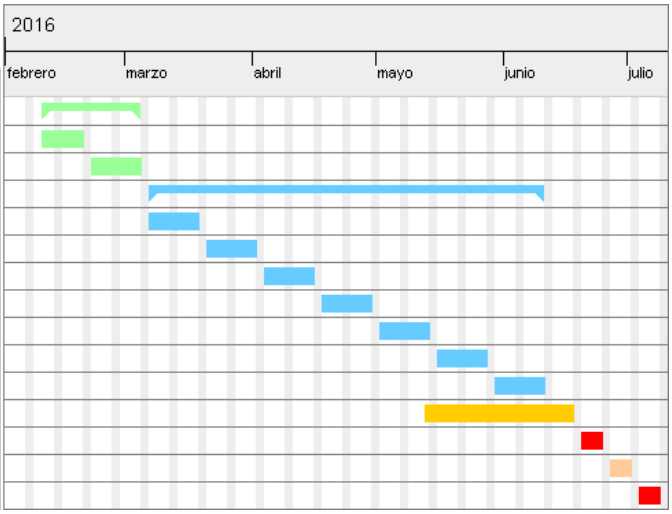

Figura 20. Diagrama de Gantt

3.2. Tiempo dedicado

Respecto al tiempo dedicado en la elaboración del proyecto, se han invertido unas 320 horas, incluyendo las fases de análisis, diseño, implementación y pruebas, además de la documentación, como se puede apreciar en la Figura 21.

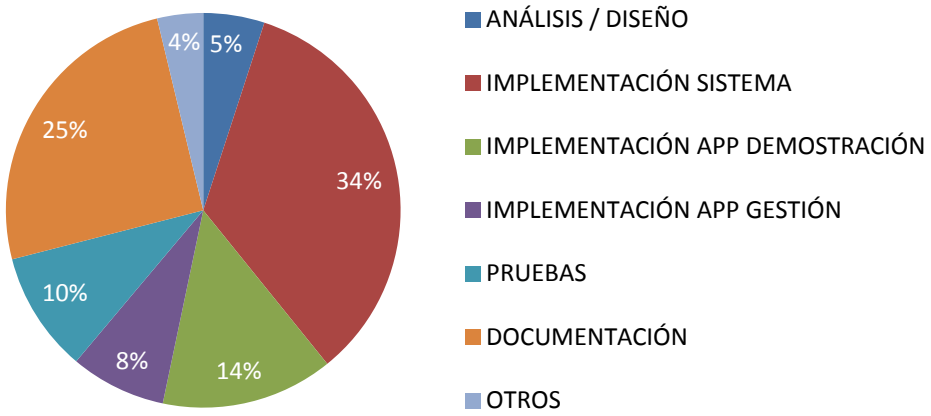

Figura 21. Tiempo dedicado por tarea

3.3. Herramientas de gestión

Entre las herramientas de gestión utilizadas para el proyecto se encuentran las siguientes:

- **GitHub.** Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git que se ha utilizado para almacenar el código del proyecto, así como la gestión global del proyecto mediante *issues* y *milestones*.
- **Dropbox.** Es un servicio de alojamiento de archivos multiplataforma en la nube que permite a los usuarios almacenar y sincronizar archivos en línea. Se ha empleado para compartir y guardar la documentación.
- **Google Drive.** Es un servicio de alojamiento de archivos perteneciente a Google, donde se ha guardado documentación referente al proyecto.
- **GanttProject.** Es una herramienta que facilita la creación de diagramas de Gantt.
- **Microsoft Word.** Es una aplicación informática orientada al procesamiento de textos, empleada para redactar documentación del proyecto.
- **Microsoft Excel.** Es un software de hojas de cálculo dedicado a la redacción de las horas dedicadas en el proyecto.
- **UMLet.** Es una herramienta UML basada en Java que facilita la creación de diagramas para el proyecto.

4. CONCLUSIONES

Una vez finalizado el proyecto, se realizan las valoraciones pertinentes sobre el desarrollo y el resultado final del proyecto, así como las lecciones aprendidas junto con una opinión personal. Por otra parte, se especifican las posibilidades de futuro que tiene el proyecto.

4.1. Resultados

Al término del proyecto, se ha conseguido implementar un sistema de gestión de geofences completamente funcional y que cumple con todas las especificaciones definidas al principio del proyecto, si bien algunos de estos requisitos se han ido concretando con mayor detalle durante el transcurso del proyecto.

