Trabajo de Fin de Grado

Grado en Ingeniería Informática

InForestal

InForestal

Aitor Bernal Falcón

La Laguna, 2 de Julio de 2017

D. **Dagoberto Castellanos Nieves**, con N.I.F. 79234766-L profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de sistema de la Universidad de La Laguna, como tutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

“InForestal”

ha sido realizada bajo su dirección por D. **Aitor Bernal Falcón**, con N.I.F. 78729165-G.

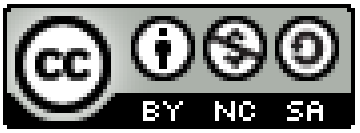
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 2 de Julio de 2017 .

Agradecimientos

A mi familia por el apoyo y su confianza en mi.

A mi tutor Dagoberto Castellanos Nieves, sin su dedicación y ayuda este trabajo de fin de grado no se hubiese llevado a cabo.

Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.

Resumen

El objetivo de este trabajo ha sido crear una plataforma capaz de satisfacer una necesidad real en la población Canaria, como es ayudar a acabar con la desinformación referente a senderos, parques recreativos y parques infantiles. Para ello se ha desarrollado una plataforma web denominada Inforestal capaz de centralizar los datos permitiendo así al usuario obtener más información de estos gracias a la disposición en la que se presentan.

Para el desarrollo de esta plataforma se ha hecho uso del aprendizaje a lo largo del Grado, aplicando conocimientos adquiridos en asignaturas como "Tecnología de la Información para las organizaciones", "Desarrollo de Sistemas informáticos", "Sistemas y tecnologías web", "Gestión del conocimiento" y "Sistemas de interacción persona computador" entre otros.

Como resultado se ha obtenido una plataforma web cuya misión es, a partir de los datos generar una información útil y lo más actualizada posible gracias a los mismos usuarios ya que la plataforma actúa a su vez como red social.

Palabras clave: Gestión del conocimiento, Turismo, Senderos, Parques.

Abstract

The objective of this work has been to create a platform capable of satisfying a real need in the Canaria population, such as helping to end the disinformation related to trails, recreational parks and playgrounds. For this, a web platform called Inforestal has been developed capable of centralizing the data, thus allowing the user to obtain more information from these thanks to the arrangement in which they are presented.

For the development of this platform has been made use of learning throughout the Degree, applying knowledge acquired in subjects such as "Information Technology for organizations", "Development of computer systems", "Systems and web technologies", "Management Of the knowledge "and" Systems of interaction person computer "among others.

As a result we have obtained a web platform whose mission is, from the data to generate useful information and the most updated possible thanks to the same users since the platform acts as a social network as well.

Keywords: Knowledge Management, Tourism, Trails, Parks.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introductorio

1.1 Introducción

A diario recibimos por distintos medios infinidad de datos referentes a distintos campos y temáticas, estos datos provienen de las noticias, de las escuelas e incluso en el momento del café con nuestro compañero de trabajo. Pero nos puede asaltar la duda si tanta información nos es útil y en caso de serlo, ¿Hasta que punto?.

Como respuesta a la pregunta anterior nace lo que hoy por hoy se conoce como la cultura de la "Gestión del conocimiento", dicha cultura es aplicada principalmente en las organizaciones. Tiene como fin transferir el conocimiento desde el lugar donde se genera hasta el lugar donde se va a emplear. Este conocimiento del que hablamos se genera tras hacer un uso correcto de los datos, ya que el conocimiento es una mezcla de experiencia, valores, información y "know-how" que sirve como marco para la incorporación de nuevas experiencias e información. El conocimiento va más allá de documentos, libros. También forma parte del conocimiento las normas, las experiencias y las prácticas.

A día de hoy son varias las aplicaciones que se centran en convertir estos datos en conocimiento o información, gracias a los propios usuarios de la plataforma que comparten sus experiencias, conocimientos y valores. Ejemplos claros son las empresas Airbnb, BlaBlaCar, Cabify.

