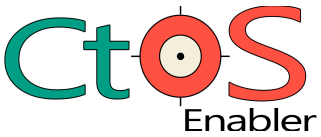
CtOS Enabler



Rodrigo Crespo Cepeda
Meriem El Yamri El Khatibi
Juan Manuel Carrera García

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

Directora: **Eva Ullán Hernández**

Co-director: **José Luis Vázquez-Poletti**

Curso 2014 / 2015

Autorización

Los abajo firmantes, Meriem El Yamri El Khatibi, Juan Manuel Carrera García y Rodrigo Crespo Cepeda, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la presente memoria: "CtOS Enabler", como los contenidos audiovisuales y la documentación desarrollados durante el curso académico 2014-2015 bajo la dirección de los profesores Eva Ullán Hernández y José Luis Vázquez-Poletti en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Meriem El Yamri El Khatibi Rodrigo Crespo Cepeda Juan Manuel Carrera García

Madrid, 19 de Junio de 2015

A Eva y José Luis, que sin su constante apoyo este proyecto no hubiera llegado tan lejos

Agradecimientos

A todos nuestros familiares y amigos que nos han apoyado desde el principio, asistiendo a los eventos en que hemos participado, y votando en lo que les pedimos que votasen.

En especial, gracias a José Luis Vázquez-Poletti y a Eva Ullán Hernández por ayudarnos tanto con este trabajo como por adoptar roles más allá de lo que se les requería para apoyarnos en todas las aventuras en las que hemos inscrito CtOS Enabler a lo largo de este año.

Gracias también a Emprende UCM por ayudarnos a ver desde una perspectiva más "profana" y menos técnica a nuestro proyecto, y por torturarnos para que lleguemos a entender lo que conlleva crear un plan de empresa.

Nuestro agradecimiento a Borja Manero Iglesias, por ayudarnos a perder el miedo al escenario.

Finalmente, gracias a Pedro Antonio González Calero y a Daniel Mozos Muñoz, Decano de la Facultad de Informática de la Universidad Complutense, por redactar esas maravillosas cartas de apoyo a nuestro trabajo dándole un empujón más hacia el triunfo.

Índice

Índice de figuras

Resumen

Las Smart Cities son, indudablemente, el futuro próximo de la tecnología al que nos acercamos cada día, lo que se puede observar en la abundancia de dispositivos móviles entre la población, que informatizan la vida cotidiana mediante el uso de la geolocalización y la información. Pretendemos unir estos dos ámbitos con CtOS Enabler para crear un estándar de uso que englobe todos los sistemas de Smart Cities y facilite a los desarrolladores de dicho software la creación de nuevas herramientas.

CtOS proporciona una capa de baja complejidad que transforma la interacción entre la información y el posicionamiento en una tarea sencilla, habilitando de esta manera la posibilidad de creación de un producto de calidad y de éxito. El sistema se centra en capacitar al usuario para definir zonas de trabajo en las cuales guardar información que interactúa con sus dispositivos cuando se encuentran dentro de dichas zonas, permitiendo realizar tareas basadas en la localización en tiempo real.

Además, como valor añadido, se proporcionan datos estadísticos del uso del servicio que ayudan a mejorar el producto del cliente. CtOS Enabler está orientado para desarrolladores y empresas que quieran usar el servicio u ofrecerlo a sus propios clientes, convirtiéndose ellos mismos en intermediarios. El modelo de pago se centra en varios tipos de suscripción, de gran flexibilidad, según las funcionalidades requeridas.

CtOS Enabler ayuda a mejorar la capacidad geolocalizadora de cualquier software, sea un chat zonal, una aplicación de parking inteligente, o algo más complejo como un sistema de control de semáforos y bolardos para emergencias de los cuerpos de seguridad y salvamento.

Palabras clave: API, Cloud Computing, Framework, Geolocalización, IaaS, PaaS, SaaS

Abstract

Smart Cities are, undoubtedly, the near future of technology we approach every day, which can be observed in the profusion of mobile devices among the population, these devices automatize daily life through the use of geolocation and information. Two areas that we intend to unite with CtOS Enabler to create a standard of use that encloses every Smart Cities system and eases the creation of new tools to developers of that kind of software.

CtOS is a framework which provides a low-complexity layer that transforms the interaction between information and positioning in a simple task, thus enabling the possibility of creating a high quality and successful product, making geolocated data useful. The complete functionality resides in the Cloud, allowing its resources to be easily scalable and accessible. The system focuses on qualifying the user to define regions of work where to store information that interacts with their devices when they are within those zones, allowing to perform location-based real time tasks.

Also, as an added value, statistical data of service usage is provided to help improve the customer's product. CtOS Enabler is geared for both developers, who can use it as a Platform as a Service (PaaS), or companies, Infrastructure as a Service (IaaS) providers, willing to use the API or acting as intermediaries to provide the service to their own customers, focusing on various highly flexible subscription models, according to the functionality required. CtOS Enabler helps to improve the geolocation capacity of any software, such as location limited chats, a smart parking app, or something more complex like a smart city control system.

Keywords: API, Cloud Computing, Framework, Geolocation, IaaS, PaaS, SaaS

Capítulo 1. Introducción

El mundo de la tecnología evoluciona rápida y ruidosamente.

Con la aparición de los smartphones y la geolocalización se ha producido un repentino y necesario crecimiento en el desarrollo de estos sistemas, con el objetivo de construir una sociedad más conectada e informatizada.

El proyecto CtOS Enabler consiste en la creación de un estándar de uso en la geolocalización que pueda ser utilizado tanto por desarrolladores, como por empresas que dispongan de apps de geolocalización, que saque el máximo partido a los datos geoposicionados y la información generada de forma geolocalizada, y haga uso del Cloud Computing tanto para desplegar el servicio como para ofrecerlo a terceros.

Para alcanzar este fin, se ha creado una Interfaz de Programación de Aplicaciones (API) de geolocalización, junto con una serie de plugins en diferentes lenguajes de programación para realizar las peticiones. Además, se han añadido un conjunto de herramientas visuales para poder crear y gestionar zonas de influencia sobre un mapa, sin necesidad de tener un perfil tecnológico ni conocimientos de programación e interacción con APIs, y poder visualizar los datos estadísticos obtenidos a partir de información generada por el geoposicionamiento.

Chapter 1. Introduction

The technological world evolves fast and loudly.

The apparition of smartphones and geolocation has produced a sudden and necessary growth in the development of these systems, with the goal of building a more connected and computerized society.

CtOS Enabler project was developed to create a standard of geolocation that can be used either by developers or companies that have apps which use geolocation. This project is intended to take maximum advantage of the geolocated data and all the information that this data generates, and to make use of Cloud Computing to deploy the service and offer it to third parties.

To that end, a geolocation Application Programming Interface (API) has been created, along with several plugins in different programming languages to handle requests to this API. Also, a set of visualization tools have been added to create and manage influence areas on the map, without the need of a technical profile or any programming knowledge, and allow to visualize statistical use data generated by geolocation.

Capítulo 2. ¿Por qué CtOS Enabler?

Motivación

Como desarrolladores vimos el problema que supone construir desde cero la parte de geolocalización de las aplicaciones, así como la gran cantidad de datos geoposicionados que las apps generan hoy en día y que están siendo desaprovechados, a pesar de que éstos pueden ser de gran utilidad para las empresas.

Objetivos

- Crear una API de geolocalización para agilizar el desarrollo de aplicaciones que hagan uso de ella.
- Aprovechar al máximo todos los datos geoposicionados que se generan y no están siendo utilizados actualmente.
- Brindar a empresas que hagan uso de CtOS Enabler ventajas competitivas utilizando los datos geoposicionados generados por sus apps respecto a sus clientes, transformándolos en datos que sean amigables y de los que puedan sacar provecho viendo la línea que sigue su negocio, llegando incluso a reorientarlo.

Entorno de CtOS Enabler

Como consecuencia del crecimiento del uso de dispositivos móviles en los últimos años, y al hecho de que cada vez vivimos más interconectados, se ha producido un aumento en el uso de tecnologías en la nube, especialmente en el uso de Plataformas como Servicio.

Este crecimiento excepcional, cuya tendencia es continuar al alza, ha promovido el desarrollo de aplicaciones software que utilizan como base los servicios disponibles en la Nube como medio de inclusión de funcionalidades.

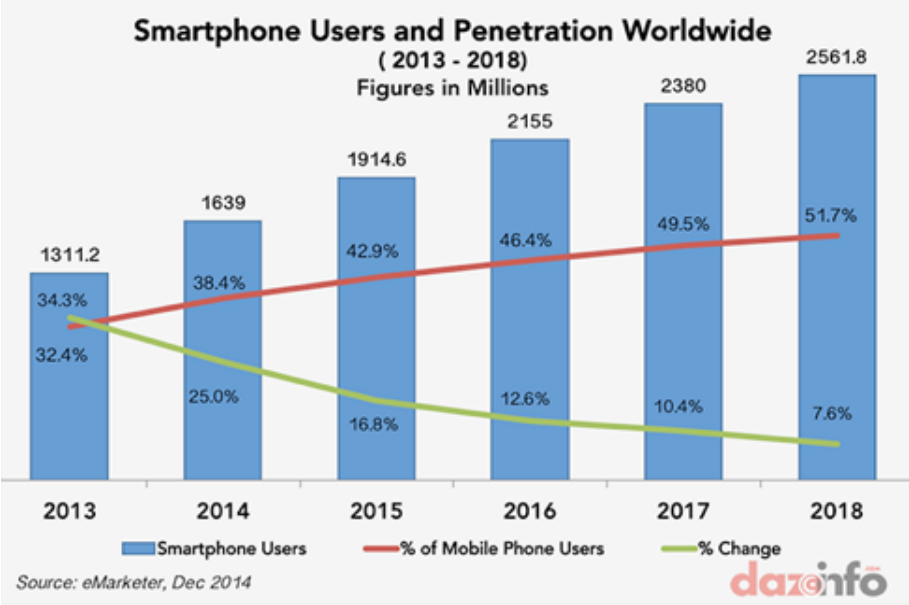


Figura 2.1. Estimación del uso de Smartphones

Además, el aumento del número de smartphones que disponen de herramientas de localización por coordenadas, como GPS y localización por triangulación de antenas mediante la conexión a internet, hacen que una gran cantidad de aplicaciones utilicen sistemas de geolocalización para ofrecer nuevos contenidos y crear experiencias de usuario más reales y cercanas a éstos. Mediante los datos generados por el geoposicionamiento los creadores pueden extraer estadísticas de utilidad para sus propios negocios.

El sector Big Data es otro en el que se encuentra CtOS Enabler y también está en pleno crecimiento. Las técnicas de minería de datos o *data mining*, serán usadas en un futuro en CtOS Enabler para recoger y procesar todos los datos geoposicionados generados por los usuarios de las aplicaciones que usen el servicio, y a partir de ellos, generar estadísticas para los propietarios de las aplicaciones, los clientes del servicio.

Esta estimación está basada en una publicación de la agencia de noticias Reuters que, a partir de los datos estadísticos obtenidos por NASSCOM (The National Association of Software and Service Companies) y CRISIL (Credit Rating Information Services of India Limited), estima que el mercado **Big Data crecerá un 45% cada año**, alcanzando los **25 billones de dólares en 2015 y con un consumo creciente de datos que en 2020 llegará los 35 Zettabytes**[1].

[1] 1 Zettabyte = 1.000.000.000 Terabytes

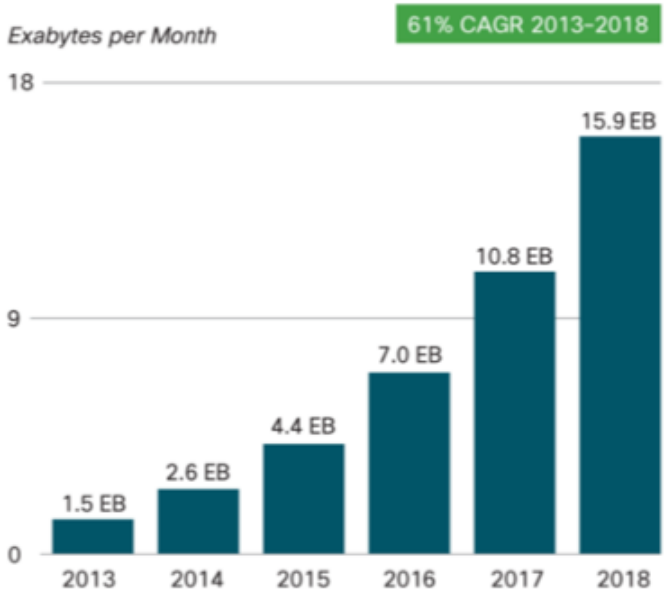


Figura 2.2. Estimación del uso de datos móviles

Por lo tanto, al ser un sistema centrado en un mercado con una previsión de crecimiento tan favorable, CtOS Enabler se presenta como una herramienta de gran utilidad para aprovechar dicho crecimiento, indispensable para el uso de estas tecnologías en los próximos años, y situada en un sector con muy buenas previsiones y en un mercado de gran potencial y en continuo crecimiento.

Capítulo 3. Estado del arte

En este capítulo presentamos el estudio de mercado realizado antes del inicio del proyecto, momento en el cual se buscaron soluciones semejantes para estudiarlas y evaluarlas de cara a obtener ideas para la realización de este trabajo.

Concluimos que no existe ningún producto en el mercado que realice exactamente las mismas funciones que CtOS Enabler. Sin embargo, existen varias empresas que proveen servicios que incluyen algunas de las funcionalidades que ofrece CtOS Enabler, y son las siguientes:

CartoDB

CartoDB es una empresa española que presenta un sistema parecido a CtOS Enabler, aunque su entorno y su destino no son los mismos. CartoDB se identifica como una plataforma online útil para la generación de mapas personalizados diseñados para ser visualizados en páginas web. Ofrece una API para gestionar los datos geoposicionados desde aplicaciones, pero su objetivo no es la creación de aplicaciones móviles que usen geolocalización, ni la simplificación del tratamiento de estos datos y obtención de estadísticas, sino que se limita a permitir la creación de mapas personalizados muy atractivos a nivel visual, cosa que CtOS Enabler no contempla tener en una primera versión.

Dirección web (URL): http://cartodb.com

PlotAPI

Plot API es un negocio con base en Ámsterdam, cuyo objetivo es el denominado **geofencing**, que consiste en enviar notificaciones a dispositivos móviles cuando éstos entran en una zona específica y generar estadísticas al respecto. Ofrece una interfaz gráfica para visualizar las estadísticas y las localizaciones definidas. En este sentido, cubre una de las funcionalidades que proporciona CtOS Enabler y está también orientado a las apps móviles. Sin embargo, la capacidad de CtOS Enabler es mucho más atractiva para la creación de aplicaciones complejas, al permitir

guardar grandes cantidades de datos en la Nube y realizar muchas más funcionalidades e interacciones entre esos datos.

Dirección web (URL): http://www.plotprojects.com/geofencing

ESRI Geotrigger

Esri Geotrigger es una empresa con sede en California que, entre sus numerosos servicios, proporciona uno que es parecido a CtOS Enabler en muchos aspectos. Podríamos considerarlo un competidor directo, porque su objetivo específico para este servicio tiene mucho en común con CtOS Enabler. Permite la creación de notificaciones push a dispositivos móviles cuando entran en una determinada área, la personalización de mapas y el cálculo de rutas y estadísticas.

Dirección web (URL): https://developers.arcgis.com/en/features/geotrigger-service

Smartme Analytics

Smartme Analytics es una empresa española que busca el mismo objetivo que CtOS Enabler para su cliente: la obtención de datos estadísticos de consumidores. La orientación de su negocio lo diferencia de CtOS Enabler, ya que proporciona una aplicación móvil que los clientes se deben descargar para obtener ventajas. Por el contrario, CtOS Enabler permite ser integrado en cualquier aplicación y los datos son de valor para el cliente que lo use en su app y no para CtOS Enabler, que de lo que se encarga es de almacenarlos y, en un futuro, de generar las estadísticas de interés para el cliente, el cual almacenará los datos geoposicionadas parametrizados y podrá visualizar estadísticas en función de los parámetros.

Dirección web (URL): http://www.smartmeanalytics.com/

Capítulo 4. Descripción

Descripción general

CtOS Enabler es un servicio en la nube que permite al usuario añadir datos geolocalizados de su interés para ser utilizados por sus aplicaciones, y se encarga de gestionar los datos geoposicionados generados por dichas aplicaciones.

Las ventajas que proporciona el servicio son diferentes en función del tipo de cliente que haga uso de él:

- Para **desarrolladores de aplicaciones que usen geolocalización**, CtOS Enabler ofrece un paquete de fácil integración en su sistema que les permitirá realizar un desarrollo más rápido y con un gran abanico de funcionalidades de geolocalización, permitiendo añadir valor a su producto. Además, el sistema es portable a cualquier plataforma y lenguaje de programación, facilitando la escalabilidad de los sistemas sin necesidad de reprogramar y la interacción entre distintos dispositivos.
- Para **empresas que tengan una aplicación que utilice geolocalización**, CtOS Enabler funciona como un sistema que les permitirá reorientar su negocio mediante la obtención de datos de uso de su aplicación por parte de sus usuarios. Además, dispondrán de un panel gráfico sencillo para gestionar sus zonas de influencia y los datos geoposicionados en ellas, sin necesidad de un perfil técnico.

Descripción técnica

Todo el sistema se basa en la gestión de **zonas de influencia**, entendiendo zona de influencia como un área geográfica que el administrador de la aplicación considera de interés, ya sea porque quiere insertar datos en esa zona o porque quiere explotar los datos de los usuarios de su aplicación que accedan a dicha zona. Las zonas de influencia están definidas como figuras geométricas situadas en el mundo real mediante el uso de coordenadas o direcciones postales.

El uso de CtOS Enabler varía según el tipo de cliente. Para **negocios**, se pueden destacar las siguientes funcionalidades, articuladas en torno al concepto de zona de influencia y manejadas desde un panel de control simple y sencillo:

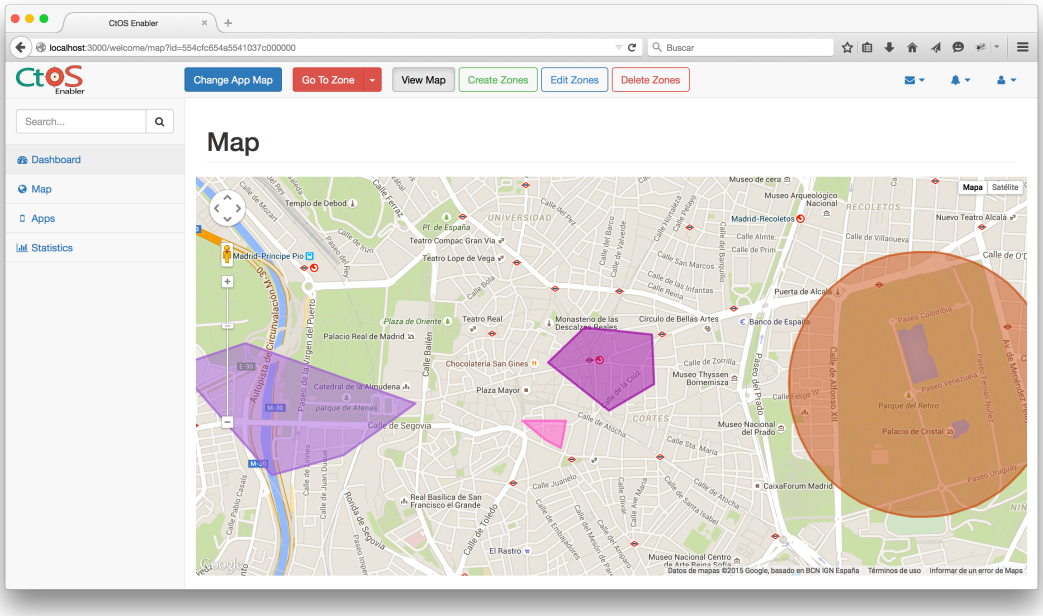


Figura 4.1. Panel de control CtOS Enabler

- **Creación de zonas:** el administrador podrá definir zonas de influencia en las que actúa el sistema. Las zonas pueden estar relacionadas entre sí, contener a otras o solaparse, favoreciendo la versatilidad al delimitar un territorio de acción.
- **Respuesta de zona:** se podrá personalizar la información asociada a una zona para que interactúe con los usuarios de una aplicación cuando entran en dicha zona.
- **Interacción con dispositivos en zona:** el administrador podrá visualizar en tiempo real los dispositivos que se localicen dentro de una determinada zona de influencia para actuar sobre ellos por separado o en conjunto. Por ejemplo, se podrá interactuar con el conjunto de clientes que están dentro de la zona de influencia de una tienda para que reciban una promoción en sus dispositivos móviles.
- **Estadísticas de uso:** disponibilidad de datos de uso de la aplicación por zonas, tipo y cantidad de usuarios, utilización del servicio en el tiempo, etcétera, permitiendo obtener información útil y aprovechable para adaptar o mejorar su negocio y poder generar ventajas competitivas.