Por lo tanto, el sistema final obtenido permite gestionar usuarios, geofences, reglas y notificaciones, es decir, facilita su creación, modificación, obtención y borrado. Todo interesado en utilizar las funcionalidades del sistema tiene que registrarse, con el fin de conseguir una cuenta de usuario que le permita autenticarse y hacer uso de las funcionalidades restringidas presentes en el sistema.

Por último, el sistema también admite el envío de la localización por parte del usuario para que pueda administrarla adecuadamente, comprobando el cumplimiento de las reglas de los geofences sobre los que se encuentre dentro y generando las notificaciones oportunas.

Todo el código generado del sistema y de las aplicaciones web se encuentra accesible en la dirección *https://github.com/IAAA-Lab/Geofences*.

4.2. Lecciones aprendidas

Durante el desarrollo del proyecto se han ido aprendiendo distintas técnicas o lecciones útiles para futuros proyectos, además de las tecnologías empleadas en el mismo. A continuación se van a especificar las principales.

- **Verificar código ajeno.** Durante la búsqueda de código que realice una funcionalidad en concreto o que solvente un error que se está teniendo, es conveniente probar ese código por separado, antes de integrarlo en el código que está en desarrollo, aunque sea una cuestión trivial. De esta manera uno se asegura que el código encontrado cumple correctamente su función y evita que genere nuevos problemas en el proyecto, dado que muchas veces se opta por añadir código sin saber exactamente cómo funciona.

- **Realizar pruebas al finalizar cada funcionalidad.** En cuanto se termine de implementar una funcionalidad en correcto es vital realizar pruebas que cubran y verifiquen que tiene el comportamiento deseado. Estas pruebas también ayudan a encontrar errores que han pasado desapercibidos durante la implementación. No realizar pruebas una vez terminada de codificar la funcionalidad puede acarrear grandes problemas, como retrasos en el proyecto en caso de que contenga errores o *bugs*, que si hubieran sido tratados en el momento en el que se originan, habría ahorrado mucho tiempo y esfuerzo.
- **Documentarse minuciosamente.** En algunos casos, el *framework* Spring facilita la implementación de determinadas tareas, porque tiene módulos predeterminados que se encargan de ello. Sin embargo, en cuanto se quiere modificar el comportamiento por defecto de esos módulos, el trabajo que lleva es bastante considerable. Por ello, a menudo es mejor leerse previamente la documentación, aunque sea muy extensa, con el fin de conocer al detalle la implementación.
- **Utilizar servicios web si se busca interoperabilidad.** Los servicios web facilitan la interoperabilidad entre sistemas, ocupándose específicamente de la lógica. Es decir, la lógica y la capa de *front-end* se encuentran separadas, lo cual reduce la complejidad a la hora de programar. Esta característica se ha aprovechado en el proyecto por parte de las aplicaciones web, que han realizado llamadas al sistema cuando han necesitado utilizar su API.

4.3. Conclusión personal

La realización del proyecto ha sido útil en varios aspectos. En primer lugar, me ha permitido la posibilidad de aprender a organizarme mejor el tiempo en proyectos de una envergadura considerable, como es el caso, dado que tiene un mayor alcance que otros en los que me he embarcado. Además, el hecho de que solo haya un único desarrollador que se encargue de toda la elaboración del proyecto, junto con la ayuda y supervisión del tutor, también me ha aportado experiencia en la toma de decisiones.

El desarrollo del proyecto también me ha permitido conocer nuevas tecnologías como Hibernate o JSON Web Tokens, así como mejorar y especializarme un poco más en el uso del *framework* Spring, tecnología que había empleado con anterioridad, pero de la que había sabido aprovechar las múltiples oportunidades que ofrece a la hora de implementar determinadas tareas como ahora en este proyecto.

También me ha aportado madurez, tenacidad y originalidad a la hora de enfrentarme a problemas complejos y sin una solución trivial, con ha ocurrido durante la autenticación de la comunicación vía WebSockets o la configuración del CORS del sistema para permitir el acceso desde cualquier origen.

Por último, la realización del proyecto me ha enseñado las ventajas de mantener una coordinación constante con el director del proyecto, que gracias a su ayuda y consejo, además de paciencia en algunos casos, ha hecho mucho más llevadera y fructífera esta labor.

4.4. Futuro del proyecto

Este proyecto es un sistema de gestión de geofences con unas funcionalidades limitadas que pueden ser ampliadas en un futuro, aumentando los servicios ofrecidos, si al usuario le surgen unas nuevas necesidades o deseos que el sistema pudiera cubrir.