El objetivo que se tiene con la plataforma "InfoRestal" es poder compartir conocimiento y que actúe como red social, ya que gracias a los usuarios podremos saber con mayor exactitud información de áreas recreativas, senderos y parques infantiles para poder tomar la decisión que más nos satisfaga o creamos más conveniente tanto para los habitantes de las islas como para aquellas personas que vienen de vacaciones a dicho paraje natural.

En Inforestal podremos encontrar información relevante sobre parques, zonas recreativas (longitud, sus clasificaciones, microclima de la zona, recomendaciones de otros usuarios, valoraciones, disponibilidad de recursos,etc) partiendo de los datos recopilados desde diversas fuentes como el Cabildo de Tenerife [1], el Gobierno de Canarias sobre la isla de Tenerife [2] y otras fuentes abiertas como es el caso de Open Data Canarias [3]. Posteriormente, la comunidad de usuarios será la que se encargará de gestionar esta información pudiendo valorar, comentar, crear incidencias y modificar los datos con el objetivo de tener una mayor exactitud.

1.2 Antecedentes y estado del arte

Las aplicaciones basadas en la gestión del conocimiento son algo con pocos años de historia, nacidas en la última década, si bien es cierto que han crecido de forma exponencial con el paso de los años.

Actualmente no existe ninguna plataforma enfocada al mismo campo para Canarias, es decir en el campo del medio ambiente y el ocio infantil (áreas recreativas, parques y senderos). Si es cierto que existen plataformas similares enfocadas a otros sectores, alguna de las cuales vamos a enumerar a continuación:

- Guachapp: Esta aplicación está destinada para dispositivos móviles que permite encontrar guachinches y locales de comida típica Canaria. Guachapp nace con el motivo de solucionar la problemática de encontrar lugares de calidad de comida típica Canaria. La aplicación es capaz de geolocalizar los restaurantes en un mapa, mostrar una ficha con toda su información (especialidades y su votación media generada gracias a los usuarios además de ubicación y críticas [4].

- Shorcial: Esta aplicación trabaja en el ámbito del turismo y está destinada principalmente para dispositivos móviles. Shorcial nos permite realizar las siguientes tareas:[5]
    - Encontrar playas cercanas.
    - Filtrar playas.
    - Subir fotos de playas concretas.
    - Ver información concreta (Nombre, temperatura, localización, ver en directo, bandera, limpieza, disponibilidad de hamacas…).
    - Comentar y valorar playas.
    - Añadir nuevas playas.

- ViaULL: Desarrollada por la empresa StartDevs con el objetivo de facilitar el uso compartido de automóviles particulares entre la comunidad universitaria. La aplicación solicita datos como el consumo del coche para poder calcular el consumo estimado del trayecto y así poder dividir los gastos entre los pasajeros. También podemos determinar otras características como es si en el coche se permiten fumadores o mascotas. Posee aplicación tanto para web como para movil [6].

- Airbnb: Esta plataforma tanto web como móvil es uno de los casos de éxito más populares de la gestión del conocimiento. Airbnb es una red social en la que personas ponen sus casas a disposición de los viajeros por el precio que cada uno estime oportuno, encargándose la web de gestionar cobro y pago. Airbnb nos permite comentar y valorar los alojamientos para permitir a futuros huéspedes saber concretamente todas las ventajas y desventajas del sitio en el que se quieren hospedar [7].

- SmartCitizen: Esta plataforma web consiste en un mapa a nivel mundial que se completa con datos facilitados por los usuarios.Estos datos son recogidos mediante unas placas programables similares a Arduino con conectividad a internet. El usuario final se encarga de adquirir estas placas y enviar datos a los servidores de SmartCitizen, este los recaba y los muestra en un mapa, así podremos saber de forma detallada las condiciones climatológicas de las zonas donde se encuentre algún dispositivo instalado [8].