Para **desarrolladores**, el sistema se presenta como un conjunto de herramientas en la nube que les proporciona las siguientes funcionalidades:

- Inclusión de un paquete de sencilla integración y fácil configuración que disminuye la carga de trabajo al no tener que realizar la gestión de la geolocalización ni las funcionalidades que CtOS Enabler proporciona al cliente empresa, con la consecuente reducción del tiempo total de desarrollo.
- Portabilidad a cualquier plataforma y lenguaje de programación, posibilitando la inclusión de CtOS Enabler en cualquier dispositivo y aplicación, además de la interacción entre ellos.
- Uso del sistema de gestión de zonas de influencia, con su inclusión en el desarrollo de la aplicación, que permite la interacción con datos geoposicionados personalizables. Con ello, se facilita la creación de sistemas complejos que repercuten en una mejora para la experiencia del usuario final y producen un valor añadido para el cliente empresa.

Capítulo 5. Arquitectura

En este capítulo presentamos una descripción más detallada de la arquitectura del sistema y su diseño.

Diseño y especificación

Diseño de clases

Se ha utilizado el patrón Model-View-Controller (MVC) para la estructuración del sistema, ya que es un método efectivo para la separación de conceptos y facilita la reutilización del código y la ampliación de funcionalidades.

La estructura general de este patrón, divide la API en tres niveles:

- View, que se corresponde con el panel gráfica que da soporte a la API.
- Model, que se corresponde con la estructura de clases que da forma a la API.
- Controller, que gestiona la lógica entre las peticiones a la API y el modelo, y el front-end del panel gráfico.

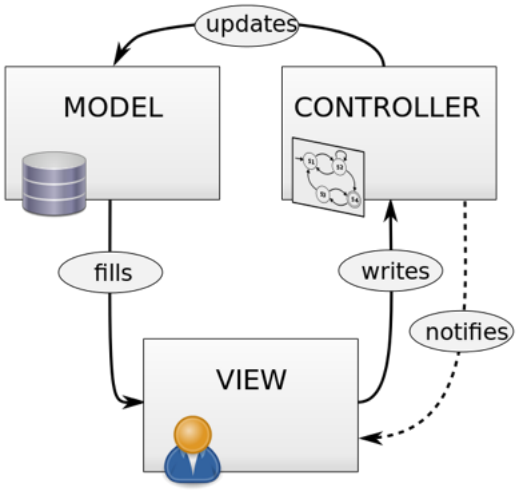


Figura 5.1. Patrón de diseño MVC

Modelo

El modelo está formado por las siguientes clases:

- **User**: El principal actor del sistema, pues es necesario el registro de un usuario para utilizar el framework, incluyendo la API.

  Tiene un código secreto que le permite realizar las distintas operaciones además de crear Apps. Contiene un sistema de autenticación que usa contraseñas encriptadas y operaciones de recuperación en caso de olvido.

- **App**: Representa una aplicación real que utiliza el framework. Se utiliza como un sistema para englobar todas las zonas e información relacionadas entre sí por una misma aplicación.

  Pertenece a un usuario y puede contener muchas zonas. También tiene un código secreto para realizar todas las operaciones ligadas a la propia aplicación.

- **Zone**: Zona de influencia que será utilizada por el sistema como elemento básico de interacción geolocalizado. Se compone de una forma geométrica, poligonal o circular, definida por un elemento GeoJSON, el cual a la vez especifica la posición en coordenadas del elemento en el mundo real.

  Puede ser definida como un "objeto", entendiendo como objeto una zona que no puede contener zonas dentro, ya que es el elemento más pequeño utilizable con forma de zona, como, por ejemplo, una estatua o un árbol.
  Tiene una "clase de zona" que representa agrupaciones de zonas (por ejemplo, monumento, jugador), aunque este elemento no es obligatorio.
  Además contiene un DataInfo, donde guarda la información de zona.

- **DataInfo**: Se trata del elemento en el que se guarda la información que va a ser usada por las aplicaciones para dar sentido a una zona. Puede estar formada por un simple número, una frase, o estructuras más complejas como elementos JSON o XML.

  Es un elemento embebido en una zona, por lo que no puede existir sin ella. Se ha creado como un elemento separado para permitir la existencia de elementos hijos, que hereden de DataInfo, y que especifiquen estructuras complejas de datos, para mejorar la experiencia de usuario. Por ejemplo, un tipo de DataInfo para definir a una persona, podría contener campos para su nombre, apellidos, edad, etc. y, gracias al

uso de bases de datos NoSQL, no tendría ningún inconveniente para su guardado y su uso.

- **QueryHistory:** Es el modelo que representa la información que se guarda cada vez que se realiza una consulta a la API. Esta clase pertenece a cada app y almacena el nombre de la consulta, la app a la que referencia, la fecha y la hora de la realización de la consulta, y el dispositivo que ha realizado dicha consulta.

  La información del dispositivo puede contener aquellos parámetros que al desarrollador le interese almacenar porque sean susceptibles de proporcionar estadísticas que al gestor de la aplicación puedan resultar útiles.

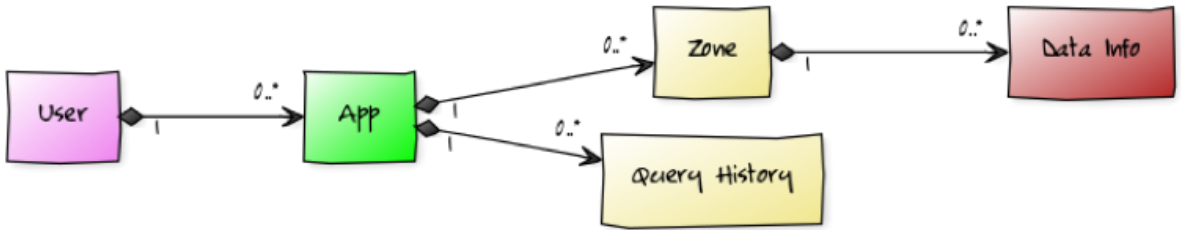


Figura 5.2. Diagrama de clases del modelo

Controlador

El controlador está formado por:

- **Controlador base:** Controlador por defecto, del que heredan los demás, y que contiene elementos comunes para los demás controladores.
- **Controlador de panel gráfico:** Controlador para la interacción con el panel gráfico y las vistas que le corresponden.
- **Controlador base global de API:** Permite establecer elementos comunes para los controladores de las distintas versiones de la API.
- **Controlador de User:** Se compone de varios controladores para las funciones específicas de usuario:
  - Controlador de **password**: Creación, modificación y recuperación de contraseñas.
  - Controlador de **registro**: Creación, modificación y borrado de cuentas de usuario.
  - Controlador de **sesión**: Conexión, desconexión y autenticación.

- **Controlador de API (v1):**
  - Controlador **base**: Contiene la funcionalidad base para la API en la versión v1.
  - Controlador **de App**: Funcionalidad de llamadas a la API para operaciones sobre apps. Antes de realizar cualquier operación se autentica el código de usuario y se verifican los permisos.
- **Controlador de Zone:** Funcionalidad de llamadas a la API para operaciones sobre zonas. Antes de realizar cualquier método se autentica el código de usuario y de app, y se verifican los permisos y la pertenencia de una app a un usuario.

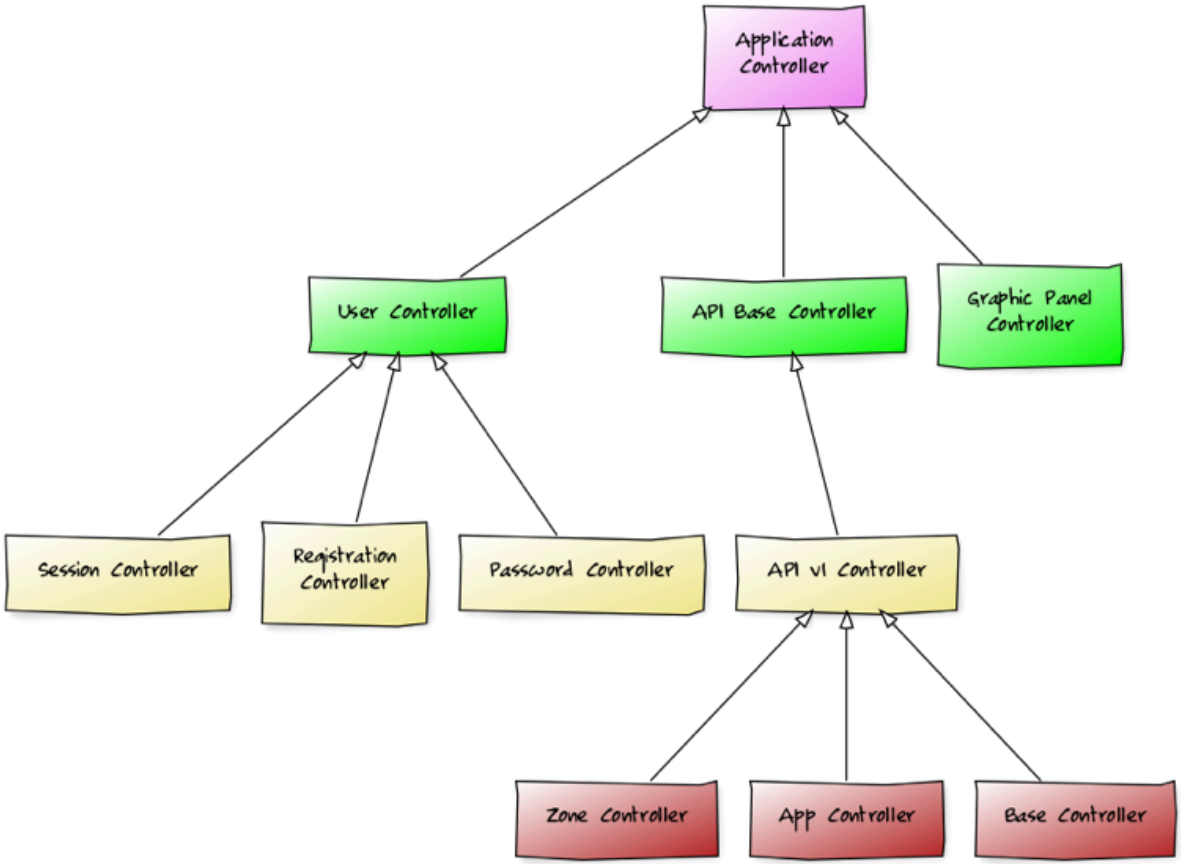


Figura 5.3. Diagrama de clases de los controladores

Existen unos módulos, llamados **helpers**, que realizan funciones de ayuda para los controladores y los modelos:

- **Helper de respuesta:** Engloba un sistema de estructuración de respuestas para que las de la API sean semejantes y estén organizadas.
- **Helper de API:** Elementos de ayuda a la API como, por ejemplo, autenticación de tokens.

- **Helper de ElasticSearch:** Funciones para el tratamiento de datos devueltos por Elasticsearch, transformación de resultados y funciones de búsqueda.
- **Helper de geometría:** Funciones relacionadas con el tratamiento de figuras geométricas y la relación entre ellas.

Vista

En la parte de la vista se encuentra el **panel de control** de la API. Un panel que se compone de las siguientes páginas: login, dashboard, apps, map y statistics. Los detalles y características del panel de control se explicarán con más detalle en la **sección Panel de Control** de este capítulo.

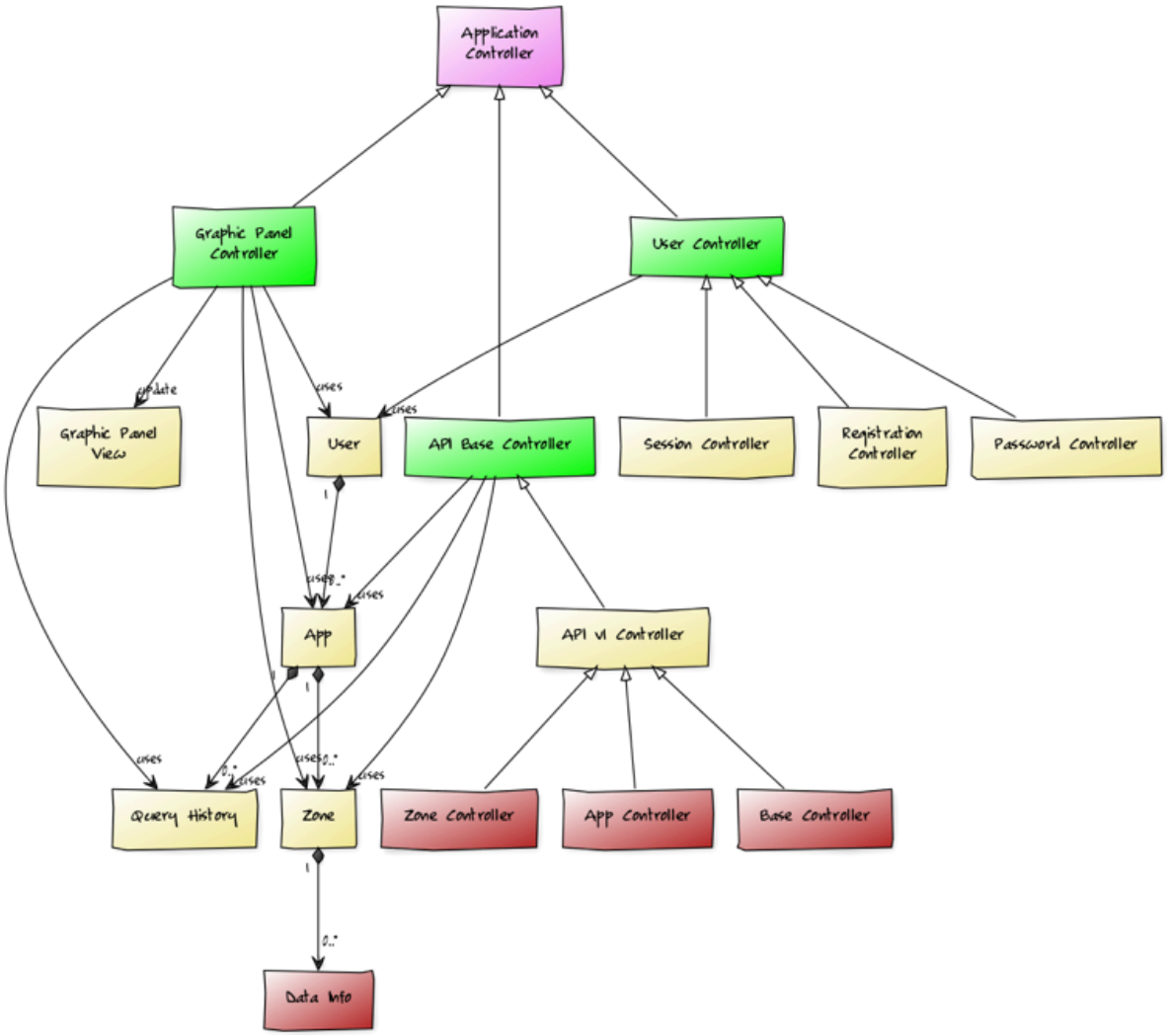


Figura 5.4. Diagrama de clases CtOS Enabler

Diseño de base de datos

La base de datos es dinámica, al tratarse de una base de datos NoSQL, lo que permite la creación de documentos de tamaño variable, con campos distintos, y también la inclusión de unos documentos en otros. La base de datos de CtOS Enabler se modela con el siguiente esquema de Entidad-Relación:

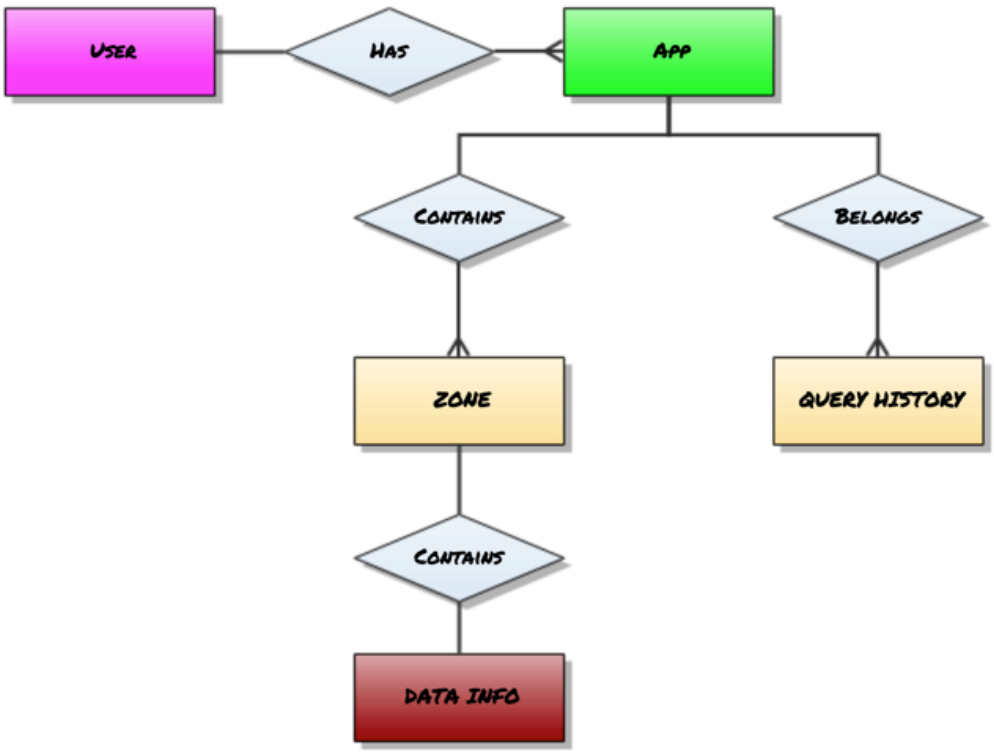


Figura 5.5. Esquema Entidad-Relación CtOS Enabler

Peticiones a la API

Para hacer las peticiones a la API se realiza la consulta sobre una determinada dirección URL, en función de la acción que el usuario desee realizar o de los datos que quiera obtener. Como formato para las consultas se utiliza JSON, tanto para los datos enviados como para los datos devueltos por la API.

Funciones

Según su cometido, podemos distinguir tres tipos de funciones: funciones que realizan operaciones con aplicaciones, funciones que realizan operaciones con zonas de influencia y consultas con coordenadas dentro del sistema terrestre.

Funciones para las aplicaciones

En este grupo se incluyen las funciones para crear, modificar y eliminar aplicaciones. Estas funciones se apoyan totalmente sobre el estándar HTTP y se implementan basándose en el concepto de API REST.

Al implementarse como API REST, estas funciones utilizan una única URI sobre la que se hacen las peticiones para las acciones de **creación, modificación y eliminación**, que es ***/api/v1/app***, y para especificar la acción a realizar utilizamos los siguientes métodos HTTP: **POST** para **crear** una nueva aplicación, **PUT** para **editar/actualizar** una aplicación existente y **DELETE** para **eliminar** una aplicación.

En el manual del desarrollador se detallan tanto el formato como los objetos JSON necesarios para cada una de estas acciones.

Funciones para las zonas de influencia y coordenadas

En este grupo se incluyen las funciones para **crear, editar y eliminar zonas de influencia** y para **consultar la posición o cercanía** de una zona de influencia respecto a otra, o respecto de una coordenada. Se realizan utilizando el **método POST de HTTP** y, en función de la acción a realizar, las consultas se hacen a una determinada dirección URL.

Consultas con coordenadas o direcciones en el sistema terrestre

En las funciones para realizar consultas con coordenadas se incluyen las que nos devuelven una **lista con las zonas que contienen una determinada coordenada** fijada por el usuario, o las que nos dicen si una **coordenada se encuentra dentro de una zona de influencia** determinada. Se realizan utilizando el **método POST de HTTP** y, en función de la petición, las consultas se hacen a una determinada dirección URL.

Llamadas a la API

Esquema de funcionamiento

En la siguiente imagen podemos ver el esquema de funcionamiento de una aplicación cliente realizando consultas a CtOS Enabler.

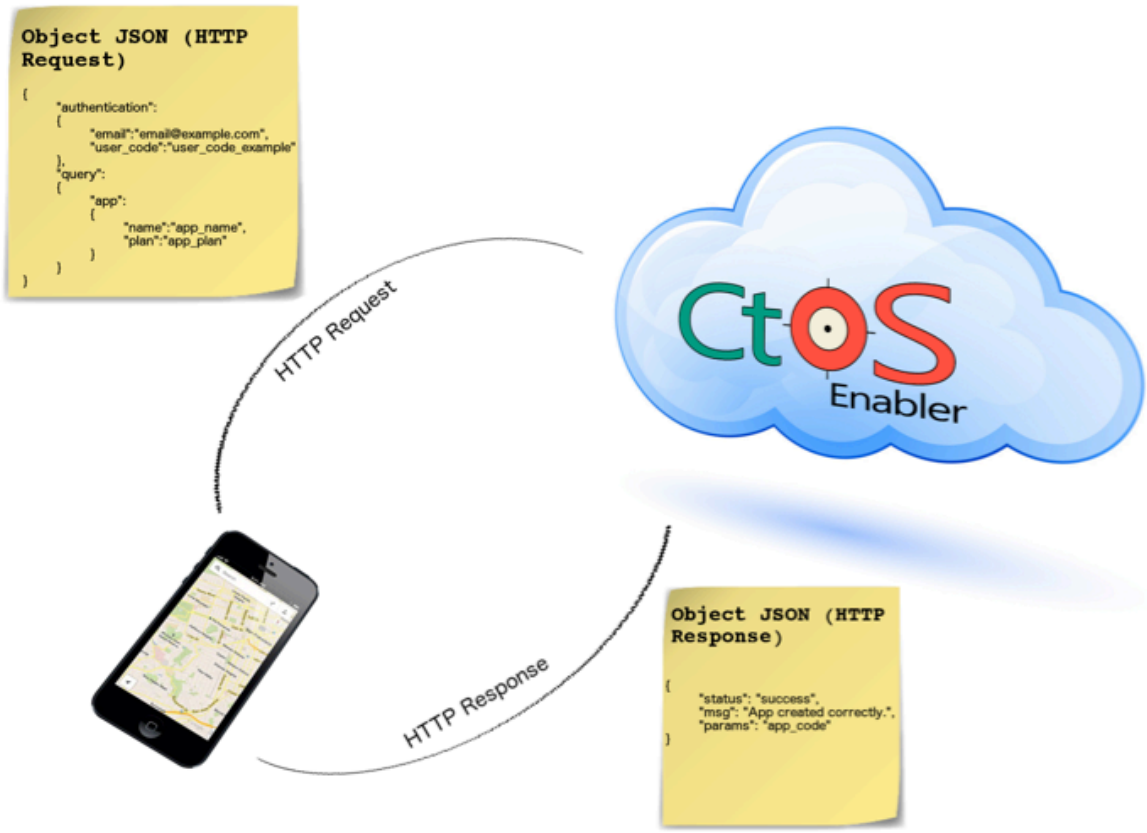


Figura 5.6. Esquema petición a la API

Las peticiones a la API se realizan enviando los datos a una determinada URL, que varía en función del tipo de consulta o acción a realizar, mediante el protocolo **HTTP**. La secuencia de acción es la siguiente:

1. Se genera el objeto JSON con los datos necesarios para la consulta a realizar.
2. Se establece el método HTTP Request (POST, GET, PUT...) y la URL sobre la que se hace la consulta.
3. Se envía un HTTP Request con los datos creados en los puntos 1. y 2.
4. La petición llega a la API.
5. La API genera el objeto JSON con los datos de la respuesta.