De igual manera, algunas partes del sistema podrían ser mejoradas si se desarrollan nuevas lógicas, como la referente al cumplimiento de las reglas de los geofences, que efectuaran el mismo cometido, pero de manera optimizada.

En cuanto a cualquier comercialización del producto o similares, se trata de un proyecto de software de código abierto, accesible a todo aquel que quiera utilizarlo, por lo que no se espera sacar beneficio alguno del mismo.

5. BIBLIOGRAFÍA

[1] Headway, «Qué es el Geofencing,» 6 Marzo 2014. [En línea]. Available: http://www.headwaydigital.com/es/que-es-el-geofencing-o-geotargeting-geolocalizacion-en-moviles/.

[2] R. Raya, «¿Qué son las metodologías ágiles?,» 24 Julio 2014. [En línea]. Available: http://blog.leanmonitor.com/es/que-son-las-metodologias-agiles/.

[3] K. Hernández, «Autenticación basada en tokens,» 15 Abril 2015. [En línea]. Available: http://koldohernandez.com/autenticacion-basada-en-tokens/.

[4] Nicanor, «Qué es WebSocket,» 13 Enero 2013. [En línea]. Available: http://queeswebsocket.blogspot.com.es/2013/01/que-es-websocket.html.

[5] S. Tilkov, «Principios de REST,» 27 Noviembre 2008. [En línea]. Available: http://www.dosideas.com/noticias/java/332-principios-de-rest.html.

[6] R. Lee, «Using Spring 4 WebSocket, SockJS and STOMP,» 19 Enero 2014. [En línea]. Available: https://raymondhlee.wordpress.com/2014/01/19/using-spring-4-websocket-sockjs-and-stomp-support-to-implement-two-way-server-client-communication/.

[7] N. Provos y D. Mazières, «A future-adaptable password scheme,» *Proceedings of 1999 USENIX Annual Technical Conference,* pp. 81-92, 1999.

[8] J. Zapata, «Niveles de prueba del software,» 21 Enero 2013 . [En línea]. Available: https://pruebasdelsoftware.wordpress.com/2013/01/21/niveles-de-prueba-del-software/.

[9] Y. Carreno, «Thymeleaf: The natural template,» 10 Agosto 2014. [En línea]. Available: http://www.yaircarreno.com/2014/08/thymeleaf-natural-template.html.

[10] I. Devlin, «Finding your position with Geolocation,» 14 Junio 2011. [En línea]. Available: http://html5doctor.com/finding-your-position-with-geolocation/.

ANEXO A

A.1. Historias de usuario especificadas

A continuación se listan cada una de las historias de usuario que describen las funcionalidades que se desea que contenga el proyecto, agrupadas por usuario, geofence, regla y notificación.

A.1.1. Relativas a los usuarios

- Como usuario, quiero poder crear una cuenta de usuario con mis datos personales en el sistema para gestionar mis geofences con sus reglas y mis notificaciones.
- Como usuario, quiero poder iniciar sesión introduciendo un nick y una contraseña para identificarme en el sistema y acceder a mi cuenta de usuario.
- Como usuario, quiero poder cerrar sesión en el sistema para dejar de tener acceso a mi cuenta de usuario.
- Como usuario, quiero poder modificar una cuenta de usuario para actualizar los datos personales de mi cuenta en cualquier momento.
- Como usuario, quiero poder eliminar mi cuenta de usuario para borrar los datos personales de mi cuenta del sistema.
- Como usuario, quiero poder visualizar mi cuenta de usuario para comprobar los datos personales de mi cuenta en cualquier momento.

A.1.2. Relativas a las geofences

- Como usuario, quiero poder crear un geofence para delimitar en un mapa un espacio determinado que tiene una especial relevancia para mí.
- Como usuario, quiero poder modificar un geofence para variar las propiedades o atributos de un geofence en cualquier momento.
- Como usuario, quiero poder eliminar un geofence para borrar del sistema un geofence que ha dejado de tener relevancia para mí.
- Como usuario, quiero poder visualizar un geofence para comprobar las propiedades o atributos de un geofence.
- Como usuario, quiero poder visualizar los geofences en un listado ordenado por nombre para que sean accesibles de manera ordenada y rápida.
- Como usuario, quiero poder visualizar los geofences en un mapa que se encuentren dentro de un radio previamente establecido para que sean accesibles de manera visual e intuitiva.