1.3 Objetivos

Los objetivos concretos de este proyecto son los siguientes:

- (OB1). Estudio del estado del arte de los sistemas de gestión del conocimiento orientados a senderos, parques y áreas recreativas.
- (OB2). Búsqueda, análisis e integración de datos abiertos (Open Data) en el ámbito de la aplicación.
- (OB3). Analizar, diseñar e implementar el módulo de datos geográficos.
- (OB4). Analizar, diseñar e implementar el módulo social.
- (OB5). Integración de los datos geográficos y sociales.
- (OB6). Analizar, diseñar e implementar el módulo para dispositivos móviles.
- (OB7). Creación de documentación y desarrollo de la memoria final.

Capítulo 2. Tendencias tecnológicas

En el siguiente capítulo vamos a proceder con la presentación de las diversas tecnologías relevantes para el desarrollo de la aplicación, así como, sus ventajas y desventajas.

2.1 Plataformas relevantes

Actualmente elegir la tecnología web y el lenguaje de programación es bastante complicado. Esto se debe a que las principales páginas web de uso diario por gran parte de los usuarios están programadas con diversas tecnologías y lenguajes, vamos a mencionar las principales redes sociales y webs y las tecnologías que usan:

- **Instagram:** Instagram fue desarrollado en Python usando el framework Django, también hace uso de NGINX y GUNICORN para el balanceo de carga, soporte HTTP y HTTPS. En lo que a Bases de datos se refiere hace uso de PostgreSQL.[1]
- **Facebook:** Por otro lado, Facebook hace uso de las tecnologías web, en lo que a lenguaje de programación se refiere, un poco peculiar ya que el lenguaje que usa es PHP, pero a esto le añade luego que hace uso de un intérprete llamado HipHop que pasa el código en PHP a C++.Por otro lado, en lo que a Bases de datos se refiere, usan un modelo no relacional como es NoSQL.[2]
- **Airbnb:** La empresa de alquiler de viviendas de forma social Airbnb usa como tecnología web React.js que tiene como dueño a Facebook. React es una librería Javascript basada en Node.js creada para construir interfaces de usuario, que te permiten crear aplicaciones SPA(single page application) más eficientes y funciona tanto en el lado cliente como en el servidor, haciendo posible la creación de aplicaciones isomórficas (podemos ejecutar el mismo código tanto en el cliente como en el servidor).

Como podemos comprobar, las principales web tecnológicas de las que se hace uso hoy por hoy son muy diversas, en lo que a tecnologías respecta.

2.2 Lenguajes y frameworks relevantes

A la hora de decidir una tecnología concreta para inforestal ha sido un proceso bastante complejo porque cada tecnología y lenguaje tiene sus pros y

contras, por ello vamos a detallar los pro y contras que se han encontrado en las distintas tecnologías:

1. HTML: Es un lenguaje estático de marcas usado para la elaboración de páginas web [3].
    a. Sintaxis:
    ```
    <html>
        <head>
            Hola soy la cabeza
        </head>
        <body>
            Hola soy el cuerpo
        </body>
    </html>
    ```
    b. Ventajas:
    Uso sencillo y fácil aprendizaje.
    Representación estructurada.
    Archivos pequeños.
    Está estandarizado, lo permiten todos los navegadores.
    c. Desventajas:
    Lenguaje estático.
    Dificultad de corrección.
    Diseño lento.
    Etiquetas limitadas.

2. Javascript: Es un lenguaje interpretado, no requiere compilación. Se usa para el desarrollo de páginas webs, para evitar incompatibilidades se establece el WC3 que establece un diseño estándar denominado DOM [4].
    a. Sintaxis:
    `<script type="text/javascript"> ... </script>`
    b. Ventajas:
    Los script tienen capacidades limitadas por cuestiones de seguridad.
    Se ejecuta el código en el cliente
    c. Desventajas:
    Código visible, editable y descargable.

3. PHP: Es un lenguaje de programación utilizado para la creación de sitios web,se ejecuta en el servidor y este genera código html, permitiendo así páginas webs dinámicas [5].
    a. Sintaxis:
    ```
    <?
    $mensaje = “Hola”;
    echo $mensaje;
    ?>
    ```
    b. Ventajas:
    Fácil de aprender.
    Lenguaje rápido
    Soporta POO
    Multiplataforma.