6. Se envía un HTTP Response con el JSON generado en el punto 5.

Respuestas de la API

Las respuestas enviadas por la API, en formato JSON, están compuestas por un objeto **response** con tres campos:

- :status. Especifica el estado de la petición. Si se ha realizado correctamente su valor es "success" y si, por el contrario, se ha producido algún error su valor es "error".
- :msg. Contiene un string con un mensaje explicando el resultado de la petición realizada a la API.
- :params. En caso de que :status sea "success", contiene un objeto JSON con los parámetros de salida necesarios en función del método llamado. Por ejemplo, si se realiza la consulta para saber las zonas de una aplicación, :params contendrá un objeto con las zonas de dicha aplicación en formato JSON. Y en caso que sea "error" :params será vacío.

Dentro de las respuestas de la API, podemos diferenciar dos grandes grupos, en función de si la respuesta es satisfactoria o, por el contrario, ha ocurrido algún error.

OK Responses

Las respuestas satisfactorias se dan cuando la petición a la API se realiza de forma correcta y no se produce ningún tipo de error. Por lo tanto, el campo **:status** del objeto JSON devuelto será **"success"**.

En la siguiente tabla se muestra el valor de los campos :msg y :params para las distintas respuestas según la función de la API a la que se llama.

	:msg	:params
ok_zone_created	“Zone created correctly.”	Nombre de la zona
ok_app_created	“App created correctly.”	Código de la app
ok_zone_removed	"Zone removed correctly."	Nombre de la zona
ok_app_removed	"App removed correctly."	Código de la app
ok_zone_updated	"Zone updated correctly."	Nombre de la zona
ok_app_updated	"App updated correctly."	Código de la app
ok_get_all_zones	"All zones listed correctly."	Datos de las zonas dentro de una aplicación
ok_container_zones	"Container zones listed correctly."	Datos de las zonas que contienen a una dada
ok_within_zones	"Zones within listed correctly."	Datos de las zonas contenidas en una dada
ok_near_zones	"Near zones listed correctly."	Datos de las zonas cercanas a una coordenada en un radio determinado
ok_within_zone	"Successful query"	Booleano para saber si una zona está dentro de otra

Figura 5.7. OK Responses de la API

ERROR Responses

Las respuestas erróneas se dan cuando en la petición a la API se produce algún tipo de error (por ejemplo, de autenticación o en la base de datos). En este tipo de respuestas el campo **:status** del objeto JSON devuelto será **“error”**, el campo :**params** será vacío y **:msg** contendrá un string explicando el tipo de error.

En la siguiente tabla se muestra el valor del campo :msg en función del tipo de error:

	:msg
error_authentication	"Error: Authentication is not valid"
error_zone_exists	"Error: Zone already exists"
error_database	"Error: Database error ocurred"
error_zone_not_found	"Error: Zone not found"
error_app_not_found	"Error: App not found"

Figura 5.8. ERROR Responses de la API

Panel de control

En esta sección explicamos el desarrollo del panel de control, junto con las partes que lo componen y las tecnologías utilizadas para ello.

El panel de control ocupa la parte de la **vista** en el patrón MVC utilizado para desarrollar la API. Se trata de un panel gráfico destinado a personas con un perfil menos técnico para que no necesiten realizar las llamadas a la API como tal, y se efectúen a través de dicho panel. En su desarrollo hemos utilizado los siguientes lenguajes:

- Haml como lenguaje de marcado en sustitución del HTML. Haml al final es traducido a HTML pero su sintaxis es mucho más clara y rápida de implementar.
- CSS y Sass como lenguajes de hojas de estilo. Sass es un metalenguaje de hojas de estilo que es traducido a CSS, pero que nos brinda ventajas en el momento del desarrollo.
- JavaScript y jQuery como lenguajes de programación en el lado del cliente.

Las páginas que componen el panel de control son las siguientes:

- **Login:** Es la página situada antes de entrar al panel de control. Da acceso al usuario y prepara el panel con los datos de las aplicaciones, zonas de influencia y estadísticas de dicho usuario.

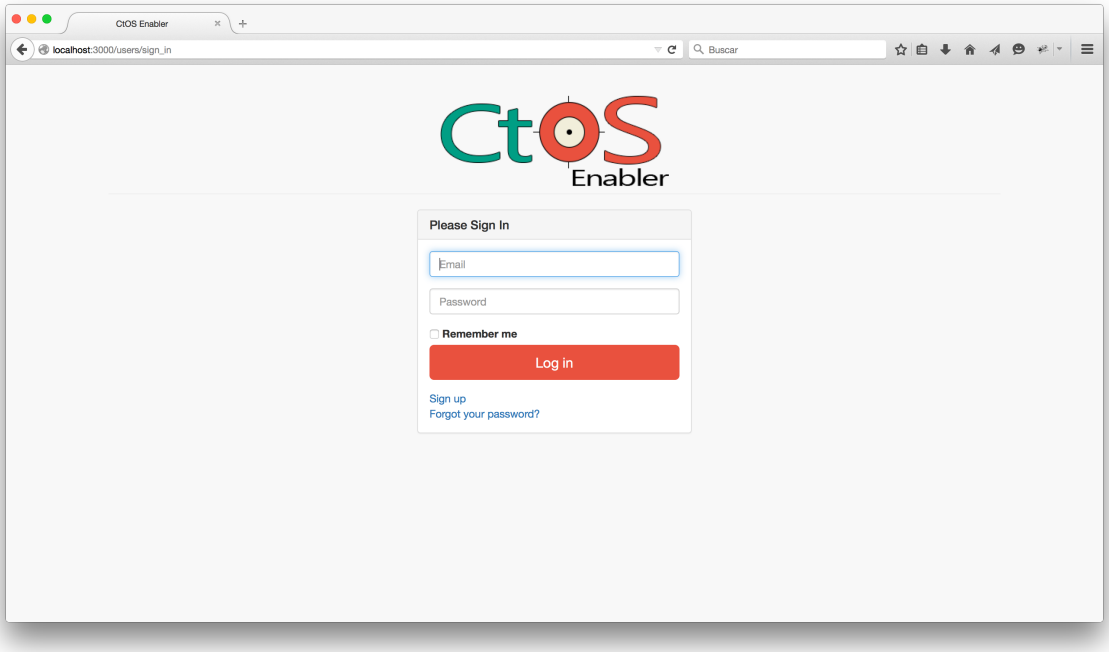


Figura 5.9. Login

- **Dashboard**: En el dashboard podemos encontrar una tabla con las aplicaciones que tenemos creados con nuestra cuenta de usuario, junto con la descripción de las características de cada una de ellas.

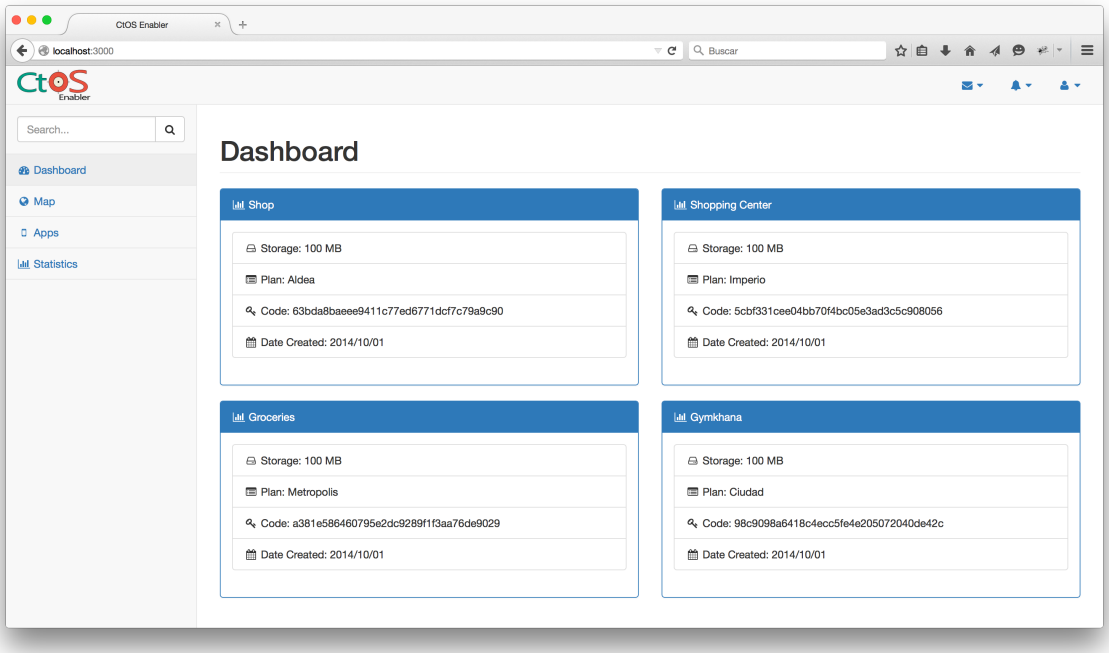


Figura 5.10. Dashboard

- **Apps:** La página de apps es en la que los usuarios pueden crear, editar y eliminar sus aplicaciones y, también, consultar los datos de las que tenían previamente creadas.

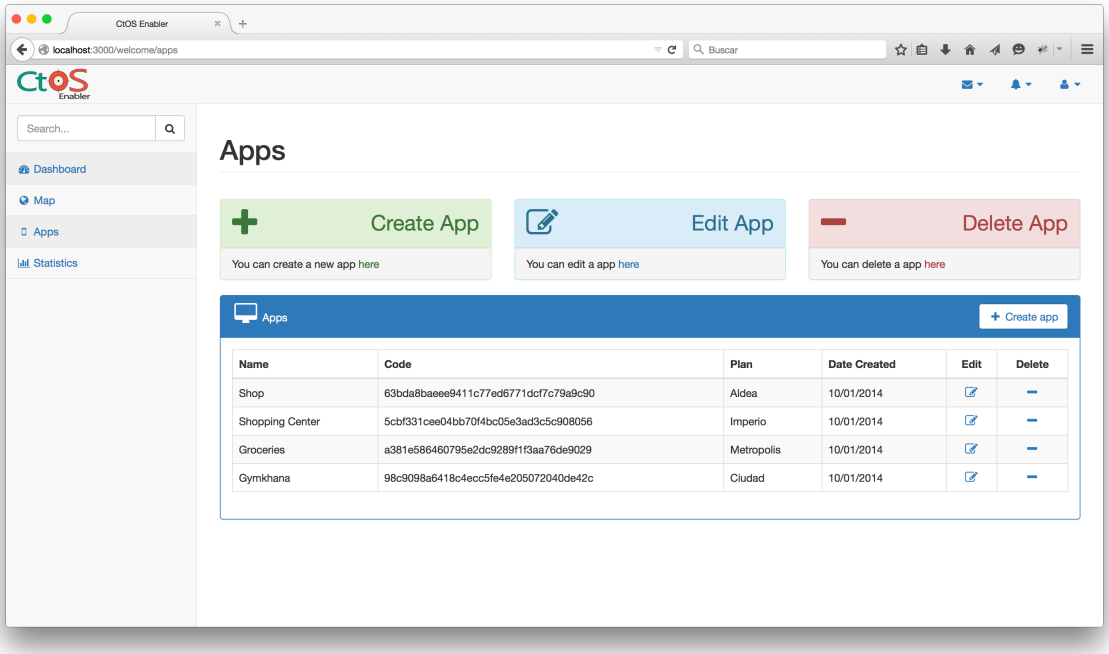


Figura 5.11. Apps

- **Map:** La página de Map se compone de dos partes, la primera de las cuales genera un pop-up con las aplicaciones disponibles en las que se podrán añadir zonas de influencia y editarlas.

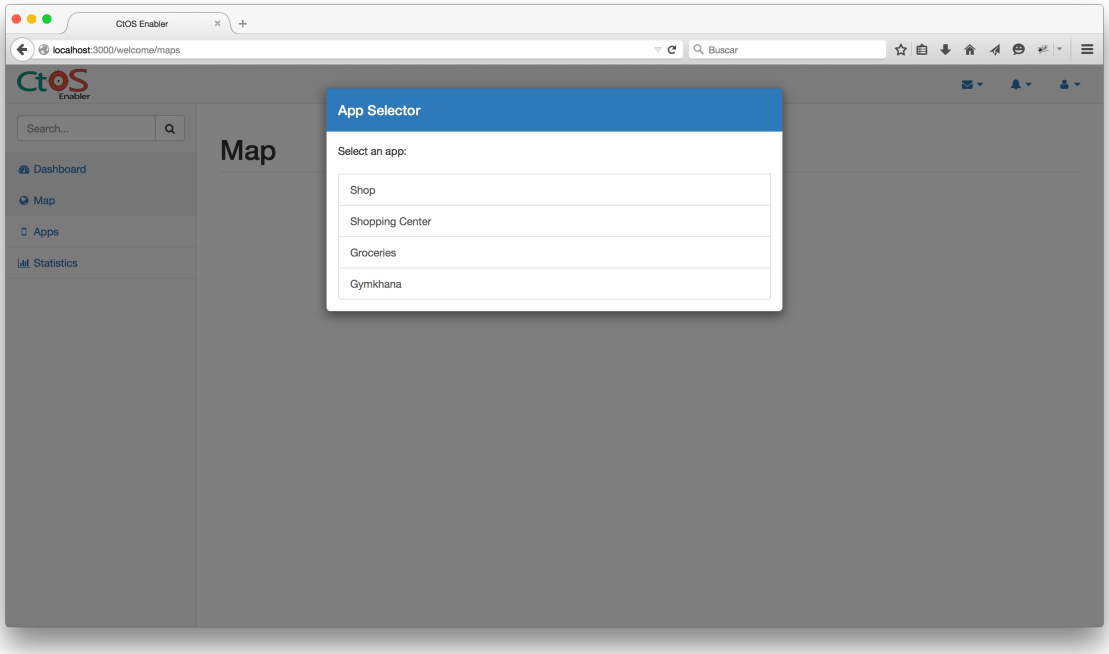


Figura 5.12. Selector de App

La segunda, teniendo la aplicación seleccionada, ofrece un mapa interactivo en el que gestionar las zonas de influencia relativas a dicha app.

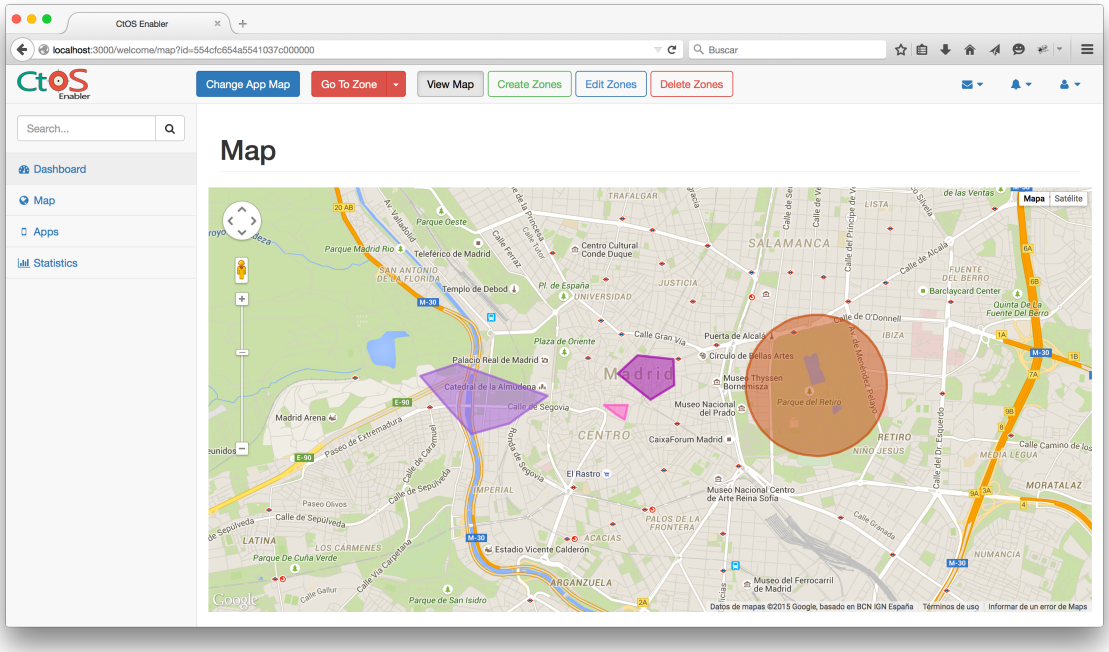


Figura 5.13. Mapa de aplicación

- **Statistics[2]:** Contendrá estadísticas generadas a partir de los datos geoposicionados de los usuarios que utilicen la aplicación que integra CtOS Enabler. Los datos se presentarán en forma de gráficas o mapas interactivos.

El manejo del panel de control, así como las diferentes funcionalidades que ofrece y la ejecución de éstas, es explicado en el **Anexo I. Manual del Administrador**.

Web

Se ha creado una *landing-page* con el objetivo de poder acceder al panel de administración y poder obtener más información sobre el proyecto como método de captación de clientes. Además se explican los diferentes proyectos creados utilizando CtOS Enabler y los posibles planes de suscripción mensual. La url de esta página es www.ctosenabler.com.

Figura 5.14. Portada

[2] La parte de estadísticas se trata de un trabajo futuro pero ya se han añadido librerías en el panel de control para generar las gráficas con los *datasets*.