A.1.3. Relativas a las reglas

- Como usuario, quiero recibir una notificación cuando entre en un geofence pasados *n* segundos, previamente establecidos, para saber en qué zona entro.
- Como usuario, quiero recibir una notificación cuando salga de un geofence pasados *n* segundos, previamente establecidos, para saber de qué zona salgo.
- Como usuario, quiero recibir una notificación si permanezco *n* segundos más de los previamente establecidos dentro del geofence para saber cuánto tiempo llevo dentro.
- Como usuario, no quiero recibir una notificación si permanezco *n* segundos menos de los previamente establecidos dentro del geofence, para no ser molestado innecesariamente.
- Como usuario, no quiero recibir una notificación si entro en un geofence, pero el geofence está desactivado, para poder regular los tiempos de envío de notificaciones de un geofence.
- Como usuario, no quiero recibir una notificación si salgo de un geofence, pero el geofence está desactivado, para poder regular los tiempos de envío de notificaciones de un geofence.
- Como usuario, no quiero recibir una notificación si entro en un geofence y permanezco *n* segundos más de los previamente establecidos dentro del geofence, pero el geofence está desactivado, para poder regular los tiempos de envío de notificaciones de un geofence.

A.1.4. Relativas a las notificaciones

- Como usuario, quiero poder crear una notificación para comunicar eventos a quienes interactúen con mis geofences.
- Como usuario, quiero poder modificar una notificación para variar el contenido de la misma cuando quiera.
- Como usuario, quiero poder eliminar una notificación para borrar el contenido de una notificación que ya no me interesa mantener.
- Como usuario, quiero poder visualizar una notificación para comprobar las propiedades o atributos de la notificación.
- Como usuario, quiero poder visualizar las notificaciones recibidas en un listado en mi cuenta de usuario para poder volver a verlas en cualquier momento.

ANEXO B

B.1. Casos de uso detallados

En este anexo se van a detallar los casos de uso más destacados de los diagramas expuestos en la memoria, que recogen las principales funcionalidades que recoge el sistema.

Caso de uso	**Crear usuario**
Actores	Usuario autenticado o usuario no autenticado
Referencias	RF1
Precondición	El usuario no debe haberse registrado previamente.
Postcondición	El usuario está registrado en el sistema.

Propósito
Crear un usuario en el sistema.

Resumen
Un usuario es creado en el sistema. Este usuario podrá ser utilizado para autenticarse en el sistema y tendrá asociados geofences y notificaciones.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce los datos necesarios para crear el usuario.	
	2. El sistema almacena en la BD los datos, creando un nuevo usuario.

Curso alternativo
2.1. El sistema ya contiene en la BD algunos datos que deben ser únicos, es decir, que no pueden repetirse en varios usuarios, y devuelve error.
2.2. El sistema no puede guardar por datos en la BD porque no recibe todos los que son necesarios para crear un usuario.

Tabla 7. Caso de uso detallado: Crear usuario

Caso de uso	Autenticar usuario
Actores	Usuario autenticado o usuario no autenticado
Referencias	RF2
Precondición	El usuario debe haberse registrado previamente en el sistema.
Postcondición	El usuario está autenticado en el sistema.

Propósito
Autenticarse en el sistema con el fin de poder ejecutar funcionalidades del sistema que requieren estar identificado.

Resumen
Un usuario se autentica en el sistema si quiere estar identificado en el sistema cuando utilice las funcionalidades del mismo o si las funcionalidades que quiere utilizar lo requieren.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.

Curso alternativo
2.1. El sistema verifica que el nick de usuario y la contraseña no se corresponden con lo almacenado en la BD y devuelve error.
2.2. El sistema no encuentra un usuario con el nick de usuario recibido y, por consiguiente, devuelve error.

Tabla 8. Caso de uso detallado: Autenticar usuario

Caso de uso	Modificar usuario
Actores	Usuario autenticado
Referencias	RF1
Precondición	El usuario debe haberse registrado previamente en el sistema.
Postcondición	El usuario ha modificado los datos de su cuenta de usuario.

Propósito
Modificar los datos de un usuario en el sistema.

Resumen
Los datos de un usuario son modificados en el sistema. Si algunos datos, que tienen que ser únicos, ya se encuentran en la BD, no se efectuarán los cambios.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario introduce los datos que desea modificar de su cuenta de usuario.	
	4. El sistema almacena en la BD los datos enviados por el usuario, actualizando aquellos que difieren de los anteriormente guardados.

Curso alternativo
4.1 El sistema ya contiene en la BD algunos datos que deben ser únicos, es decir, que no pueden repetirse en varios usuarios, por lo que no realiza los cambios y devuelve error.

Tabla 9. Caso de uso detallado: Modificar usuario

Caso de uso	Eliminar usuario
Actores	Usuario autenticado
Referencias	RF1
Precondición	El usuario debe haberse registrado previamente en el sistema.
Postcondición	El usuario está eliminado del sistema.