Soporta módulos.
Seguridad.

c. Desventajas:
Necesidad de tenerlo instalado en servidor web.
No delega en el cliente.
Legibilidad compleja al mezclar PHP con HTML

4. ASP: Dicha tecnología está del lado del servidor, y fue desarrollada por Microsoft para poder desarrollar páginas web dinámicas [6].
   a. Sintaxis:
   <% %>
   b. Ventajas:
   Usa Visual Basic siendo esto más fácil para el usuario.
   Buena comunicación con SQL Server.
   c. Desventajas:
   Desorden.
   Se necesita mucho código para funciones sencillas.
   El hospedaje es costoso.

5. ASP.NET: El ASP.NET intenta solucionar los problemas que se presentan en ASP, para el desarrollo de ASP.NET se puede utilizar C#,VB.NET o J#.
   a. Ventajas:
   Orientado a objetos.
   División entre diseño,aplicación y código.
   Buen manejo de aplicaciones grandes.
   Mayor velocidad y seguridad.
   b. Desventajas:
   Alto consumo de recursos.
6. Python: Este lenguaje es interpretado, es multiparadigma y permite programar con cierta claridad para el programador.
   a. Sintaxis:
   ```
   def suma(a,b):
                return a+b;
   ```
   b. Ventajas:
   Gran cantidad de librerías.
   Multiplataforma
   Sencillo
   Orientado a objetos
   Portable
   c. Desventajas:
   Lentitud al ser un lenguaje interpretado.
7. Ruby:Es un lenguaje interpretado a alto nivel, es dinámico y nos permite programar orientado a objetos de una forma sencilla.
   a. Sintaxis:
   puts “hola”
   b. Ventajas:
   Permite desarrollar a bajo costo
   Software libre
   Multiplataforma
   Manejo de excepciones

Portátil

c. Desventajas:
No está muy extendido, poca comunidad de desarrollo.

Tras analizar las tecnologías más usadas, finalmente se trabajará con node.js, los motivos han sido los siguientes.

1. Node.js: Es un entorno Javascript del lado del servidor, utiliza un model asíncrono y dirigido a eventos.
   a. Ventajas:
   Podemos utilizar Javascript como lenguaje de scripting desde nuestra consola.
   Se basa en eventos, por lo que podemos hacer uso de la filosofía que se usa en AJAX desde el servidor.
   Nos permite usar mismo lenguaje tanto en cliente como servidor.
   Muy buena gestión de paquetes gracias a NPM.
   Muy buena comunidad de desarrolladores.
   Podemos hacer desde el servidor todo lo que necesitemos.

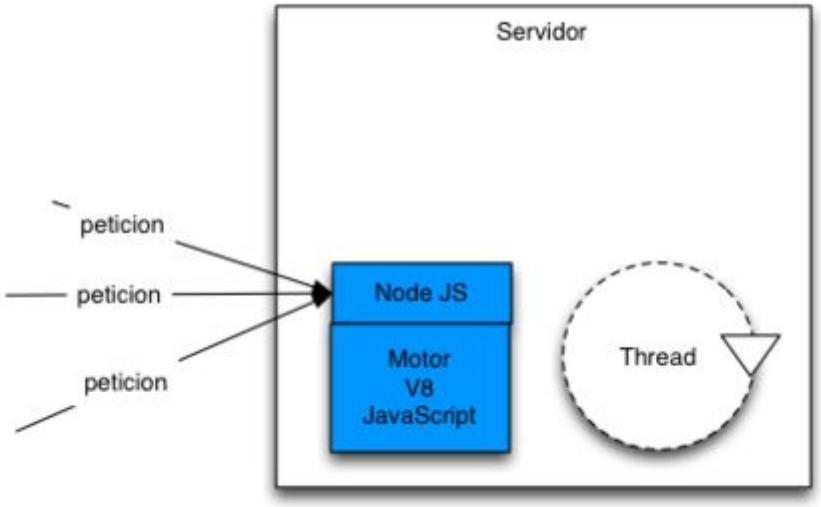


Figura 1: Funcionamiento de Node.JS [9]