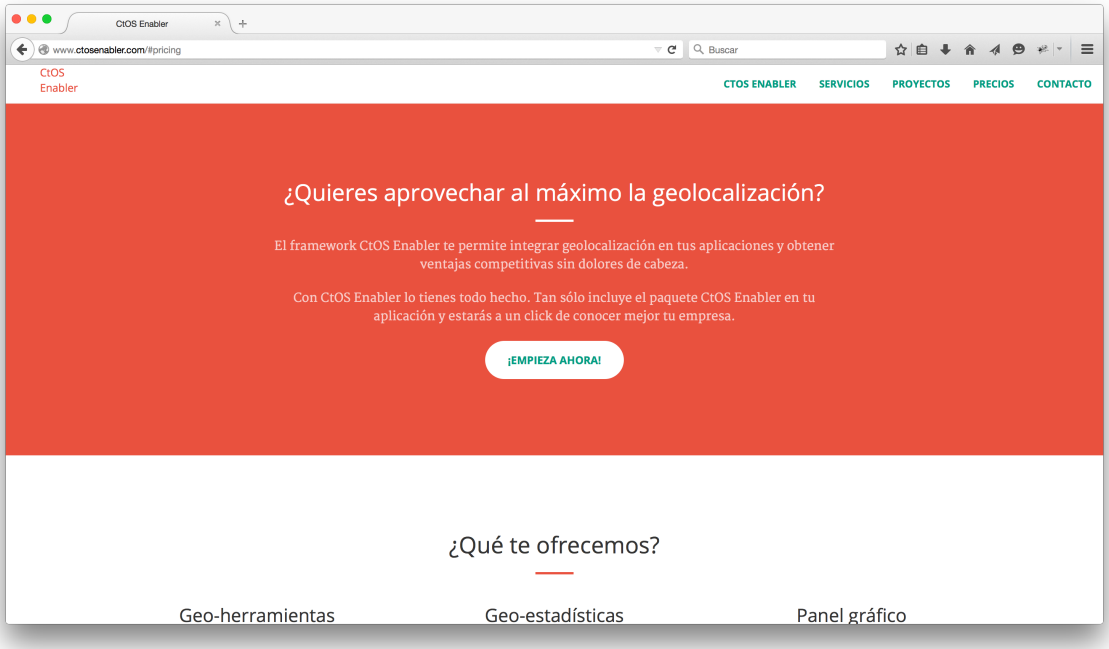


Figura 5.15. Descripción

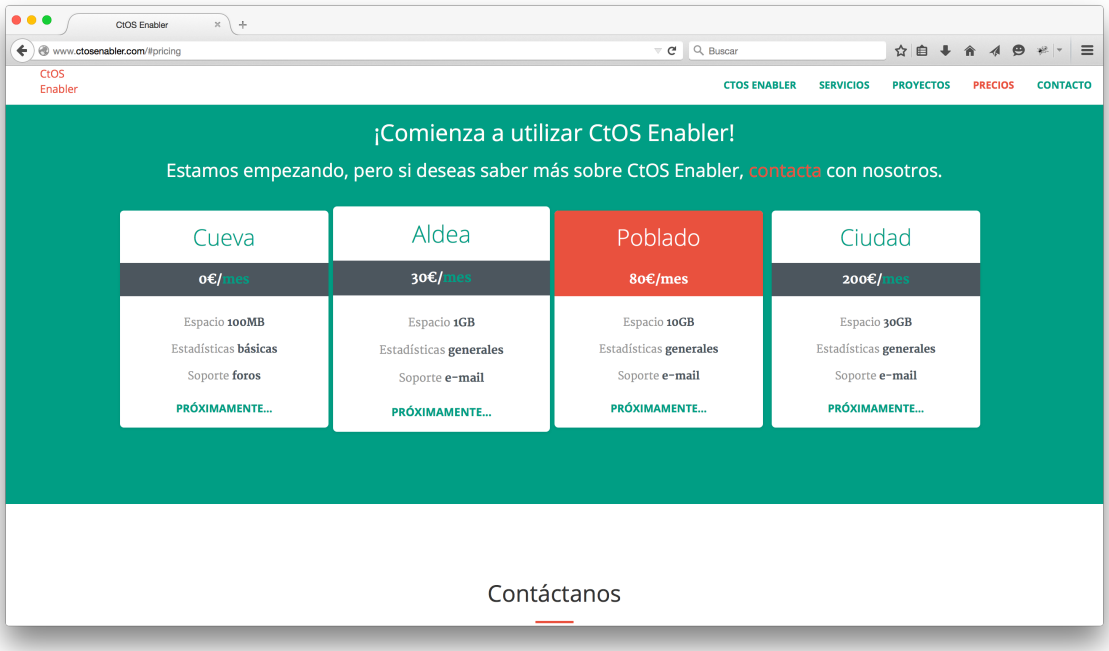


Figura 5.16. Precios

Capítulo 6. Tecnologías

Antes de comenzar a desarrollar el framework, hicimos un exhaustivo estudio de las tecnologías más usadas para realizar APIs y aplicaciones web, tras el cual nos decantamos por las que detallamos a continuación.

Tecnologías utilizadas

Ruby (Ruby on Rails): Como lenguaje de desarrollo de la API se ha optado por Ruby, enmarcado en el framework de Ruby on Rails. Rails es un framework de desarrollo de aplicaciones web escrito en el lenguaje de programación Ruby. Está diseñado para hacer que la programación de aplicaciones web sea más fácil, haciendo supuestos sobre lo que cada desarrollador necesita para comenzar. Permite escribir menos código realizando más que muchos otros lenguajes y frameworks.

Rails es un software dogmático. Éste asume que existe una forma "mejor" de hacer las cosas, y está diseñado para fomentar esa forma - y en algunos casos para desalentar alternativas.

La filosofía de Rails se basa en estos dos principios:

- DRY (del inglés, *"Don't Repeat Yourself"*) - sugiere que escribir el mismo código una y otra vez es una mala práctica.
- "Convención sobre Configuración" - significa que Rails hace algunas suposiciones sobre lo que el programador quiere hacer y cómo va a hacerlo, en lugar de requerir que especifique cada pequeña cosa a través de archivos de configuración.

La herramienta de desarrollo o IDE que se ha usado es **RubyMine**, un entorno de desarrollo pensado para trabajar con Rails y desarrollado por JetBrains. Se ha escogido este IDE ya que proporciona autocompletado de código, refactorización y análisis de código en tiempo real, lo cual economiza el tiempo de desarrollo en gran medida.

Javascript: El lenguaje JavaScript se ha usado esencialmente para el panel gráfico de CtOS Enabler, y para desarrollar las partes de interacción con el cliente. Se ha utilizado la API de Google Maps para Javascript de cara a permitir la creación de zonas sobre el mapa y su interacción con éste. Las llamadas a la API se han implementado con funciones AJAX.

CSS: Para los estilos del panel de control se ha partido del proyecto SB-Admin2 como base, y se han ido haciendo modificaciones para permitir que el panel ofrezca las posibilidades de creación y manipulación de zonas, apps y usuarios.

Haml. Haml, cuya traducción viene a ser abstracción del lenguaje de marcado HTML, es un lenguaje de marcado ligero con el cual podemos generar HTML de forma rápida, con una sintaxis muy simplificada, con el menor número de caracteres necesarios y utilizando la indentación para formar la estructura HTML. Haml se basa en un principio fundamental: el marcado debe ser bonito. Por ello se centra en la limpieza y legibilidad, y proporciona mayor velocidad de producción.

```
<div class="surf-spot" id="salinas">
  <header>
    <h3>Playa de Salinas</h3>
    <ul class="options">
      <li>

 <a class="view" title="View spot details" href="/surf_spots/5">View</a>
      </li>
      <li>

 <a class="edit" title="Edit spot details" href="/surf_spots/5/edit">Edit</a>
      </li>
    </ul>
  </header>
  <div class="details">
    <dl>
      <dt class="type">Type</dt>
      <dd>Beach brake</dd>
      <dt class="swell-dir">Swell direction</dt>
      <dd>NW</dd>
      <dt class="wind-dir">Wind direction</dt>
      <dd>S, SW, SE</dd>
    </dl>
  </div>
</div>
```

```
#salinas.surf-spot
  %header
    %h3 Playa de Salinas
    %ul.options
      %li
        %a.view{title: "View spot details", href: "/surf_spots/5"} View
      %li
        %a.edit{title: "Edit spot details", href: "/surf_spots/5/edit"} Edit
  .details
    %dl
      %dt.type Type
      %dd Beach brake
      %dt.swell-dir Swell direction
      %dd NW
      %dt.wind-dir Wind direction
      %dd S, SW, SE
```

Figura 6.1. Traducción HTML a Haml

ElasticSearch: Para hacer una relación entre las zonas geográficas y buscar por formas geométricas de forma efectiva y rápida, se ha optado por usar ElasticSearch, que es una potente herramienta que permite indexar un gran volumen de datos y posteriormente hacer consultas sobre ellos soportando, entre otras muchas opciones, búsquedas aproximadas, facetas y resaltado. Se basa en Lucene pero expone su funcionalidad a través de una interfaz REST recibiendo y enviando datos en formato JSON y ocultando mediante esta interfaz los detalles internos de Lucene.

El mapping **geo_shape** permite la indexación y búsqueda con formas geométricas arbitrarias, como rectángulos, círculos o polígonos. También existe la posibilidad de indexar determinados puntos o coordenadas en el mapa, mediante el mapping **geo_point**, que hace que la búsqueda de puntos geométricos en el mapa sea muy rápida.

Para representar las formas en ElasticSearch se utiliza el estándar GeoJSON.

ElasticSearch ha sido una herramienta esencial a la hora de desarrollar todas las funciones de relaciones entre zonas, tales como contención, superposición o cercanía. Además, permite establecer la precisión y los niveles de división del mapa, es decir, a más niveles de división, más precisión y más necesidad de procesamiento es necesaria.

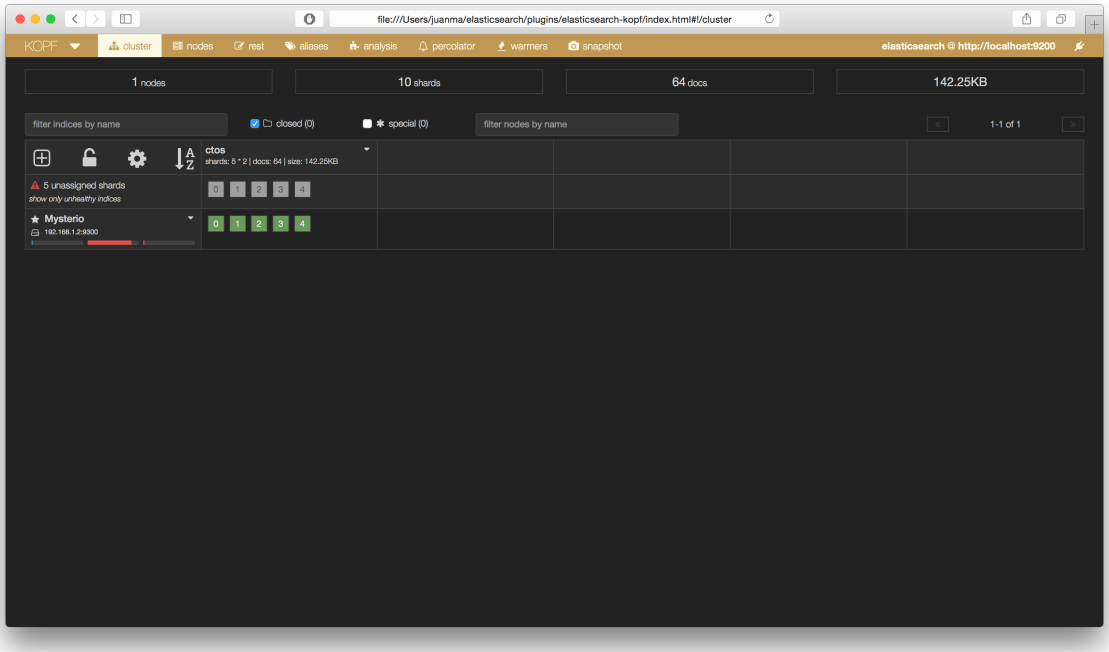


Figura 6.2. Panel de control de ElasticSearch

MongoDB: Es una base de datos NoSQL muy ágil que permite a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan, proporcionando siempre la funcionalidad que los desarrolladores esperan de las bases de datos tradicionales, tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta.

MongoDB proporciona escalabilidad, rendimiento y gran disponibilidad, escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos. MongoDB ofrece un elevado rendimiento, tanto para lectura como para escritura, potenciando la computación en memoria (in-memory). La replicación nativa de MongoDB y la tolerancia a fallos automática ofrece fiabilidad a nivel empresarial y flexibilidad operativa.

El principal motivo por el que se ha optado por usar esta base de datos NoSQL es por la heterogeneidad que supone la creación de zonas geográficas de diferentes formas, y porque funciona muy bien con ElasticSearch.

Git: Se ha usado Git como software de control de versiones. El principal motivo por el que se ha optado por el uso de este software es que los miembros del equipo ya se sentían cómodos usándolo, además de que funciona muy bien y, actualmente, se podría decir que es el más usado, especialmente para el framework de Ruby on Rails. En específico, se han usado los repositorios de Bitbucket y Github a lo largo del proyecto. El repositorio de Bitbucket permite la creación de repositorios privados de manera gratuita para grupos de tamaño reducido y Github proporciona repositorios privados gratuitos para estudiantes. Además, Github es el principal repositorio utilizado para la subida de proyectos de código abierto.

DigitalOcean: Debido a que DigitalOcean proporciona a estudiantes, a través de Github, saldo gratuito para la creación de servidores de *hosting*, se ha utilizado su *IaaS* para la localización de la API en la nube y las pruebas en casos de uso. Se ha montado sobre un servidor con Ubuntu 14.04 x64 y las conexiones se han realizado mediante **ssh**.

Heroku: Es un PaaS que permite la subida de código fuente y realiza automáticamente la instalación en sus servidores y el consecuente despliegue de funcionalidad, proporcionando un panel sencillo y unas herramientas fácilmente utilizables para la subida

constante de cualquier aplicación web. Tiene una alta usabilidad con Ruby on Rails y por eso lo hemos utilizado para subir algunos de los proyectos, entre ellos la landing-page.

Tecnologías descartadas

A continuación se describen todas aquellas tecnologías que se barajaron en un principio para realizar el proyecto, pero que finalmente no se utilizaron.

- **Python**: Se barajó la posibilidad de usar Python como lenguaje para los métodos de la API, pero no se usó finalmente ya que era un lenguaje que no conocía ninguno de los integrantes del equipo y, tras buscar información, se pudo ver que no tenía tanto soporte para creación de servicios web como el seleccionado.
- **Java**: Este lenguaje se descartó a pesar de que todos los integrantes del equipo lo dominan, ya que se hizo una investigación al respecto, y no se recomendaba para el desarrollo de APIs.

También hubo otras tecnologías que se usaron durante un periodo del desarrollo del proyecto, pero por diferentes motivos, se acabaron rechazando.

- **MySQL**: Al empezar a desarrollar CtOS Enabler, se utilizó MySQL como gestor de base de datos, sin embargo, después de algunas semanas se optó por utilizar otro gestor, ya que las bases de datos relacionales son muy rígidas y suponen una traba a la hora de gestionar grandes volúmenes de información variable, como es el caso de CtOS Enabler.
- **XML-RPC**: El protocolo de comunicación con el que actuaba la API durante los seis primeros meses de desarrollo era XML-RPC. El problema surgió cuando se comenzaron a implementar los clientes en los diferentes lenguajes, que permitieron ver que XML-RPC es un protocolo muy costoso de implementar, además de que las respuestas que proporciona son muy complejas de leer y de procesar.

Clientes

Se han creado clientes para CtOS Enabler en diferentes lenguajes para hacer el acceso a la API sencillo y orientado a objetos, de forma que el usuario del framework no se tenga que preocupar por programar ninguna llamada, sino simplemente ejecutar el cliente y acceder a los métodos.

- **Ruby (Ruby on Rails - *gema*-):** El cliente para CtOS Enabler para Ruby se ha creado como una *gema* (paquete estándar y autocontenido) para Rails.
- **Java:** El cliente para Java se ofrece como un proyecto que se puede incluir directamente en cualquier aplicación, o también como plugin en un repositorio Maven.
- **Javascript:** El cliente para Javascript se ha desarrollado para ser incluido como plug-in en aplicaciones web y poder realizar las peticiones a la API haciendo uso del complemento.

Capítulo 7. Casos de uso

Proyectos paralelos

A la vez que se ha ido construyendo el framework de CtOS Enabler, hemos ido desarrollando una serie de aplicaciones que lo utilizan para probar y ejemplificar su funcionalidad. Estas aplicaciones han servido, además, para reorientar el desarrollo de CtOS Enabler y ver qué funcionalidades adicionales resultaban necesarias o aconsejables.

Entre estos proyectos se pueden destacar los siguientes:

Map of History

Map of History es un proyecto que se ha desarrollado utilizando CtOS Enabler para gestionar la geolocalización y definición de zonas.

Se trata de un juego de realidad aumentada, para plataformas móviles (Android y iOS) en el cual hay que seguir una serie de pistas en un mapa real hasta llegar a un objetivo o pista final.

El juego se ha planteado como una ejemplificación del potente funcionamiento de CtOS Enabler y se utilizará en un evento organizado en El Escorial a finales de julio de 2015.

Tecnologías utilizadas

- **Java**: El lenguaje base para desarrollar el juego es Java, ya que los miembros del equipo tienen bastante experiencia con este lenguaje y es muy robusto.
- **Android**: Se han utilizado ciertas funciones de Android para dibujar el mapa directamente para esta plataforma, ya que la calidad del mapa es mejor que si fueran imágenes y la API de Google Maps para Android permite un grado de interacción con el mapa muy alto.
- **iOS**: Del mismo modo que en Android, se procederá a utilizar determinadas funciones de iOS para dibujar el mapa de la aplicación de forma nativa.
- **LibGDX**: LibGDX es un framework multiplataforma de desarrollo de juegos para Windows, Linux y Android. Está escrito en Java con una mezcla de C/C++ para dar soporte y rendimiento a tareas relacionadas con el uso de la física y procesamiento de audio.

LibGDX permite generar una aplicación para escritorio y utilizar el mismo código en Android, de esta manera el proceso de pruebas y depuración se realiza de forma más rápida y cómoda. Con LibGDX nos aseguramos de que la misma aplicación puede funcionar correctamente en diferentes dispositivos. LibGDX está compuesto por una serie de componentes que serán comunes a todas las aplicaciones.

  - Marco de Aplicación, que maneja el bucle principal y además estará encargado del ciclo de vida, es decir, los eventos de creación, destrucción y pausa de la misma.
  - Un componente de Gráficos que permite gestionar la representación de imágenes y objetos gráficos en la pantalla.
  - Un componente de Audio que facilita el acceso a los sonidos y música de la aplicación.
  - Un componente de Entrada y Salida para leer y escribir los diferentes ficheros de datos como, por ejemplo, imágenes, archivos de configuración, sonidos, música, texturas,…
  - Un componente de Entrada que gestiona la entrada a través del teclado, pantalla táctil o acelerómetro.

- **CtOS Java Client**: El cliente para Java de CtOS Enabler se usa en Map of History para hacer las llamadas a la API de forma sencilla.

Tecnologías descartadas

- **Unity3D**: Al comenzar el desarrollo del juego se utilizó Unity3D, ya que es un motor de videojuegos muy potente, cuyo sistema de construcción del entorno 3D es muy sencillo. Sin embargo, después de algunas semanas de pruebas, se desestimó ya que no existe una API de Google Maps para C#, y las llamadas externas desde Unity3D eran muy complejas de realizar con el protocolo XML-RPC, ya que no existían librerías que facilitaran el proceso. Además, el mapa usado eran imágenes estáticas que no proporcionaban una visión realista de calidad del mapa, además de limitar y aumentar la complejidad de la programación del entorno.

Smart Lights

Smart Lights es un proyecto desarrollado en el plazo de dos días en el Smart Cities Urban Datafest, un hackathon organizado por la UCM en el mes de marzo.

La aplicación gestiona los semáforos en los cruces para dar prioridad a vehículos en estado de emergencia: ambulancias, policía, bomberos, etc.

Figura 7.1. Diagrama de funcionamiento SmartLights

El proyecto está estructurado en tres partes principales:

- **Servidor**: El servidor se encarga de los cálculos y envío de información relacionada con los semáforos y la autenticación del cliente.

  Se desarrolló usando el lenguaje Ruby con el framework de Ruby on Rails. Además usa CtOS Enabler, que almacena todos los datos relacionados con los semáforos, como su localización, la zona de influencia, clave de seguridad, etc.

  Se usa basándose en la posición actual del cliente, dada en coordenadas, para devolver los datos de autenticación al cliente.

- **Cliente**: El cliente está localizado en el vehículo o autoridad en estado de emergencia o que necesita regular el tráfico.

Controla la comunicación mediante Bluetooth entre el Arduino conectado al semáforo y el servidor.

Muestra la localización actual y la ruta usada por el vehículo. También da la opción de elegir el nivel de emergencia.

- **Arduino**: Es el dispositivo que controla el semáforo.

  Necesita un código de seguridad para identificar a los usuarios con autoridad para manipularlo y prevenir un uso indebido.

  Tiene un radio de 30 metros para conectar con el cliente antes de que alcance el semáforo.

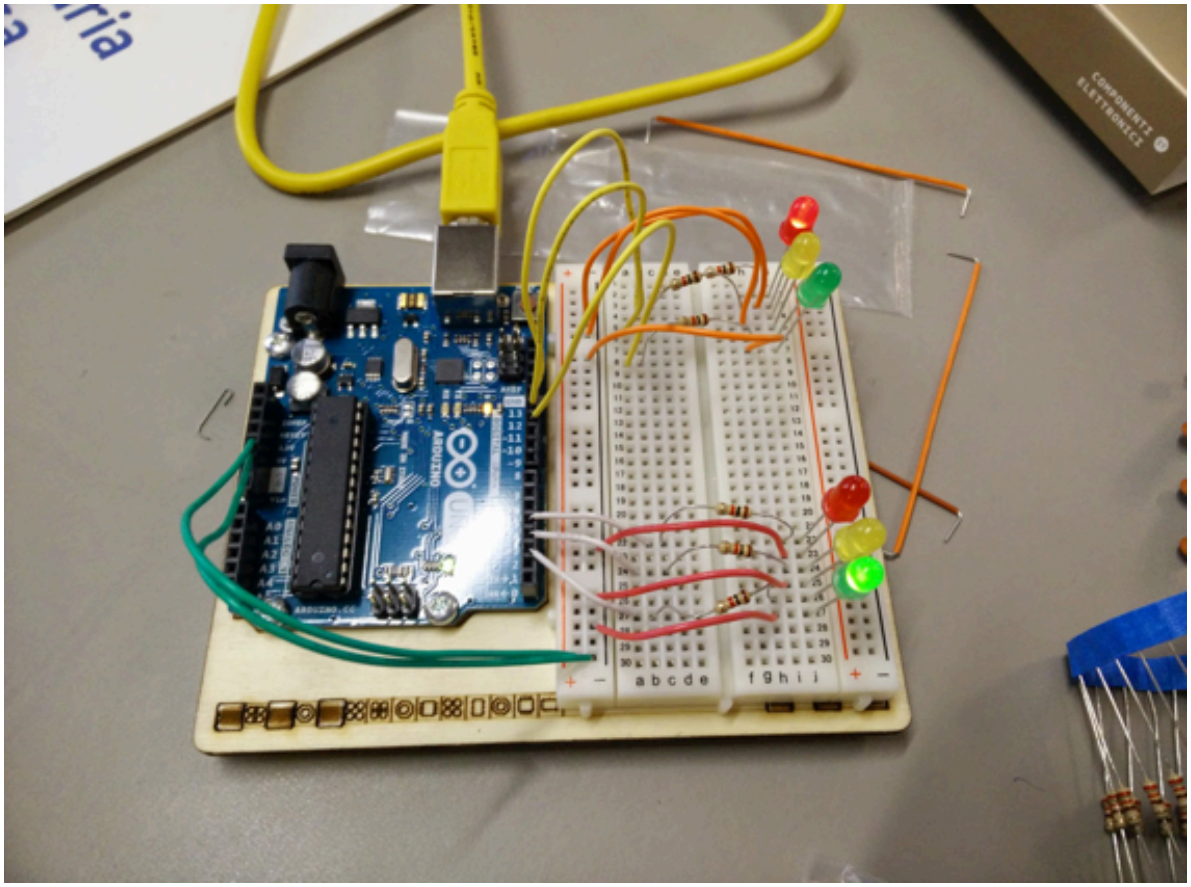


Figura 7.2. Dispositivo Arduino

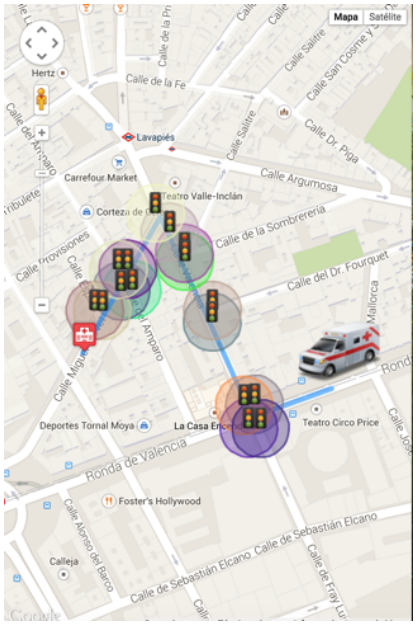


Figura 7.3. Interfaz de la aplicación

Georep

Georep es una aplicación web que se desarrolló en un plazo de dos días. Utiliza CtOS Enabler como base para gestionar las zonas de influencia. Es una aplicación que muestra la reputación de zonas geográficas o barrios según valoraciones de los usuarios, y según bases de datos abiertas de criminalidad y datos de interés.