Propósito
Eliminar un usuario del sistema.

Resumen
Un usuario es borrado del sistema. Este usuario ya no podrá utilizar esa cuenta de usuario para autenticarse y también perderá todos los geofences y notificaciones asociados a ese usuario.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía petición de borrado de usuario adjuntando el nick de usuario.	
	4. El sistema busca el usuario al que le corresponda ese nick, verifica que el nick de usuario y el de la autenticación coinciden y lo borra del sistema, junto con sus geofences y notificaciones.

Curso alternativo
4.1 El sistema busca el usuario al que le corresponda ese nick, verifica que el nick de usuario y el de la autenticación no coinciden, por lo que no efectúa el borrado y devuelve un error.

Tabla 10. Caso de uso detallado: Eliminar usuario

Caso de uso	Obtener usuario
Actores	Usuario autenticado o usuario no autenticado
Referencias	RF1
Precondición	El usuario debe conocer el nick de usuario sobre el que obtener la información.
Postcondición	El usuario obtiene datos de una cuenta de usuario.

Propósito
Obtener información de una cuenta de usuario.

Resumen
El usuario obtiene información de un usuario a partir de su nick de usuario. Si está autenticado recibirá toda la información completa de su cuenta usuario. Si no está autenticado, recibirá información parcial de cualquier cuenta de usuario.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de obtención de información de su cuenta de usuario, identificado por su nick.	
	4. El sistema busca el usuario al que le corresponda ese nick y devuelve la información completa, siempre que el nick enviado y el de la autenticación coincidan.

Curso alternativo
3.1 El usuario envía una petición de obtención de información de cualquier cuenta de usuario sin estar autenticado, recibiendo del sistema información parcial y no sensible de ese usuario.
3.2 El usuario envía una petición de obtención de información de un usuario identificado con un nick no existente en la BD, por lo que da error.
3.3 El usuario envía una petición de obtención de información de una cuenta de usuario que no se corresponde con la que se ha autenticado, recibiendo por consiguiente solo información parcial y no sensible de ese usuario.

Tabla 11. Caso de uso detallado: Obtener usuario

Caso de uso	Crear geofence
Actores	Usuario autenticado
Referencias	RF3
Precondición	El usuario debe haberse registrado previamente en el sistema.
Postcondición	Se ha creado un geofence asociado a un usuario.

Propósito
Crear un geofence asociado a un usuario.

Resumen
El usuario tiene que estar registrado en el sistema para poder crear un geofence, el cual estará asociado a esa cuenta de usuario.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de creación de un geofence, con los datos necesarios.	
	4. El sistema crea un geofence con los datos obtenidos y lo asocia al usuario que ha realizado la petición.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de creación de un geofence con los datos necesarios, recibiendo un error del sistema por la falta de autenticación.

Tabla 12. Caso de uso detallado: Crear geofence

Caso de uso	Modificar geofence
Actores	Usuario autenticado
Referencias	RF3
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence.
Postcondición	Se ha modificado un geofence asociado a un usuario.

Propósito
Modificar un geofence asociado a un usuario.

Resumen
El usuario tiene que estar registrado en el sistema para poder modificar los datos de un geofence, previamente creado, el cual estará asociado a esa cuenta de usuario.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de modificación de un geofence, con los datos necesarios.	
	4. El sistema modifica un geofence existente con los datos obtenidos.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de modificación de un geofence con los datos necesarios, recibiendo un error del sistema por la falta de autenticación.
3.2 El usuario autenticado envía la petición de modificación de un geofence no existente en la BD, recibiendo un error por parte del sistema.

Tabla 13. Caso de uso detallado: Modificar geofence

Caso de uso	Eliminar geofence
Actores	Usuario autenticado
Referencias	RF3
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence.
Postcondición	Se ha eliminado un geofence asociado a un usuario.

Propósito
Eliminar un geofence asociado a un usuario.

Resumen
El usuario tiene que estar registrado en el sistema para poder eliminar un geofence, previamente creado, el cual estará asociado a esa cuenta de usuario.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de borrado de un geofence concreto.	
	4. El sistema elimina un geofence existente de la BD.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de borrado de un geofence, recibiendo un error del sistema por la falta de autenticación.
3.2 El usuario autenticado envía la petición de borrado de un geofence no existente en la BD, recibiendo un error por parte del sistema.