2. Motor de vistas EJS: EJS es el motor de vistas que vamos a usar, desde el back-end desarrollado con node.js enviaremos los datos para que el motor de vistas se encargue de renderizar los datos sobre los templates.
   a. Ventajas:
   Las páginas de nuestra aplicación no serán estáticas, dependerán de los datos enviados desde el servidor.
3. Express.js: Express.js es una infraestructura de aplicaciones web sobre Node.js flexible que proporciona un conjunto de características, es decir, nos facilita la conección entre front-end y back-end mediante HTTP. Otra de las características que nos proporciona Express.js es la de poder hacer uso de "middleware" que son funciones que tienen acceso al objeto de solicitud (req) , de respuesta (res) pudiendo enviar contenido a los mismos.

2.3 Sistemas de Bases de Datos relevantes

Las Bases de Datos pueden ser clasificadas según la variabilidad de los datos en: estáticas y dinámicas.

1. Bases de Datos estáticas: Estas bases de datos son utilizadas principalmente para datos históricos o datos que no se van a modificar con el tiempo y van a ser solo de lectura.

2. Bases de datos dinámicas:Este tipo de base de datos permite modificar la información con el tiempo, a diferencia de las Bases de datos estáticas, en estas podemos modificar el contenido.

Según el modelo de administración de los datos, las bases de datos pueden ser de los siguientes tipos:

1. Bases de datos jerárquicas: Esta base de datos almacena los datos en una estructura jerárquica. Estas bases de datos disponen de nodos padres y nodos hijos, permitiendo así compartir gran parte de los datos.
    a. Ventajas: Buen manejo de alto volumen de datos.
    b. Desventajas: Incapacidad de representar eficientemente la redundancia.
2. Bases de datos transaccionales: Son bases de datos que tienen como fin el envío y recepción de datos a grandes velocidades. Estas bases de datos son muy poco comunes y están destinadas al entorno de análisis de calidad, datos de producción e industrial. No debemos olvidar que su principal objetivo es obtener y enviar los datos con la mayor brevedad posible, por ello no es problema la redundancia y la duplicación de los datos.
3. Bases de datos relacionales: Éste es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas".

    a. Ventajas:
    Favorece la normalización
    Garantiza la integridad referencial, así, al eliminar un registro elimina todos los registros relacionados dependientes.
    Garantiza la integridad referencial, así, al eliminar un registro elimina todos los registros relacionados dependientes.
    b. Desventajas:
    Presentan deficiencias con datos gráficos, multimedia.
    No se manipulan de forma manejable los bloques de texto como tipo de dato.
    No se suelen usar para redes sociales.
4. Bases De Datos Orientadas a Objetos:Este modelo, bastante reciente, y propio de los modelos informáticos enfocado a objetos, trata de

almacenar en la base de datos los objetos completos (estado y comportamiento).

a. Ventajas:
   Mayor capacidad de modelado.
   Ampliabilidad
   Lenguaje de consulta más expresivo.
   Adecuación a las aplicaciones avanzadas de base de datos.
   Mayores prestaciones.
b. Desventajas:
   Mayores prestaciones.
   Carencia de experiencia.
   Carencia de estándares.
   La optimización de consultas compromete la encapsulación.

5. NoSQL: Las Bases de datos no relacionales de alto desempeño. Las bases de datos NoSQL utilizan varios modelos de datos, incluidos los de documentos, gráficos, claves-valores y columnas. Las bases de datos NoSQL son famosas por la facilidad de desarrollo, el desempeño escalable, la alta disponibilidad y la resiliencia. A continuación, presentamos varios recursos que le servirán.