Es una aplicación sencilla que permite al usuario saber, por ejemplo, si sería interesante mudarse a un barrio determinado o qué ventajas tiene una zona.

Otros posibles casos de uso

Chat de zona

Consiste en un chat que permite la creación de grupos por zonas, e inclusión de usuarios en estos grupos. Cuando un usuario de la aplicación es geolocalizado en un determinado lugar, el framework de CtOS Enabler manda una respuesta a esta aplicación con los datos para incluir al usuario en un grupo de chat de esa zona.

Esta aplicación se había planteado en un principio para el ambiente universitario, por ejemplo, para el Campus de Ciudad Universitaria, en el que las zonas se organizan por Facultades y Aulas, y en el que el usuario, aparte de tener acceso al chat grupal, también tendría acceso a materiales lectivos según dónde esté geográficamente situado.

Eventos geolocalizados

Aplicación que gestiona los determinados eventos de una compañía de forma geolocalizada, permitiendo definir zonas de eventos nuevas y modificarlas, y ofreciendo información sobre el evento a los usuarios. Toda esta información y el proceso de gestión de las zonas geográficas se puede realizar mediante CtOS Enabler, de forma que posteriormente se obtengan datos de uso por parte de los usuarios para un determinado evento, y así concluir estadísticas al respecto de éste.

Capítulo 8. Conclusión y discusiones

El objetivo principal de este Trabajo de Fin de Grado es, tal y como se ha explicado en el presente documento, la creación de un mecanismo computacional de geolocalización que, por un lado, permita optimizar los tiempos de producción de aplicaciones con geolocalización y, por otro, sea capaz de recoger y presentar los datos generados por el geoposicionamiento.

En la actualidad existen diferentes tecnologías dedicadas al ámbito de la geolocalización, desde aplicaciones web para crear mapas personalizados, hasta potentes APIs de geolocalización; pero ninguna de estas APIs permite crear zonas de influencia en las que recoger datos geoposicionados, para a posteriori generar datos de uso a partir de ellos.

Por todo ello, hemos desarrollado para este TFG CtOS Enabler, un framework de geolocalización destinado tanto a desarrolladores que quieran agilizar el tiempo de producción de sus apps y disponer de un potente set de herramientas para consultas relacionadas con geoposicionamiento, como a empresas que dispongan de aplicaciones para sus clientes y quieran obtener de éstos datos de uso basados en la geolocalización.

Además, se han creado clientes en diferentes lenguajes de programación para que los desarrolladores únicamente necesiten incluirlos como plugins en sus apps, y usar las funciones que harán las peticiones a la API. También hemos desarrollado diferentes casos de uso de la API, en forma de aplicaciones, para demostrar el funcionamiento de ésta. El desarrollo de estas aplicaciones nos han permitido aprender nuevas tecnologías y demostrar la aplicabilidad de nuestro proyecto.

Para los usuarios que no tengan conocimientos técnicos se ha desarrollado un panel de control, con el que gestionar sus aplicaciones y los datos geoposicionados de éstas, desde la creación de zonas de influencia sobre un mapa interactivo hasta la visualización de de los datos de uso.

La experiencia desarrollando el proyecto ha sido muy satisfactoria y, aunque no supiéramos desarrollar en Ruby on Rails a principios de curso, los resultados obtenidos han sido muy gratificantes.

Además del desarrollo técnico del proyecto, hemos adquirido muchos conocimientos del ámbito del emprendimiento y la creación de empresas, ya que hemos presentado este trabajo a un concurso de *startups,* que nos ha permitido aprender el proceso que conlleva emprender, y sobre todo, cómo reorientar CtOS Enabler para que sea un servicio atractivo y facilite las tareas a los desarrolladores y las empresas del mundo real.

En definitiva, consideramos que la realización de CtOS Enabler nos ha aportado conocimientos en áreas muy distintas que nos han ayudado a acercarnos al mundo empresarial y a visualizar las complejidades que conlleva desarrollar un proyecto de larga duración, logrando llevarlo a cabo satisfactoriamente.

Chapter 8. Conclusion and discussions

The main objective of this Final Year Project is, as has been explained herein, the creation of a geolocation computing mechanism which, on the one hand, allows to optimize the production speed of geolocation apps and, on the other hand, is capable of collecting and show data generated by the geopositioning.

Currently there are different technologies dedicated to the field of geolocation, from web applications used to create customized maps, to powerful geolocation APIs; but none of these APIs can create areas of influence where to collect geo-data to retrospectively generate usage data from them.

Therefore, we have developed for this FYP CtOS Enabler, a geolocation framework intended for both, developers who want to speed up production time of their apps and have a powerful set of tools for queries related to geopositioning, and companies that have applications for their customers and want to get usage data based on geolocation.

In addition, there have been created clients in different programming languages for the developers to only need to include them as plugins in their apps, and use the included functions that will make requests to the API. We have also developed different use cases of the API, as applications to show how the system works. The development of these applications has enabled us to learn new technologies and demonstrate the usability of our project.

For users who do not have technical skills, it has been developed a control panel to manage their applications and the geo-data, from the creation of zones of influence on an interactive map to the visualization of data usage.
Experience developing the project has been very successful and, even starting without any knowledge in the use of Ruby on Rails, the results have been very gratifying.

Besides the technical development of the project, we have gained a lot of knowledge in the field of entrepreneurship and business creation, as we have presented this work to a contest of startups, which has allowed us to learn the process undertaking involves and, above all, how to reorient CtOS Enabler to be an attractive service and to facilitate the tasks to the developers and companies in the real world.

In conclusion, we believe that the realization of CtOS Enabler has provided us knowledge in very different areas that have helped us to approach the business world and to visualize the complexities of developing a long-term project, managing to carry it out successfully.

Capítulo 9. Trabajo futuro

Lo que se ha ido explicando en apartados anteriores es el trabajo realizado hasta la fecha con CtOS Enabler. Sin embargo, a lo largo del curso han ido surgiendo ideas y modificaciones a realizar que no eran factibles en este periodo de tiempo, pero que nos gustaría implementar en un futuro próximo para mejorar el sistema. A continuación se detallan algunas de ellas.

Minería de Datos (*Data Mining*)

Puesto que la cantidad de datos recogidos por CtOS Enabler es muy grande, ya que se almacenan todas las peticiones al sistema, y datos concernientes a éstas, sería recomendable aplicar técnicas de minería de datos a estos datos para obtener información a partir de la ya existente.

Minería de datos: El objetivo general del proceso de minería de datos consiste en extraer información de un conjunto de datos y transformarla en una estructura comprensible para su uso posterior.

La aplicación de estas técnicas permitiría deducir información nueva que serviría para hacer recomendaciones al usuario, y generar estadísticas más complejas que interrelacionen zonas, apps y usuarios.

Plataforma de computación distribuida: Hadoop

La precisión a la hora de indexar las zonas que permite ElasticSearch es regulable. Sin embargo, una precisión muy alta requiere una capacidad de cómputo muy grande. Por tanto, al hacer la búsqueda por zonas con una precisión muy alta (por ejemplo, en metros), los tiempos de respuesta del sistema se vuelven inadmisibles.

Hadoop es un framework para computación distribuida que soporta aplicaciones con uso intensivo de datos. Utiliza el algoritmo de MapReduce para agilizar las operaciones sobre grandes cantidades de datos, y HDFS, un sistema de archivos distribuido y el principal sistema de almacenamiento en Hadoop.

Hadoop podría ser una buena solución a este problema, permitiendo repartir la capacidad de cómputo entre varios procesos y así obtener una precisión muy alta de búsqueda en zonas.

Generación de zonas automática

En el sistema de CtOS Enabler actual, es necesario que las zonas sean creadas por el cliente, es decir, que se haga una llamada a la API para generar la zona, o que se cree la zona desde el panel gráfico.

En un futuro sería interesante poder generar zonas automáticamente a través de las peticiones que se hagan a la API. Es decir, organizar las peticiones recibidas por coordenadas, y a partir de éstas generar zonas e interrelacionarlas, sin necesidad de que el usuario lo haga de forma activa.

Bases de datos abiertas

Para que el sistema sea más completo, se contempla la posibilidad de incluir bases de datos abiertas, que contengan información sobre elementos geolocalizados (restaurantes, monumentos, edificios, facilidades, etc.) y así incluirlos en los mapas para enriquecer la experiencia de usuario.

Panel de control personalizable

Se considera para un futuro próximo la creación de un panel de control libre para que pueda ser personalizado y modificado al gusto de las empresas, de forma que aquellas que provean servicios en la nube a terceros puedan reutilizar este panel y modificarlo con su logo o añadirle funcionalidad aparte de la que ya ofrece.

Capítulo 10. División de trabajo

A continuación se pasan a detallar las tareas que ha realizado cada uno de los miembros del equipo durante el desarrollo de CtOS Enabler.

En general, se ha seguido un sistema de trabajo equitativo, en el cual se han dividido las diferentes tareas entre todos los miembros del equipo por igual, sin llegar a haber una separación específica de las distintas partes del proyecto. Se ha utilizado este esquema de trabajo al tratarse de un equipo de tres personas, lo que facilitaba la toma de decisiones, y el uso de una metodología de desarrollo ágil, con un continuo contacto entre los componentes del grupo, asegurando que el proceso de diseño y construcción del sistema se realizaba con una evolución rápida y en conocimiento continuo.

Rodrigo Crespo Cepeda

Rodrigo se ha encargado, durante la etapa de definición y diseño del trabajo, de aportar ideas y definir funcionalidad para el proyecto.

Ha colaborado junto con los demás integrantes del equipo en el desarrollo de la base de la API en Ruby on Rails, ha ayudado en la configuración del proyecto y ha hecho la mayoría de los métodos relacionados con la creación, modificación y eliminación de zonas. También ha implementado métodos auxiliares para la generación de respuestas de éxito y error, y para la conversión de los valores de las zonas a JSON.

Ha colaborado en la creación de los métodos encargados de almacenar los datos de uso de la aplicación. En la parte de las búsquedas mediante ElasticSearch, ha realizado los métodos de búsqueda de zonas intersecantes y contenidas en otras.

Además ha realizado las modificaciones y la reestructuración necesarias para convertir el protocolo de llamadas desde XML-RPC a llamadas con JSON. Junto con ello, ha implementado un sistema de versionado de la API para mantener todas las versiones disponibles, y que no sea necesario actualizar las llamadas en el momento exacto en el que hay una nueva versión de la API.

En la parte de los plugins que realizan llamadas a la API de CtOS Enabler, ha hecho el diseño y organización de las clases del plugin en Java, además de implementar las funcionalidades de los modelos básicos que componen dicho plugin.

Ha realizado parte del trabajo en lo que refiere a la construcción de la web landing-page de CtOS Enabler.

En la realización de la app de Smart Lights, se ha encargado de implementar métodos de la parte de *backend* de la app y de la interacción de dicho *backend* con Arduino, es decir, los métodos para establecer la conexión con el dispositivo cliente y verificar su código de seguridad, y los métodos de interacción con el semáforo. Estos últimos se simularon mediante LEDs unidos a la placa de Arduino.

En la construcción de la app de Georep, Rodrigo se encargó de realizar la comunicación con la API y las funciones de *backend* necesarias para la app, como el cálculo de la reputación por zona y la gestión de las opiniones de los usuarios.

Con respecto al juego de realidad aumentada desarrollado utilizando CtOS Enabler, se ha encargado de colaborar en el diseño y especificación de las clases, además de implementar la funcionalidad nativa final del mapa de Android, integrar el cliente de Java en la aplicación y configurar los parámetros de LibGDX para permitir una correcta visualización de los objetos. Esta última tarea se compone de la configuración de la cámara, la luz y el movimiento del mapa.

Además se ha encargado de configurar y gestionar los servidores externos, a los cuales se han subido los distintos proyectos para su visualización y pruebas.

Ha colaborado también en la redacción de esta memoria, y en la búsqueda de recursos que apoyen los datos y tecnologías mencionados en este documento.

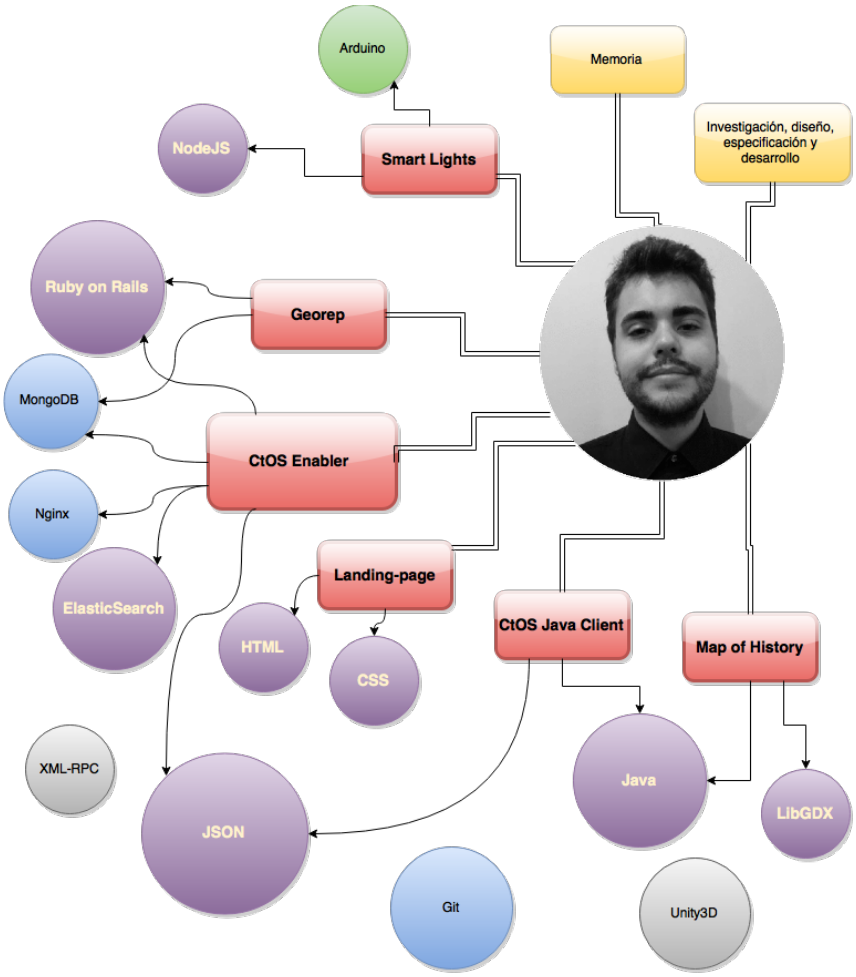


Figura 10.1. Trabajo Rodrigo Crespo

Meriem El Yamri El Khatibi

Durante la etapa de definición y diseño del trabajo, Meriem ha colaborado aportando ideas, definiendo funcionalidad para el proyecto y buscando posibles herramientas que ayudasen al desarrollo del proyecto.

Ha colaborado junto con los demás integrantes del equipo en el desarrollo de la base de la API en Ruby on Rails, ha ayudado en la configuración del proyecto y ha creado gran parte de los métodos de creación, modificación y eliminación de apps. También ha implementado métodos auxiliares para la generación de códigos automáticos para los usuarios y las apps, y se ha encargado de configurar la gestión de usuarios por el sistema.

Ha colaborado en la creación de los métodos encargados de almacenar los datos de uso de la aplicación. En la parte de las búsquedas mediante ElasticSearch, ha desarrollado los métodos concernientes a la búsqueda de zonas contenedoras de una o varias coordenadas.

En la parte de los plugins que realizan llamadas a la API de CtOS Enabler, se ha encargado de implementar el plugin para Ruby, encapsulado en una "gema" para Ruby on Rails. También ha colaborado en la construcción de las llamadas a la API del plugin de Java, de forma que todas las funciones de la API se puedan llamar pasando los parámetros necesarios e indicando la consulta a hacer, pero sin tener que generar consultas en JSON.

Ha realizado parte del trabajo en lo que refiere a la construcción de la web landing-page de CtOS Enabler. También ha creado el logo y los colores del proyecto, los cuales han sido utilizados en la página web, el logo y en documentos.

En la realización de la app de Smart Lights, se ha encargado de colaborar en la creación del *frontend*, la parte visual de la aplicación y la interacción con el usuario final. También ha colaborado en el *backend* de la aplicación implementado las llamadas a las funciones de la API y los métodos necesarios para la gestión de los datos de la aplicación.

En la construcción de la app de Georep, Meriem se ha encargado de proporcionar apoyo en la realización de las funciones de llamadas a la API en la parte de *backend*, pero se ha encargado de la parte visual y de interacción con el usuario en mayor medida.

Con respecto al juego de realidad aumentada desarrollado utilizando CtOS Enabler, se ha encargado de colaborar en el diseño y especificación de las clases, y de adaptar la creación del mapa nativo de Android a que sea reconocido por el *framework* de LibGDX. También se ha

encargado de la creación de diálogos en dicho *framework* que muestren las pistas, y de los métodos auxiliares que permiten la obtención de los datos desde la API y su transformación al formato del juego. Mediante herramientas de diseño ha colaborado en la creación de modelos y pantallas para los diálogos y las ventanas del juego.

Ha colaborado también en la redacción de esta memoria, y en la búsqueda de recursos que apoyen los datos y tecnologías mencionados en este documento.

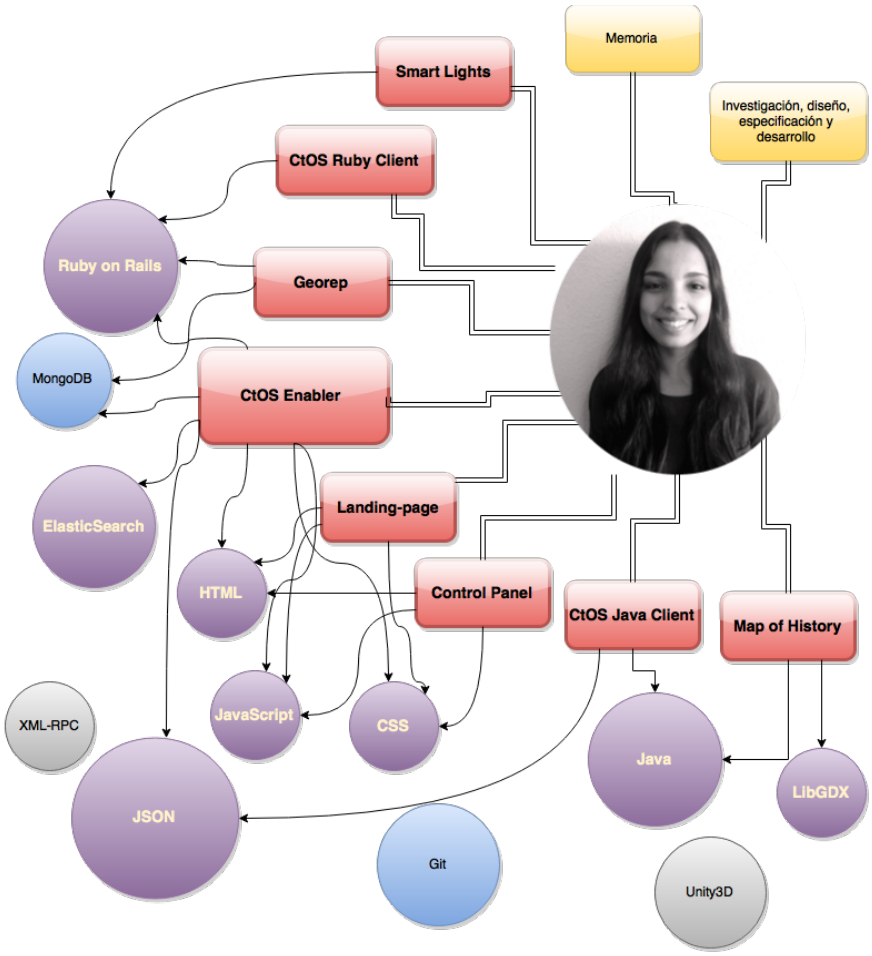


Figura 10.2. Trabajo Meriem El Yamri

Juan Manuel Carrera García

En la fase de definición y diseño del trabajo, Juan Manuel ha colaborado aportando ideas, proponiendo nuevas funcionalidades y usos del sistema, y se ha encargado de buscar sistemas similares y datos sobre geolocalización que pudiesen ayudar a definir mejor el proyecto.