Tabla 14. Caso de uso detallado: Eliminar geofence

Caso de uso	Obtener geofences
Actores	Usuario autenticado o usuario no autenticado
Referencias	RF3
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence.
Postcondición	Se ha obtenido uno o varios geofences asociados a un usuario.

Propósito
Obtener uno o varios geofences asociados a un usuario.

Resumen
El usuario obtiene la información completa sobre uno o varios geofences si está autenticado e información parcial si no lo está.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de obtención de información de uno o varios geofences.	
	4. El sistema devuelve la información completa de uno o varios geofences, dependiendo de la petición realizada.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de obtención de uno o varios geofences, recibiendo información parcial de los mismos.
3.2 El usuario autenticado en el sistema envía una petición de obtenicion de un geofence determinado no existente en la BD, recibiendo un error.

Tabla 15. Caso de uso detallado: Obtener geofences

Caso de uso	Crear regla
Actores	Usuario autenticado
Referencias	RF4
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence.
Postcondición	Se ha creado una regla asociada a un geofence previamente creado.

Propósito
Crear una regla asociada a un geofence.

Resumen
El usuario crea una regla asociada a un geofence. Este usuario tiene que estar autenticado en el sistema para poder hacerlo.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de creación de una regla, con los datos pertinentes.	
	4. El sistema ha creado una regla asociada a un geofence existente.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de creación de una regla, recibiendo un error por parte del sistema por falta de autenticación.
3.2 El usuario autenticado en el sistema envía una petición de creación de una regla asociada a un geofence que no existe en la BD, por lo que devuelve un error.

Tabla 16. Caso de uso detallado: Crear regla

Caso de uso	Modificar regla
Actores	Usuario autenticado
Referencias	RF4
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence y una regla.
Postcondición	Se ha modificado una regla asociada a un geofence.

Propósito
Modificar una regla asociada a un geofence.

Resumen
El usuario tiene que estar registrado en el sistema para poder modificar los datos de una regla, así como haber creado previamente la regla a modificar y el geofence al que está asociado la regla.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de modificación de una regla, con los datos necesarios.	
	4. El sistema modifica la regla existente con los datos obtenidos.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de modificación de una regla con los datos necesarios, recibiendo un error del sistema por la falta de autenticación.
3.2 El usuario autenticado envía la petición de modificación de una regla no existente en la BD, recibiendo un error por parte del sistema.

Tabla 17. Caso de uso detallado: Modificar regla

Caso de uso	Eliminar regla
Actores	Usuario autenticado
Referencias	RF4
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence y una regla.
Postcondición	Se ha eliminado una regla asociada a un geofence.

Propósito
Eliminar una regla asociada a un geofence.

Resumen
El usuario tiene que estar registrado en el sistema para poder eliminar una regla, previamente creada, la cual estará asociada a un geofence.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de borrado de una regla concreta.	
	4. El sistema elimina la regla del geofence al que está asociado.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de borrado de una regla, recibiendo un error del sistema por la falta de autenticación.
3.2 El usuario autenticado envía la petición de borrado de una regla no existente en la BD, recibiendo un error por parte del sistema.

Tabla 18. Caso de uso detallado: Eliminar regla

Caso de uso	Obtener regla
Actores	Usuario autenticado o usuario no autenticado
Referencias	RF4
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence y una regla.
Postcondición	Se ha obtenido una regla asociada a un geofence.

Propósito
Obtener una regla asociado a un geofence.

Resumen
El usuario obtiene la información completa de una regla si está autenticado e información parcial si no lo está.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de obtención de información de una regla determinada.	
	4. El sistema devuelve la información completa de una regla concreta.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de obtención de una regla determinada, recibiendo información parcial de ella.
3.2 El usuario autenticado en el sistema envía una petición de obtención de una regla determinada no existente en la BD, recibiendo un error.

Tabla 19. Caso de uso detallado: Obtener regla

Caso de uso	Crear notificación
Actores	Usuario autenticado
Referencias	RF5
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence y una regla.
Postcondición	Se ha creado una notificación asociada a una regla previamente creada.

Propósito
Crear una notificación asociada a una regla.

Resumen
El usuario crea una notificación asociada a una regla. Este usuario tiene que estar autenticado en el sistema para poder hacerlo.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de creación de una notificación, con los datos necesarios.	
	4. El sistema ha creado una notificación asociada a una regla existente.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de creación de una notificación, recibiendo un error por parte del sistema por falta de autenticación.
3.2 El usuario autenticado en el sistema envía una petición de creación de una notificación asociada a una regla que no existe en la BD, por lo que devuelve un error.