Ha colaborado junto con los demás integrantes del equipo en el desarrollo de la base de la API en Ruby on Rails, ha ayudado en la configuración del proyecto y se ha encargado de realizar métodos auxiliares de búsqueda con funciones que proporciona MongoDB, así como diversos métodos que ofrece la API.

También ha sido el encargado de integrar la funcionalidad de transformación de calles a coordenadas y viceversa.

En lo concerniente al panel de control de CtOS Enabler, ha realizado la mayoría de las tareas. Ha maquetado el panel para que se adapte a las necesidades de los usuarios, y ha creado todos los métodos de interacción con el usuario. Estos métodos son los que permiten a un usuario con un perfil no técnico poder crear zonas de influencia, modificarlas o eliminarlas. También se pueden crear apps a partir del panel de control, visualizar los datos de uso y varias funciones más que permiten interactuar con las apps desde un panel gráfico y sencillo.

En la parte de los plugins que realizan llamadas a la API de CtOS Enabler, se ha encargado de implementar el plugin para Javascript, que en un principio se iba a usar para el panel de control del sistema. Posteriormente, al estar el panel de control integrado en el mismo proyecto, no se ha necesitado. Este plugin funciona para cualquier aplicación web que incorpore CtOS Enabler y quiera crear su propio panel de control personalizado, sin tener porque utilizar el que se proporciona.

En la realización de la app de Smart Lights, se ha encargado en mayor medida del *frontend*, la parte visual de la aplicación y la interacción con el usuario final. Ha creado los métodos de dibujo de zonas y semáforos en el mapa, así como toda la gestión de información proveniente de la API para mostrarla posteriormente.

Mediante herramientas de diseño, Juan Manuel se ha encargado de gran parte de los trabajo de maquetación de los documentos, y de la organización y búsqueda de recursos para todos los problemas que han ido surgiendo a lo largo del desarrollo de este proyecto. También ha colaborado con los demás miembros del equipo en la redacción de esta Memoria y en la

búsqueda de información y referencias para apoyar los datos mencionados en este documento.

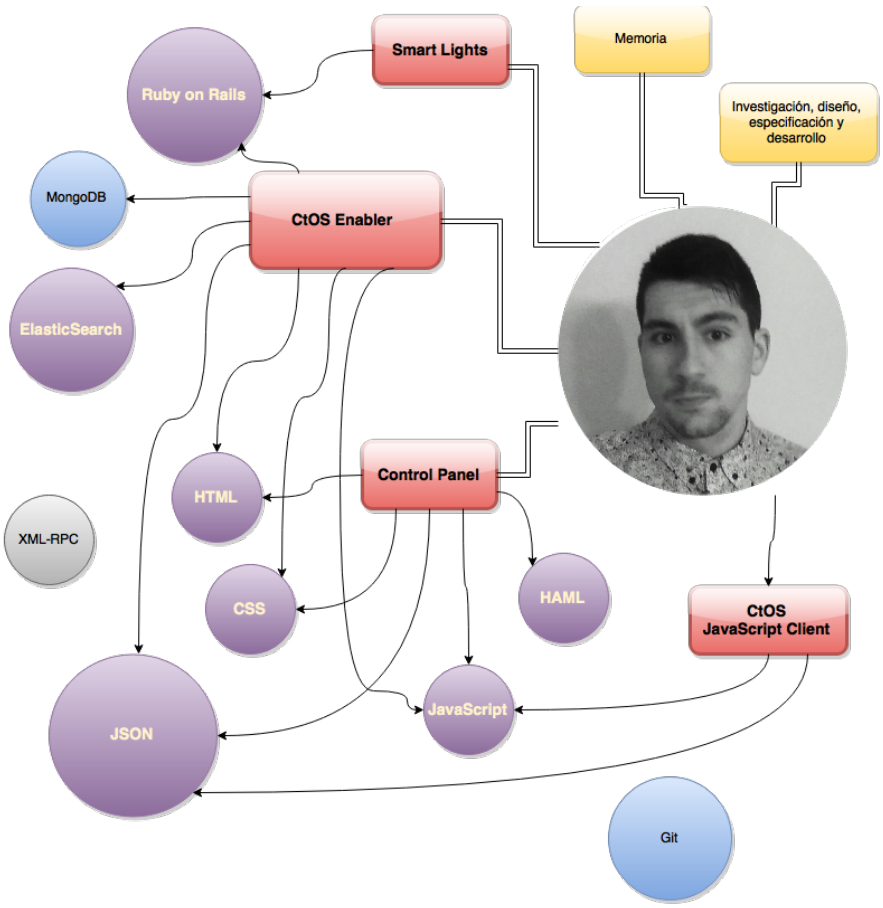


Figura 10.3. Trabajo Juan Manuel Carrera

Referencias

1. *Ruby: Documentation*. http://ruby-doc.org
2. *Ruby on Rails: Documentation.* http://rubyonrails.org/documentation
3. *Rails for Zombies: Code School.* http://railsforzombies.org
4. *RailsCasts: Ruby on Rails Screencasts.* http://railscasts.com
5. *Go Rails: Tutorials.* https://gorails.com
6. *ElasticSearch: Guide.* https://www.elastic.co/guide/index.html
7. *MongoDB: Manual.* http://docs.mongodb.org/manual/
8. *Mongoid: Documentation.* http://mongoid.org/en/mongoid/index.html
9. *GeoJSON.* http://geojson.org
10. *LibGDX: Documentation.* https://github.com/libgdx/libgdx/wiki
11. *W3schools: JavaScript and HTML DOM.* http://www.w3schools.com/jsref
12. *Bootstrap: Documentation.* http://getbootstrap.com
13. *Arduino: Reference.* http://www.arduino.cc/en/Reference/HomePage
14. *Android: Documentation*. http://developer.android.com/develop/index.html
15. *RubyMine: Documentation.* https://www.jetbrains.com/ruby/documentation
16. *Git: Documentation.* https://git-scm.com/documentation
17. *Wikipedia: XML-RPC.* https://en.wikipedia.org/?title=XML-RPC
18. *Unity3D: Manual.* http://docs.unity3d.com/Manual/index.html
19. *Unity3D: Tutorials.* http://unity3d.com/learn/tutorials/modules
20. *Nginx: Documentation*. http://nginx.org/en/docs
21. *HTTP Apache: Documentation.* http://httpd.apache.org/docs/2.2/es
22. *Heroku: Dev Center.* https://devcenter.heroku.com
23. *DigitalOcean: Tutorials* https://www.digitalocean.com/community/tutorials
24. *Bluemix: Documentation*. https://www.ng.bluemix.net/docs

Anexo I. Manuales de usuario

Desarrollador

Crear una Aplicación

URL	HTTP Method
/api/v1/app	POST

```
{
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example"
      },
      "query":
      {
            "app":
            {
                  "name":"app_name",
                  "plan":"app_plan"
            }
      }
}
```

Figura AI.1. Crear aplicación

Editar/Actualizar una Aplicación

URL	HTTP Method
/api/v1/app	PUT

```
{
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example"
      },
      "query":
      {
            "app":
            {
                  "code": "app_code",
                  "name": "app_name",
                  "plan": "app_plan"
                }
      }

}
```

Figura AI.2. Editar aplicación

Eliminar una Aplicación

URL	HTTP Method
/api/v1/app	DELETE

```
{
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example"
      },
      "query":
      {
            "app":
            {
                  "code": "app_code"
            }
      }
}
```

Figura AI.3. Eliminar aplicación

Mostrar las zonas de influencia

URL	HTTP Method
/api/v1/zone/show	POST

```
{
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example",
            "app_code":"app_code"
      },
      "query":
      {
            "zone":
            {
                  "name": "zone_name"
            }
      }
}
```

Figura AI.4. Mostrar zonas de influencia

El objeto JSON expuesto se utiliza para obtener los datos de una determinada zona de influencia, en este caso de zone_name. Si, por el contrario, se quieren obtener los datos de todas las zonas de influencia de una app, la consulta se realiza sin objeto "name":"zone_name", es decir, con el objeto "query" vacío.

Crear zonas de influencia

En la creación de zonas de influencia hay que destacar dos factores: la forma geométrica de la zona y la forma de expresar los puntos para definir la zona. Respecto a las formas geométricas, las zonas pueden ser poligonales y circulares; y respecto a la forma de expresar los puntos para definir las zonas, hay que diferenciar que pueden ser definidas por coordenadas geográficas o mediantes direcciones postales[3].

[3] Las zonas de influencia se podrán definir a partir de direcciones postales siempre que sea posible.

URL	HTTP Method
/api/v1/zone/create	POST

```
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example",
                  "app_code":"app_code"
}
```

```
"query":
{
      "zone":
      {
            "name": "zone_name",
            "is_object": true,
            "zone_class": "zone_class",
            "shape":
            {
                  "type": "circle",
                  "coordinate":
                  [
                              40.458944,
                              -3.689712
                  ],
                  "radius": 3
             },
            "data_info":
            {
                  "data": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua..."
            }
      }
}
```

Figura AI.5. Crear zona circular con coordenadas geográficas

URL	HTTP Method
/api/v1/zone/create	POST

```
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example",
                  "app_code":"app_code"
}
```

```
"query":
{
      "zone":
      {
            "name": "zone_name",
            "is_object": true,
            "zone_class": "zone_class",
            "shape":
            {
                  "type": "circle",
                  "coordinate": "Av. de la Alameda, 16, 28140, Fuente
el Saz de Jarama, España",
                  "radius": 3
             },
            "data_info":
            {
                  "data": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua..."
            }
      }
}
```

Figura AI.6. Crear zona circular con direcciones postales

URL	HTTP Method
/api/v1/zone/create	POST

```
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example",
                  "app_code":"app_code"
}
```

```
"query":
{
      "zone":
      {
            "name": "zone_name",
            "is_object": true,
            "zone_class": "zone_class",
            "shape":
            {
                  "type": "polygon",
                  "coordinate":
                  [
                        [ 1, 1 ],
                        [ 1, 2 ],
                        [ 2, 2 ],
                        [ 2, 1 ]
                  ]
             },
            "data_info":
            {
                  "data": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua..."
            }
      }
}
```

Figura AI.7. Crear zona poligonal con coordenadas geográficas

URL	HTTP Method
/api/v1/zone/create	POST

```
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example",
                  "app_code":"app_code"
}
```

```
"query":
{
      "zone":
      {
            "name": "zone_name",
            "is_object": true,
            "zone_class": "zone_class",
            "shape":
            {
                  "type": "polygon",
                  "coordinate":
                  [
                        "Calle de Méndez Álvaro, 83, 28045, Madrid,
                  España",
                       "Calle del Dr Calero, 37, 28221, Majadahonda,
                  España"
                  ],
             },
            "data_info":
            {
                  "data": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua..."
            }
      }
}
```

Figura AI.8. Crear zona poligonal con direcciones postales

Editar zonas de influencia

En la edición de zonas de influencia, al igual que en la creación, los datos son diferentes en función de la forma de la zona y de si usamos coordenadas o direcciones postales para definirla.

Vamos a ver un ejemplo de los datos necesarios para modificar una zona de influencia circular definida por coordenadas:

URL	HTTP Method
/api/v1/zone/update	POST

```
{
            "authentication":
            {
                  "email":"email@example.com",
                  "user_code":"user_code_example",
                  "app_code":"app_code"
            },
            "query":
            {
                  "zone":
                  {
                        "name": "Clue 1 updated",
                        "new_name": "Clue 1",
                        "shape":
                        {
                                    "type": "circle",
                                    "coordinate":
                                          [ 40.477998, -3.713164 ],
                                    "radius": 45
                         },
                         "data_info": {
                                    "data": "Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
labore et dolore magna aliqua..."
                        }
                  }
             }
}
```

Figura AI.9. Editar zona

Eliminar zonas de influencia

URL	HTTP Method
/api/v1/zone/destroy	POST

```
{
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example",
            "app_code":"app_code"
      },
      "query":
      {
            "zone":
            {
                  "name": "zone_name"
            }
      }
}
```

Figura AI.10. Eliminar zona

Relación de zonas y coordenadas

En este punto se muestran las funciones que proporciona la API para poder realizar consultas acerca de la posición de las zonas de influencia respecto a otras, o respecto a coordenadas específicas.

within

Esta función se encarga de devolver al usuario las zonas de influencia contenidas dentro de una dada. Para ello, necesita como parámetro de entrada una zona determinada de una aplicación, sobre la que se realiza la consulta, para así devolver una respuesta con todas las zonas de influencia de la aplicación contenidas dentro de la zona input.

URL	HTTP Method
/api/v1/zone/within	POST

```
{
            "authentication":
            {
                  "email":"email@example.com",
                  "user_code":"user_code_example",
                  "app_code":"app_code"
            },
            "query":
            {
                  "zone":
                  {
                        "name": "zone_name"
                  }
            }
}
```

Figura AI.11. Consulta "within"

container

Esta función, al contrario que la anterior, se encarga de devolver al usuario las zonas de influencia que contienen a una dada. Para ello, necesita como parámetro de entrada una zona determinada de una aplicación, sobre la que se realiza la consulta, para así devolver una respuesta con todas las zonas de influencia de la aplicación que contiene a la zona input.

URL	HTTP Method
/api/v1/zone/container	POST

```
{
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example",
            "app_code":"app_code"
      },
      "query":
      {
            "zone":
            {
                  "name": "zone_name"
            }
      }
}
```

Figura AI.12. Consulta "container"

zone_within

Esta función se encarga de devolver al usuario una respuesta que especifique si una determinada zona de influencia está dentro de otra. Para ello, necesita como parámetros de entrada los nombres de dos zonas de la aplicación, donde una es la "parent_zone" y otra la "child_zone".

URL	HTTP Method
/api/v1/zone/zone_within	POST

```
{
        "authentication":
        {
            "email":"email@example.com",
            "user_code":"user_code_example",
            "app_code":"app_code"
        },
        "query":
        {
            "parent_zone":
            {
                "name": "zone_name"
            },
            "child_zone":
            {
                "name": "zone_name"
            }
        }
    }
```

Figura AI.13. Consulta "zone_within"

near

Esta función se encarga de devolver las zonas cercanas a una determinada coordenada estableciendo el radio de acción sobre el que se desea buscar. Para ello, necesita como parámetro de entrada una coordenada y la distancia en la que se busca alrededor de dicha coordenada.

URL	HTTP Method
/api/v1/zone/near	POST

```
{
            "authentication":
            {
                  "email":"email@example.com",
                  "user_code":"user_code_example",
                  "app_code":"app_code"
            },
            "query":
            {
                  "coordinate": "[\"-20.1\" , \"-20.1\"]",
                  "range": 10000
            }
}
```

Figura AI.14. Consulta "near"

coordinates_within

Esta función se encarga de devolver al usuario si una determinada zona de influencia contiene una serie de coordenadas. Para ello, necesita como parámetros de entrada el nombre de la zona y un array con las coordenadas que se quiere consultar si están dentro de dicha zona.

URL	HTTP Method
/api/v1/coordinates/within	POST

```
{
      "authentication":
      {
            "email":"email@example.com",
            "user_code":"user_code_example",
            "app_code":"app_code"
      },
      "query":
      {
            "coordinates":
            [
                  [ 40.478108, -3.712747 ]
            ],
            "parent_zone":
            {
                  "name": "zone_name"
            }
      }
}
```

Figura AI.15. Consulta "coordinates_within"

coordinates_container

Esta función se encarga de devolver en las que están dentro unas determinadas coordenadas. Para ello, necesita como parámetros de entrada un array de las coordenadas que se quieren consultar.

URL	HTTP Method
/api/v1/coordinates/container	POST

```
{
	"authentication":
	{
		"email":"email@example.com",
		"user_code":"user_code_example",
		"app_code":"app_code"
	},
	"query":
	{
		"coordinates":
		[
			[ 40.478108, -3.712747 ]
		]
	}
}
```

Figura AI.16. Consulta "coordinates_container"

A todas las consultas anteriores, se le añade un parámetro "stats" para almacenar datos de la petición. Este parámetro tiene como dato obligatorios el ID del dispositivo y de forma opcional la localización del dispositivo y cualquier otra información que sea de interés.

Stats

```
"stats":
{
	"device_id":"device_id_example",
	"location":[ 40.478108, -3.712747 ],
	"age":"29",
	"sex":"female",
	...

},
```

Figura AI.17. Stats

Administrador

El usuario Administrador es un usuario que tendrá acceso al panel de control una vez creada la aplicación que utiliza CtOS Enabler. Es un usuario que podrá definir las zonas de influencia, modificarlas, y ver datos de uso al respecto de éstas.

También podrá crear, modificar y eliminar aplicaciones, ya que un mismo administrador puede tener varias aplicaciones usando CtOS Enabler.

Este usuario no tiene por qué tener un perfil técnico, ya que la interacción con el panel gráfico es sencilla y el usuario no necesita realizar llamadas a la API, porque el panel actúa como una capa de abstracción que hace esto por el usuario.

Las acciones que podrá realizar un administrador desde el panel de control son acciones relacionadas con la gestión de apps y zonas de influencia, y con la consulta de datos de uso relacionados con los datos geoposicionados generados por las apps.

A continuación, se explicarán dichas acciones con capturas de pantalla explicativas.

Apps

La página desde la que se gestionan las aplicaciones es la siguiente:

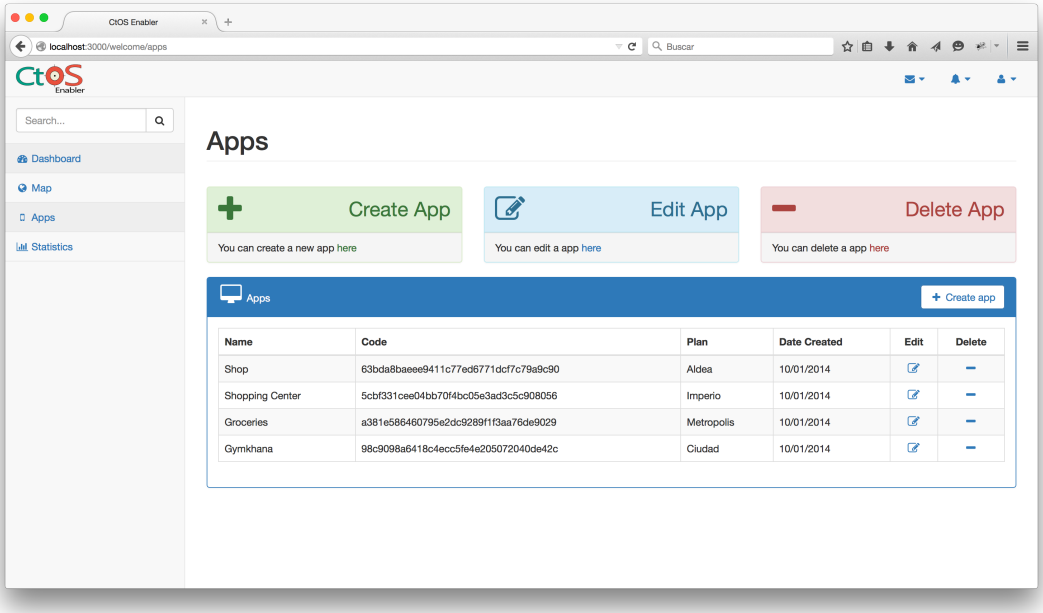


Figura AI.18. Página principal aplicaciones

Crear una aplicación

Para crear una app se pulsará en el botón “Create App” y en el pop-up, se completarán los datos necesarios que son el nombre de la aplicación y el plan que desea el usuario para ella.