Tabla 20. Caso de uso detallado: Crear notificación

Caso de uso	Modificar notificación
Actores	Usuario autenticado
Referencias	RF5
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence, una regla y una notificación.
Postcondición	Se ha modificado una notificación asociada a una regla.

Propósito
Modificar una notificación asociada a una regla.

Resumen
El usuario tiene que estar registrado en el sistema para poder modificar las notificaciones, así como haber creado previamente la notificación a modificar y la regla a la que está asociada la notificación.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de modificación de una notificación, con los datos necesarios.	
	4. El sistema modifica la notificación existente con los datos obtenidos.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de modificación de una notificación con los datos necesarios, recibiendo un error del sistema por la falta de autenticación.
3.2 El usuario autenticado envía la petición de modificación de una notificación no existente en la BD, recibiendo un error por parte del sistema.

Tabla 21. Caso de uso detallado: Modificar notificación

Caso de uso	Eliminar notificación
Actores	Usuario autenticado
Referencias	RF5
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence, una regla y una notificación,
Postcondición	Se ha eliminado una notificación asociada a una regla.

Propósito
Eliminar una notificación asociada a una regla.

Resumen
El usuario tiene que estar registrado en el sistema para poder eliminar una notificación, previamente creada, la cual estará asociada a una regla.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de borrado de una notificación en específico.	
	4. El sistema elimina la notificación de la regla a la que está asociada.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de borrado de una notificación, recibiendo un error del sistema por la falta de autenticación.
3.2 El usuario autenticado envía la petición de borrado de una notificación no existente en la BD, recibiendo un error por parte del sistema.

Tabla 22. Caso de uso detallado: Eliminar notificación

Caso de uso	Obtener notificaciones
Actores	Usuario autenticado o usuario no autenticado
Referencias	RF5
Precondición	El usuario debe haberse registrado previamente en el sistema y creado un geofence, una regla y una notificación.
Postcondición	Se ha obtenido una notificación asociada a una regla.

Propósito
Obtener una notificación asociada a una regla.

Resumen
El usuario obtiene la información completa de una o varias notificaciones si está autenticado e información parcial si no lo está.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía una petición de obtención de información de una o varias notificaciones.	
	4. El sistema devuelve la información completa de una o varias notificaciones.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía una petición de obtención de una notificación determinada, recibiendo información parcial de ella.
3.2 El usuario autenticado en el sistema envía una petición de obtención de una notificación determinada no existente en la BD, recibiendo un error.

Tabla 23. Caso de uso detallado: Obtener notificaciones

Caso de uso	Enviar posición
Actores	Usuario autenticado
Referencias	RF6
Precondición	El usuario debe haberse registrado previamente en el sistema.
Postcondición	Se ha enviado la posición al sistema.

Propósito
Enviar la posición del usuario al sistema.

Resumen
El usuario envía la posición en la que encuentra al sistema, para que la gestione. Para poder hacerlo tiene que estar autenticado en el sistema.

Curso normal	
Acciones del actor	*Respuesta del sistema*
1. El usuario introduce su nick de usuario y contraseña en la petición de autenticación que hace al sistema.	
	2. El sistema comprueba que el nick de usuario y contraseña coinciden con los almacenados en la BD y devuelve el token de autenticación.
3. El usuario envía su posición al sistema.	
	4. El sistema gestiona y compara la posición con respecto a los geofences y le devuelve la posición recibida.

Curso alternativo
3.1 El usuario no autenticado en el sistema envía su posición, recibiendo un error por parte del sistema.

Tabla 24. Caso de uso detallado: Enviar posición

ANEXO C

C.1. Relaciones entre requisitos y casos de uso

A continuación se van a exponer las relaciones existentes entre los requisitos funcionales y los casos de uso del sistema, poniendo de manifiesto los vínculos de dependencia que hay entre ellos.

- El requisito funcional 1, que se refiere a la creación, modificación, obtención y borrado de usuarios, está vinculado al caso de uso “crear usuario”, “modificar usuario”, “eliminar usuario” y “obtener usuario”.
- El requisito funcional 2, que se refiere al inicio de sesión, está vinculado al caso de uso “autenticar usuario”.
- El requisito funcional 3, que se refiere a la creación, modificación, obtención y borrado de geofences, está vinculado al caso de uso “crear geofence”, “modificar geofence”, “eliminar geofence” y “obtener geofences”.
- El requisito funcional 4, que se refiere a la creación, modificación, obtención y borrado de reglas, está vinculado al caso de uso “crear regla”, “modificar regla”, “eliminar regla” y “obtener regla”.
- El requisito funcional 5, que se refiere a la creación, modificación, obtención y borrado de notificaciones, está vinculado al caso de uso “crear notificación”, “modificar notificación”, “eliminar notificación” y “obtener notificaciones”.
- El requisito funcional 6, que se refiere al envío de la posición para que el usuario compruebe el cumplimiento de las reglas, está relacionado con el caso de uso “enviar posición”.
- El requisito funcional 7, que se refiere a definir una regla de tipo entrada o de tipo salida, está relacionado con el caso de uso “crear regla” y “modificar regla”.
- El requisito funcional 8, que se refiere a definir una regla de tipo permanencia, está relacionado con el caso de uso “crear regla” y “modificar regla”.
- El requisito funcional 9, que se refiere a definir un tiempo de espera a una regla, está relacionado con el caso de uso “crear regla” y “modificar regla”.

ANEXO D

D.1. Estructura de entidades en JSON

En este anexo se especifica la estructura en formato JSON de un Usuario, en la Figura 22, de una Regla, en la Figura 23, y de una Notificación, en la Figura 24. Los datos que se muestran son aleatorios, pero válidos.


```
{
    "id" : 1,
    "nick" : "example.gmail.com",
    "password" : "$2a$10$FiigyLASJZICbTAaLZ2JCu3Cv7lPc6y2Zz25rVctpUoQoA92H81/G",
    "birthday" : "1992-08-07",
    "imei" : "356938035643809",
    "geofences" : [],
    "enabled" : true,
    "role" : "ROLE_USER",
    "notifications" : [],
    "firstName" : "First",
    "lastName" : "Last",
    "lastPasswordResetDate" : 1463616000000,
    "geofencesRegistry" : []
}
```


Figura 22. Usuario en formato JSON


```
{
    "id" : 3,
    "enabled" : true,
    "type" : "INSIDE",
    "time" : 10,
    "message" : "You are inside",
    "days" : [],
    "notifications" : [],
    "geofence" : {
        "id" : 2,
        "type" : "Feature",
        "properties" : null,
        "geometry" : {
            "type" : "Point",
            "coordinates" : [1.0, 2.0]
        },
        "user" : {
            "id" : 1,
            "nick" : "example.gmail.com",
            "password" : "$2a$10$FiigyLASJZICbTAaLZ2JCu3Cv7lPc6y2Zz25rVctpUoQoA92H81/G",
            "birthday" : "1992-08-07",
            "imei" : "356938035643809",
            "geofences" : [],
            "enabled" : true,
            "role" : "ROLE_USER",
            "notifications" : [],
            "firstName" : "First",
            "lastName" : "Last",
            "lastPasswordResetDate" : 1463616000000,
            "geofencesRegistry" : []
        },
        "rules" : []
    }
}
```


Figura 23. Regla en formato JSON

```
{
    "id" : 4,
    "rule" : {
        "id" : 3,
        "enabled" : true,
        "type" : "INSIDE",
        "time" : 10,
        "message" : "You are inside",
        "days" : [],
        "notifications" : [],
        "geofence" : {
            "id" : 2,
            "type" : "Feature",
            "properties" : null,
            "geometry" : {
                "type" : "Point",
                "coordinates" : [1.0, 2.0]
            },
            "user" : {
                "id" : 1,
                "nick" : "example.gmail.com",
                "password" : "$2a$10$FiigyLASJZICbTAaLZ2JCu3Cv7lPc6y2Zz25rVctpUoQoA92H81/G"
                "birthday" : "1992-08-07",
                "imei" : "356938035643809",
                "geofences" : [],
                "enabled" : true,
                "role" : "ROLE_USER",
                "notifications" : [],
                "firstName" : "First",
                "lastName" : "Last",
                "lastPasswordResetDate" : 1463616000000,
                "geofencesRegistry" : []
            },
        "rules" : []
        }
    },
    "user" : {
        "id" : 1,
        "nick" : "example.gmail.com",
        "password" : "$2a$10$FiigyLASJZICbTAaLZ2JCu3Cv7lPc6y2Zz25rVctpUoQoA92H81/G",
        "birthday" : "1992-08-07",
        "imei" : "356938035643809",
        "geofences" : [],
        "enabled" : true,
        "role" : "ROLE_USER",
        "notifications" : [],
        "firstName" : "First",
        "lastName" : "Last",
        "lastPasswordResetDate"  : 1463616000000,
        "geofencesRegistry" : []
    },
    "status" : "No leído",
    "date" : "2016-01-19"
}
```


Figura 24. Notificación en formato JSON