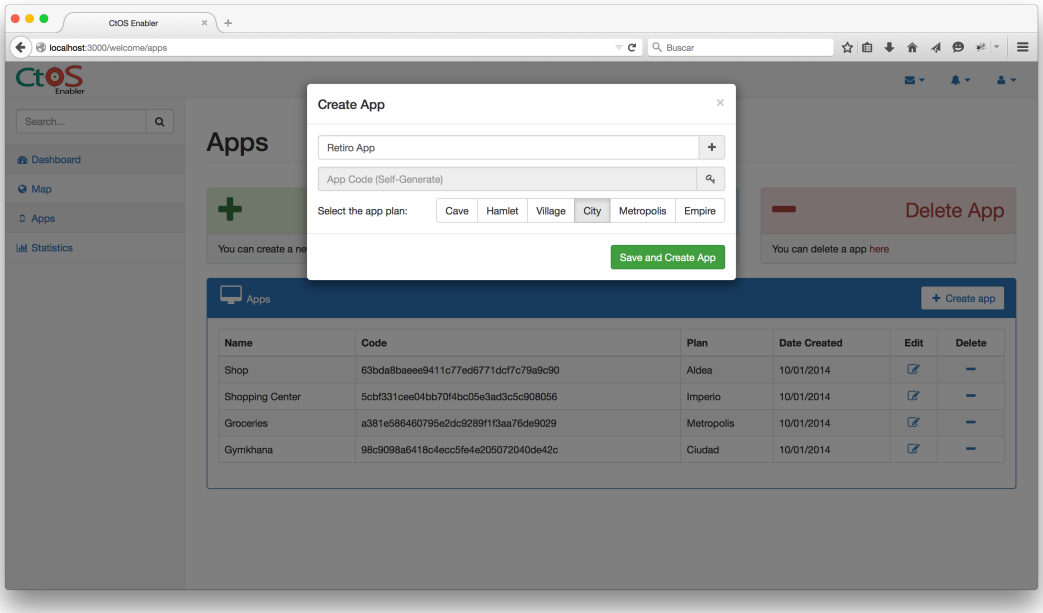


Figura AI.19. Crear aplicación

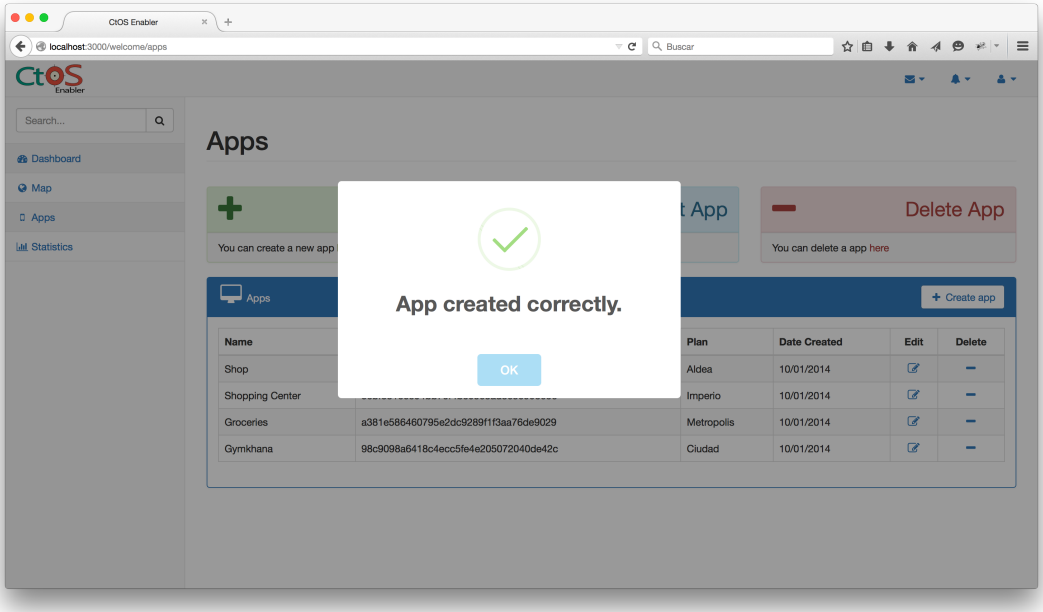


Figura AI.20. Aplicación creada

Editar una aplicación

Para editar o actualizar una app se pulsará el botón de "Edit" de la aplicación correspondiente y en el pop-up, se completarán los datos necesarios que son el nombre de la aplicación y el plan que desea el usuario para ella.

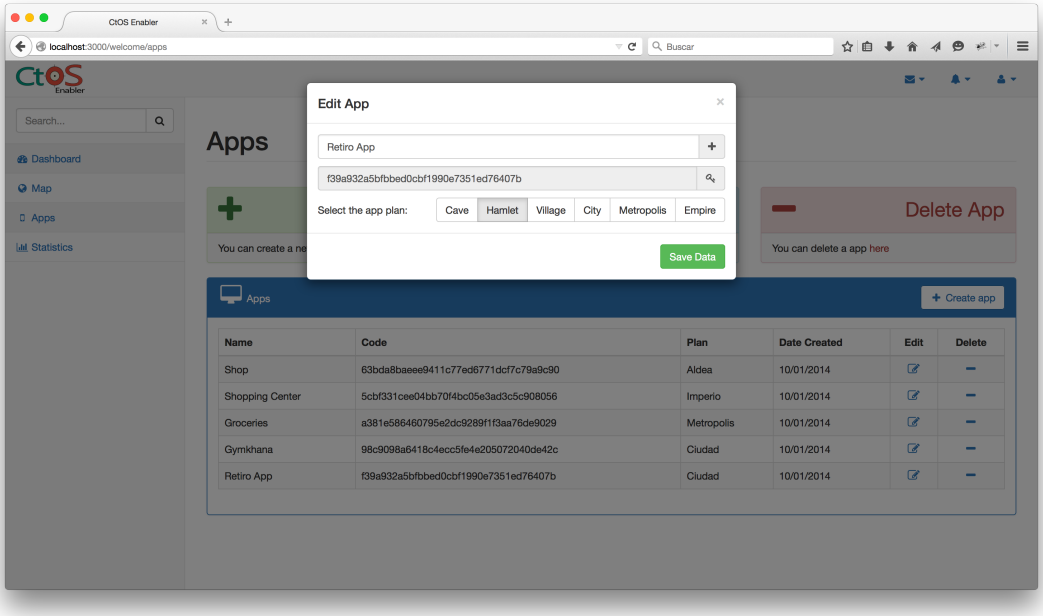


Figura AI.21. Editar aplicación

Eliminar una aplicación

Para eliminar una app se pulsará el botón de "Delete" de la aplicación correspondiente y después de confirmar el borrado se procederá a ello.

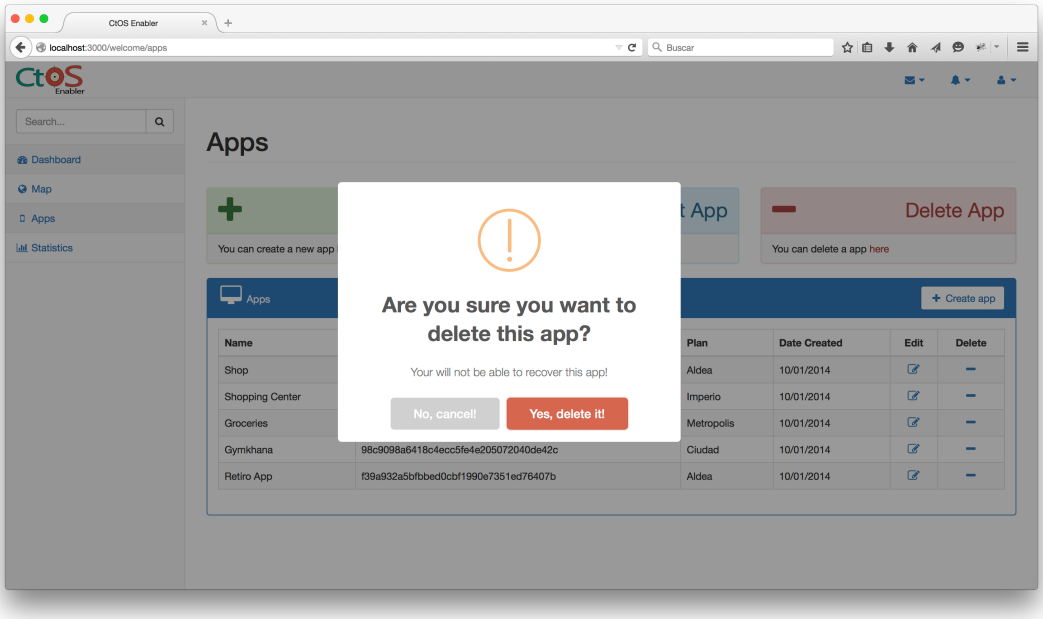


Figura AI.22. Eliminar aplicación

Maps

Para acceder a la página desde la que se gestionan las zonas de influencia de las aplicaciones tenemos que seleccionar primero una aplicación después se cargará el mapa, y finalmente lo tendremos listo para gestionar las zonas de la aplicación seleccionada.

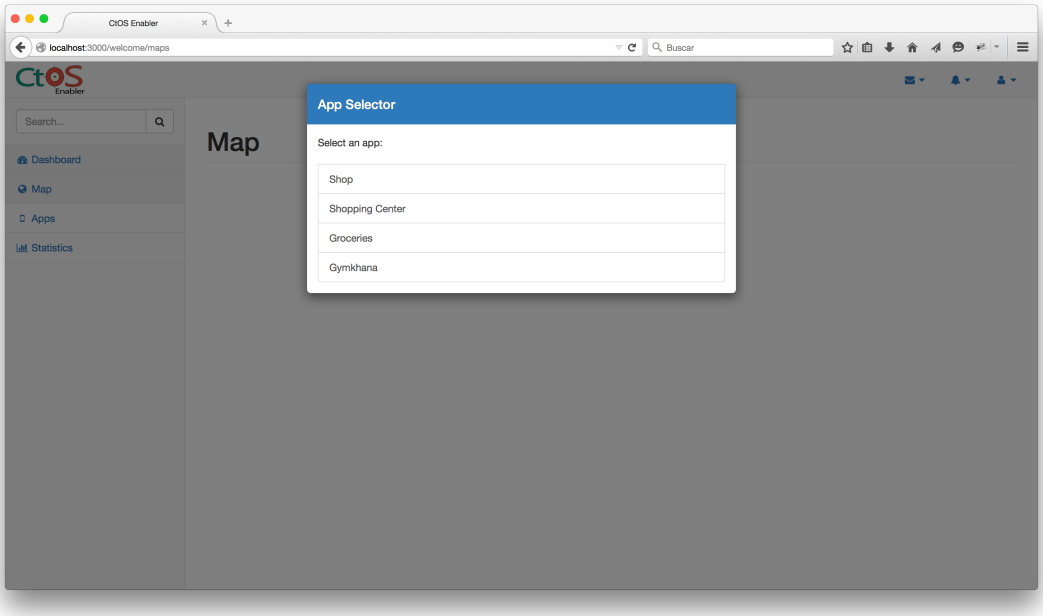
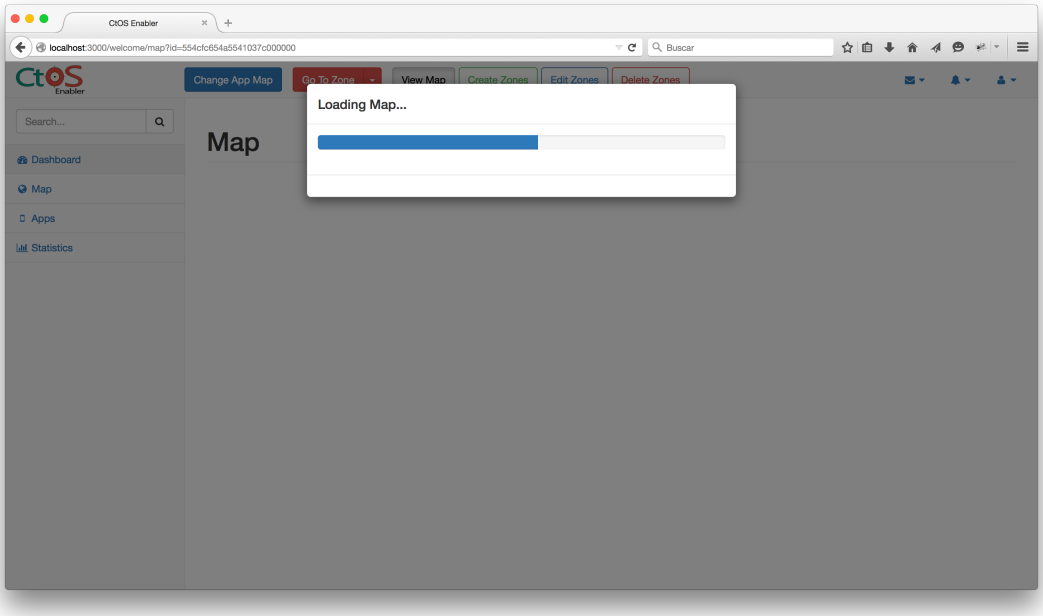


Figura AI.23. Selector aplicación



FiguraAI.24. Cargando mapa

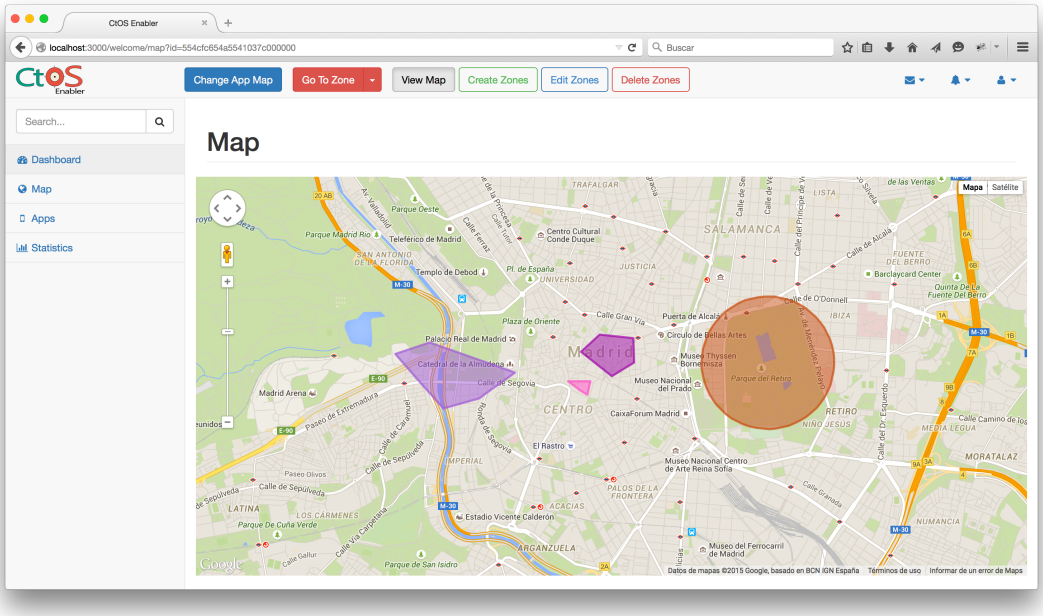
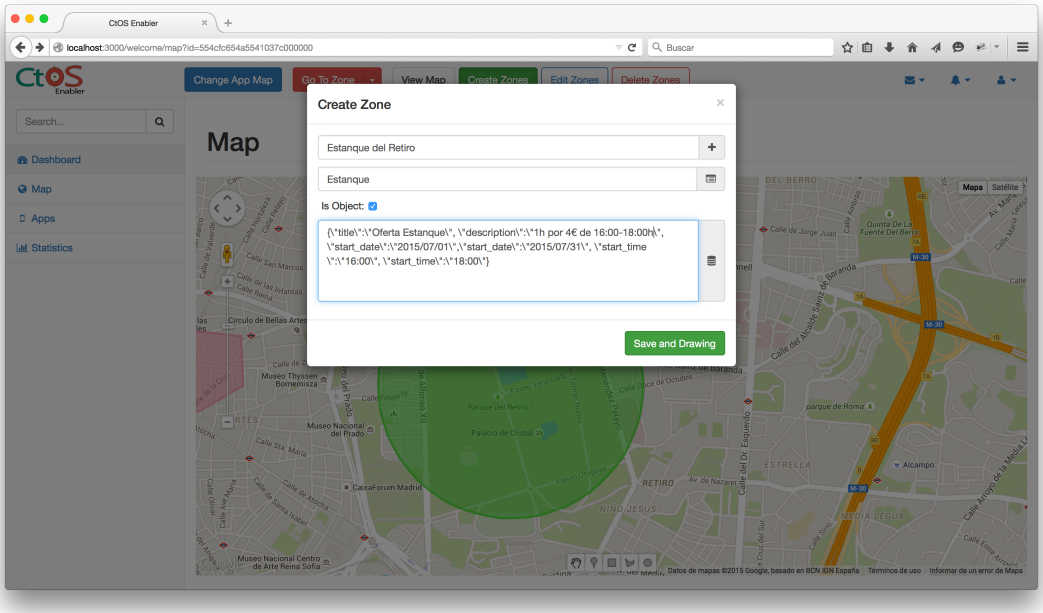


Figura AI.25. Mapa de aplicación

Crear una zona de influencia

Para crear una zona se pulsará en el botón “Create Zones” y en el pop-up emergente se completarán los datos necesarios que son el nombre, la clase y el campo data de la zona indicando si se trata de un objeto o no.

A continuación, se pintará la zona de influencia sobre el mapa con la forma geométrica deseada y se creará la zona.



FiguraAI.26. Crear zona

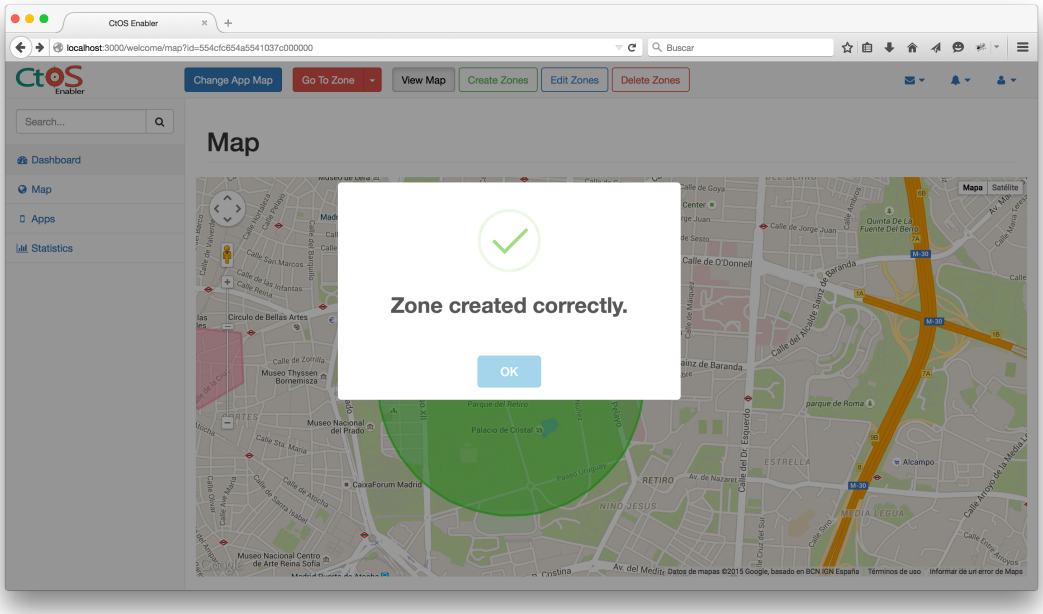
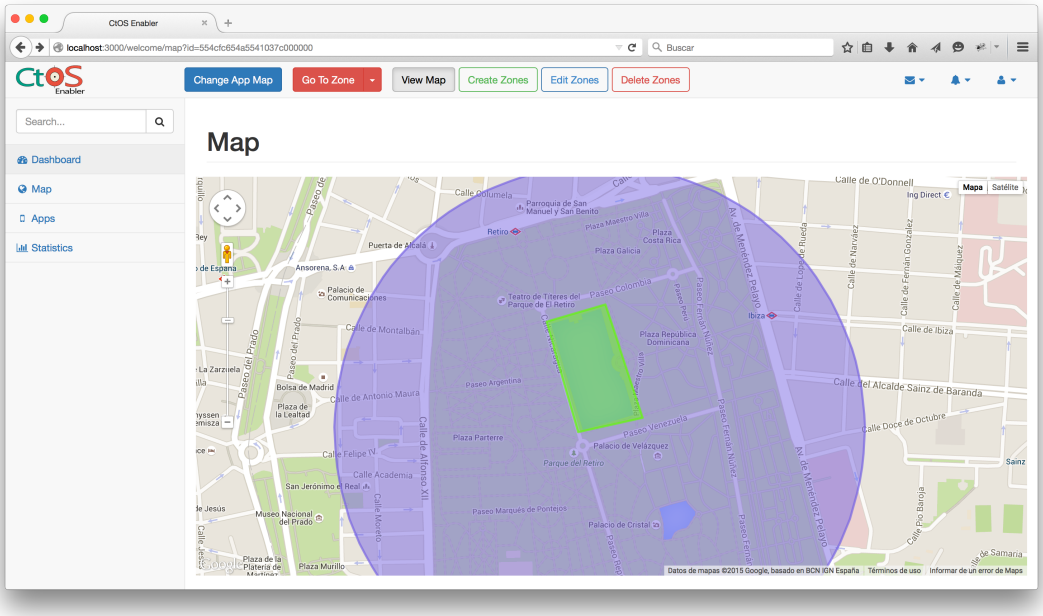


Figura AI.27. Confirmación creación zona

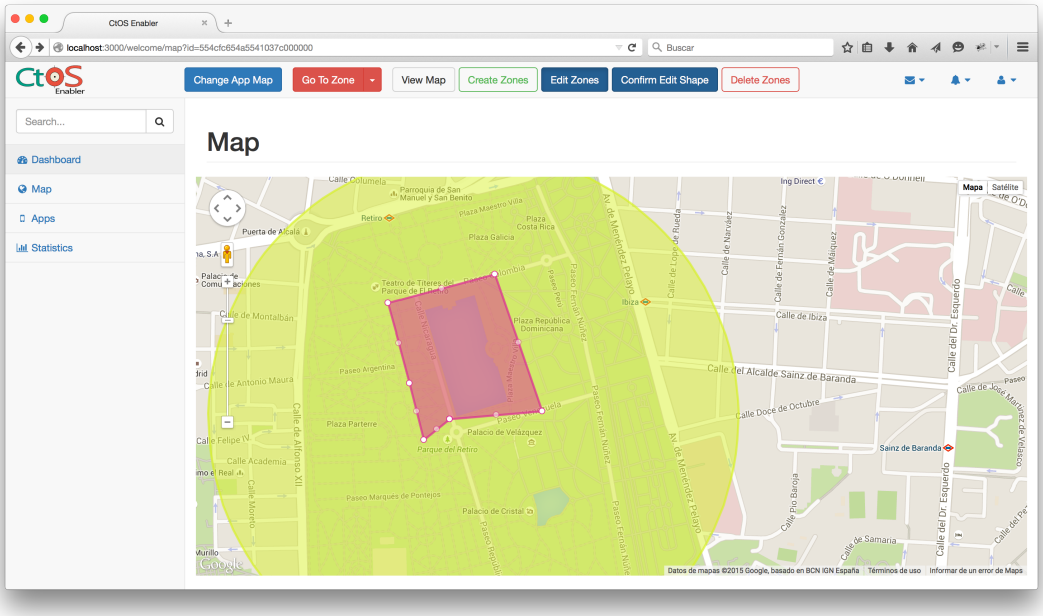


FiguraAI.28. Zona creada

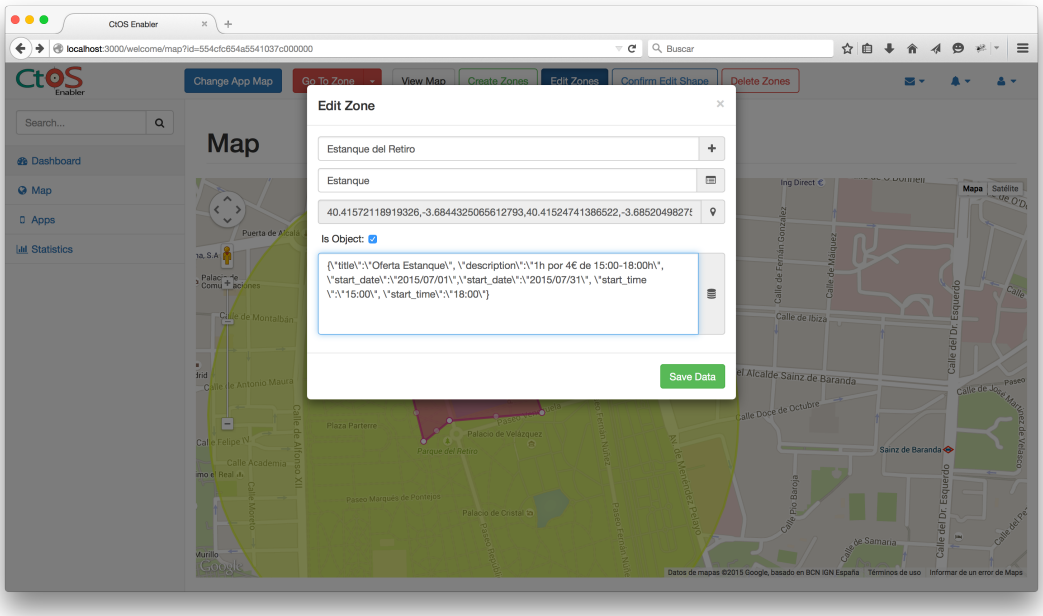
Editar una zona de influencia

Para editar una zona se pulsará en el botón “Edit Zones”, se modificará y/o moverá la forma geométrica sobre el mapa, en el caso de que el usuario lo desee, y se confirmará con “Confirm Edit Shape”.

Al confirmar la forma, aparecerá un pop-up en el que se permitirá modificar el nombre, la clase y el campo data de la zona indicando si se trata de un objeto o no, en el caso de que el usuario así lo desee, y se confirmarán los datos.



FiguraAI.29. Seleccionar zona



FiguraAI.30. Editar zona

Eliminar una zona de influencia

Para eliminar una zona de influencia se pulsará el botón de “Delete Zones”, se seleccionarán las zonas a eliminar marcando su forma geométrica, y se confirmará con el botón “Delete Selected Zones”.

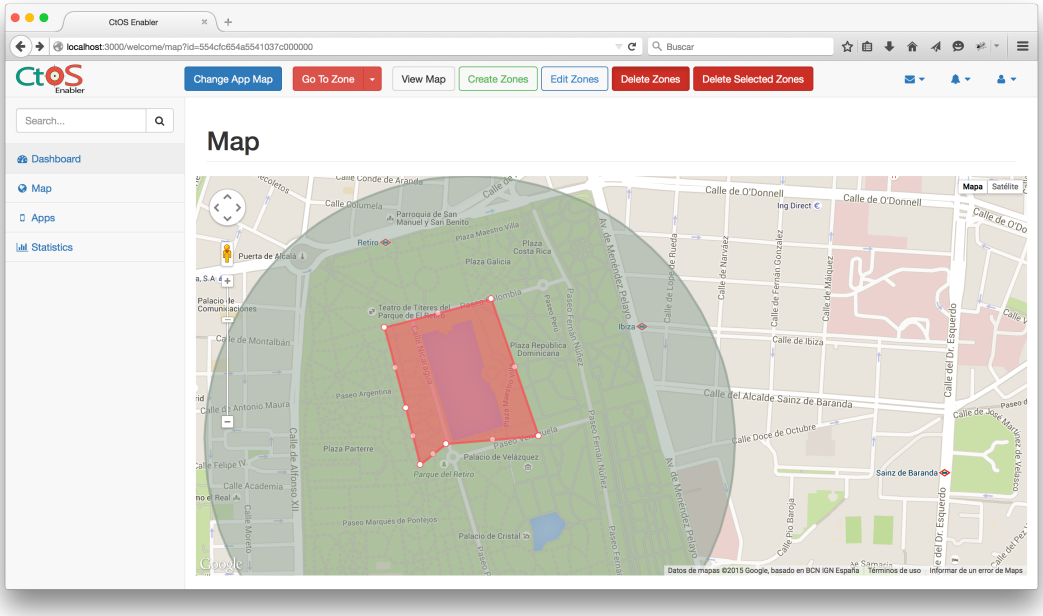


Figura AI.31. Seleccionar zonas

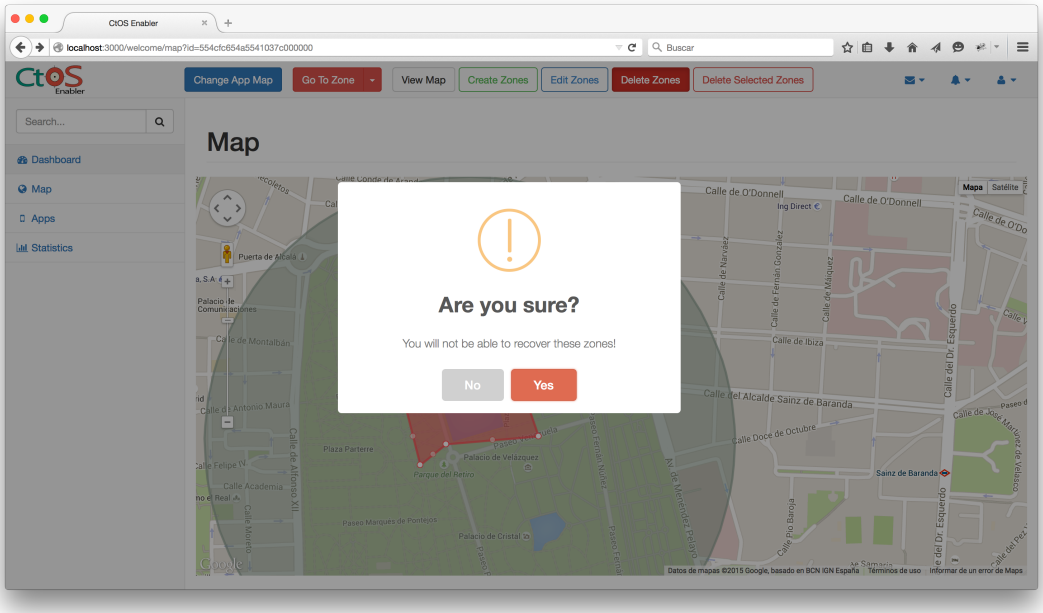


Figura AI.32. Eliminar zonas

Anexo II. Repercusión

El proyecto CtOS Enabler no se ha limitado a ser un proyecto de TFG, sino que ha ido más allá, ya que se ha presentado a diferentes concursos y eventos, obteniendo mucho reconocimiento. En el siguiente cronograma se detallan los eventos ocurridos durante este año académico y algunos futuros, y a continuación, se explican más en detalle los concursos y eventos, junto con los premios ganados.

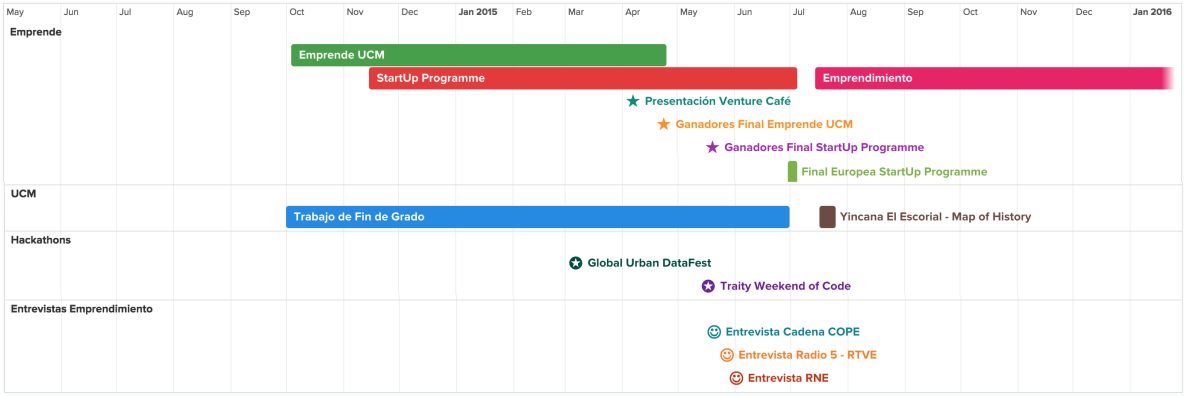


Figura AII.1. Cronograma CtOS Enabler

StartUp Programme

Figura AII.2. Ganadores StartUp Programme

Startup Programme es un programa educativo que tiene como objetivo fundamental fomentar el espíritu emprendedor en el ámbito universitario y favorecer la creación de empleo.

Es un proyecto de índole internacional que se desarrolla de manera simultánea en 14 países europeos, miembros de Junior Achievement – Young Enterprise Europe. A través del desarrollo de un Plan de Empresa y acompañados por tutores de las universidades y asesores voluntarios de organizaciones especializadas y empresas de referencia, los participantes analizan la viabilidad de su idea de negocio y adquieren las competencias personales y técnicas necesarias para su desarrollo y sostenibilidad.

El programa culmina con una competición interuniversitaria a nivel nacional cuyo ganador representa a España en la competición europea Junior Achievement – Young Enterprise Europe Enterprise Challenge.

En este programa CtOS Enabler ha conseguido ganar la semifinal universitaria, representando a la Universidad Complutense en la final nacional, que tuvo lugar el 20 de mayo de 2015, y en esta final consiguió tres de los principales premios que se daban.

Por un lado, CtOS Enabler ganó el primer premio de la competición nacional, lo que le lleva a representar a España en la final europea de este mismo concurso en Lisboa a principios de julio de 2015.

Además de ese premio, CtOS Enabler ha ganado una dotación económica y sesiones de asesoramiento por parte de PwC, que premia al proyecto con más potencial de crecimiento en mercados globales. Para ello, la Fundación PwC valora la estrategia del mismo, su adaptación para entornos multinacionales, las capacidades del equipo y su capacidad de crecimiento, y también el premio que daba la fundación Rafael del Pino, que premia al mejor proyecto tecnológico que destaque por su excelencia. Además, becará a los promotores del proyecto para participar en el Programa Encuentro TR35 Week, que tendrá Iugar en el MIT (Massachusetts Institute of Technology) en el mes de octubre.

Global Urban Datafest

Figura AII.3. Global Urban Datafest

Hackathon organizado por la Universidad Complutense junto con IBM y Optiva Media entre otras, con una duración de un día en el que había que desarrollar una aplicación relacionada con las *smart cities*.

Para este hackathon se desarrolló, usando CtOS Enabler como framework de geolocalización, la aplicación de Smart Lights explicada en el **Capítulo 6. Aplicaciones como casos de uso**.

La aplicación ganó el premio que otorgaba IBM en dicho evento, 12.000$ en infraestructura Cloud de IBM.

Traity Weekend of Code

Figura AII.4. Traity Weekend of Code

Hackathon organizado por el empresa Traity, en el que había que programar algo funcional en el plazo de un fin de semana.

Puesto que la temática era libre, se optó por utilizar CtOS Enabler para crear una aplicación web de reputación de zonas geográficas. La aplicación es Georep y está descrita en el **Capítulo 6. Aplicaciones como casos de uso**.

Con dicha aplicación se ganó el premio al mejor proyecto del Hackathon, que incluía una pequeña dotación económica en BitCoins, un Dron y una Raspberry Pi para cada participante.

Anexo III. Dossier de prensa

A raíz de la participación en Startup Programme, CtOS Enabler ha aparecido en diferentes medios de comunicación. Algunos de los enlaces a las noticias y entrevistas se exponen a continuación:

- Post en el blog de Startup Programme: https://startupprogramme.wordpress.com/2015/05/21/cosechando-premios/
- Notas de prensa:
  - RedEmprendia: http://www.redemprendia.org/es/actualidad/noticias/startup-programme-premia-el-proyecto-ctos-enabler-desarrollado-por-tres-alumnos-de-la-universidad-complutense-de-madrid
  - El economista: http://ecoaula.eleconomista.es/ecoaula-emprendedores/noticias/6728210/05/15/Ctos-Enabler-elegido-el-mejor-proyecto-emprendedor-universitario.html
  - Te Interesa: http://www.teinteresa.es/espana/ESTUDIANTES-REPRESENTARAN-COMPETICION-EUROPEA-ACHIEVEMENT_0_1361264340.html
  - La información: http://noticias.lainformacion.com/educacion/estudiantes/estudiantes-de-la-ucm-representaran-a-espana-en-la-competicion-europea-junior-achievement_fbV09z2bc2ddG9bHjDR8m3/
  - Europa Press: http://www.europapress.es/campusvivo/actualidad-universitaria/noticia-estudiantes-ucm-representaran-espana-enterprise-challenge-2015-20150521150458.html
  - Ingenieros: http://www.ingenieros.es/noticias/ver/ctos-enabler-elegido-mejor-proyecto-emprendedor-universitario-de-la-final-de-startup-programme/5475
  - El Referente: http://www.elreferente.es/tecnologicos/startup-programme-anuncia-sus-proyectos-ganadores-28508
  - Servimedia: http://www.servimedia.es/Noticias/Detalle.aspx?n=453245&s=23

- Discapnet: http://www.discapnet.es/Castellano/Actualidad/Linea_Social/estudiantes-de-la-ucm-representaran-a-espanha-en-la-competicion-europea-junior-achievement.aspx
- Diario Siglo XXI: http://www.diariosigloxxi.com/texto-s/mostrar/161571/estudiantes-de-la-ucm-representaran-a-espana-en-la-competicion-europea-junior-achievement#.VV9N0HMU6ko
- Emes: http://www.emes.es/Actualidad/Noticias/Noticia/ctos/enabler/fue/tabid/581/itemid/5200/type/noticia/Default.aspx?utm_source=&utm_medium=RSS&utm_campaign=

- Entrevistas en radio:
  - Entrevista en COPE en el programa La Linterna: http://www.cope.es/player/Los-creadores-de-City-os-Eneibler-en-La-Linterna&id=2015052121590002&activo=10 http://vod.cope.es/audio/2015/05/21/audio_1432247235574511.mp3
  - Entrevista en Radio 5 (RNE) en el programa Entre Paréntesis (a partir del minuto 29): http://www.rtve.es/alacarta/audios/entre-parentesis/entre-parentesis-28-05-15/3145802/
  - Entrevista en RNE en el programa 24 Horas (minuto 26:30): http://www.rtve.es/alacarta/audios/24-horas/24-horas-tertulia-tematica-02-06-15/3152741/

- Entrevistas en revistas:
  - Revista Emprendedores: http://www.emprendedores.es/ideas-de-negocio/ctos-enabler

Glosario de términos

Cloud Computing

CtOS Enabler se despliega en la nube, por lo tanto, la infraestructura necesaria para dar soporte al servicio se localiza en un entorno de cloud computing, y se evaluarán soluciones tanto IaaS como PaaS para ver la que mejor se adapta al servicio.

La elección de las tecnologías de computación cloud como infraestructura de soporte, permiten la provisión de un servicio **escalable, dinámico, elástico y bajo demanda de recursos**, pudiendo así responder en tiempo real a situaciones con necesidades computacionales cambiantes en el tiempo y con un **coste óptimo**.

CtOS Enabler pone a disposición de terceros un servicio enmarcado en el modelo de **Plataforma como Servicio** (PaaS), que permite a los desarrolladores eliminar la necesidad de implementar ciertas características en las aplicaciones, proporcionando su uso a través de Internet.

Los diferentes modelos de servicios que se dan dentro del ámbito del Cloud Computing son: IaaS, PaaS y SaaS.

IaaS

El modelo más básico de *cloud* que conforma la capa de primer nivel o inferior y, por lo tanto, sirve de base para los modelos PaaS y SaaS, es el **IaaS** (“Infrastructure as a Service”) o **infraestructura como servicio**, cuya misión es ofrecer, como si de un suministro se tratara, capacidad de cómputo, memoria, almacenamiento persistente y conexiones de red. También son conocidos como servidores de hosting en la nube.

PaaS

La capa intermedia dentro de los modelos de servicio en el cloud es **PaaS** ("Platform as a Service") o **plataforma como servicio**, que se despliega sobre una infraestructura cloud, y permite facilitar el despliegue de aplicaciones software sin incurrir en el coste y en la complejidad de comprar y administrar el hardware, el sistema operativo o el aprovisionamiento de capacidades de almacenamiento, porque al usuario se le ofrecen una serie de características y herramientas preinstaladas sin conocer el sistema a bajo nivel.

SaaS

En la última capa, la que se encuentra por encima de IaaS y PaaS, se encuentra el modelo **SaaS** ("Software as a Service") o **software como servicio** que se caracteriza por ofrecer una aplicación completa y funcional para sus clientes, que se ejecuta sobre la infraestructura del proveedor *cloud*, y que es accesible por medio de un navegador web, lo que evita costes de soporte y almacenamiento local tanto software como hardware para los clientes de dichas aplicaciones.

API

Una **API** ("Application Programming Interface") es un conjunto de funciones y procedimientos, ya implementados, que proporcionan a otro software una serie de funcionalidades y características adicionales, sin necesidad de volver a ser implementadas, y que son transparentes para el usuario, logrando así una capa de abstracción.

Una API tiene la función de servir como interfaz entre dos programas diferentes que necesitan comunicarse entre ellos, o bien, uno de los programas necesita obtener información de otro, por ejemplo: con una base de datos, con el sistema operativo o con alguna red social, a través de Internet.

Geolocalización

El término **geolocalización** es un concepto relativamente nuevo, que ha surgido con la creación de dispositivos y aplicaciones móviles cada vez más complejos y tecnológicos, y se puede definir como la **posición geográfica de un dispositivo** dentro de un sistema de coordenadas determinado.

La **geolocalización de un móvil** se puede determinar por el **sistema de GPS**, por **localización de IP** y/o por **triangulación de las señales de radio** del dispositivo **entre varias torres de la red telefónica**, y puede obtenerse por la proximidad a la torres de telefonía, por el tiempo que tarda la señal en ir de torre a torre y por la fuerza de la señal recibida.

Framework

La palabra inglesa "**framework**" (marco de trabajo) define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de

problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

En el ámbito de la informática, un ***framework*** o ***infraestructura digital***, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, que puede servir de base para la organización y desarrollo de *software*. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Zona de influencia

El concepto de zona de influencia, dentro de la API, hace referencia a un área geográfica que el administrador de una aplicación considera de interés, ya sea porque quiere insertar datos en esa zona o porque quiere explotar los datos de los usuarios de su aplicación que accedan a dicha zona. Las zonas de influencia están definidas como figuras geométricas situadas en el mundo real mediante el uso de coordenadas o direcciones postales.

JSON

JSON (de sus siglas en inglés *JavaScript Object Notation*) es un formato ligero para el intercambio de datos estructurados, que surgió como alternativa de XML, y que actualmente es uno de los más utilizados por su simplicidad y sencillez a la hora de generar información estructurada.

GeoJSON

GeoJSON surge como un formato de intercambio de datos geoespaciales basado en JSON. Los objetos GeoJSON tienen la misma estructura principal que un objeto JSON, pero tienen características especiales para definir las figuras geométricas contenidas en su interior. GeoJSON permite codificar diferentes estructuras geográficas en función de la figura geométrica deseada.