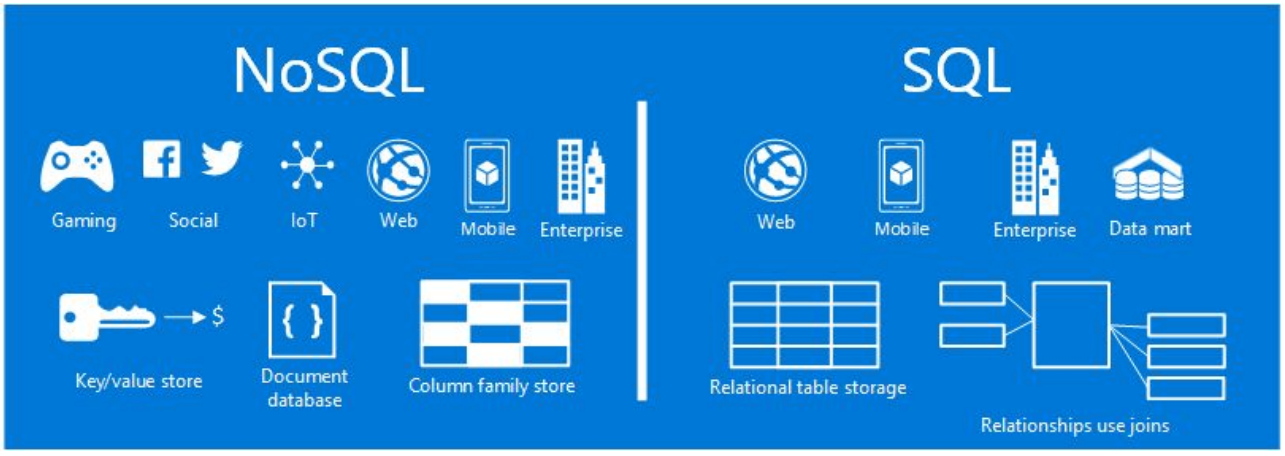


Figura 2: Imagen comparativa SQL frente a NoSQL [10]

Entre los modelos mencionados anteriormente nos hemos decantado por "NoSQL" aunque hemos dudado ya que el Modelo relacional podría estar a la altura dando un buen rendimiento para nuestra aplicación (Figura 2).
A la hora de decidirnos por NoSQL hemos obtenido una serie de usos que nos han hecho decantarnos por este modelo frente al modelo relacional. Los usos mencionados son los siguientes:

1. Usos del Modelo relacional:
   a. Educación: para estructurar información, y aportar conocimiento lógico al estudiante.
   b. Desarrollos web: para mantener jerarquía de datos, siempre y cuando la capacidad de concurrencia, almacenamiento y mantenimiento no sean de considerable dificultad y la información sea consistente.
   c. Negocios: inteligencia y análisis de negocios, son temas que requieren el uso de SQL para facilitar el consumo de la información

y la identificación de patrones en los datos.
   d. Empresarial: porque tanto el software a la medida y el software empresarial, poseen la característica de mantener información con estructura consistente.
2. Usos de NoSQL:
   a. Redes sociales: Este es el principal argumento a la hora de decantarnos por NoSQL.
   b. Desarrollo Web: debido a la poca uniformidad de la información que se encuentra en Internet; aun cuando también puede emplearse SQL.
   c. BigData: debido a la administración de grandísimas cantidades de información y su evidente heterogeneidad.

Como sistema gestor de bases de datos NoSQL usaremos MongoDB ya que es el más extendido en la comunidad de desarrolladores.

2.4 Sistemas de cartografía relevantes

Con respecto a la forma de visualizar los datos, haremos uso de mapas con ubicaciones geolocalizadas en la cual colocaremos cada uno de las áreas recreativas, senderos y parques, cada uno con un icono distinto para que sea fácilmente distinguible a simple vista en el mapa. El mapa elegido ha sido un proceso complicado ya que se han presentado diversas alternativas con sus ventajas y desventajas.

Las alternativas trabajadas han sido las siguientes:
OpenStreetMaps:

a. Ventajas:
   Software libre.
   Ofrece Api.
   Plataforma en constante cambio y crecimiento ya que su trabajo es de forma colectiva. A continuación una gráfica con el crecimiento de OpenStreetMaps siendo azul la curva de nuevos usuarios y morada la curva de puntos nuevos registrados.

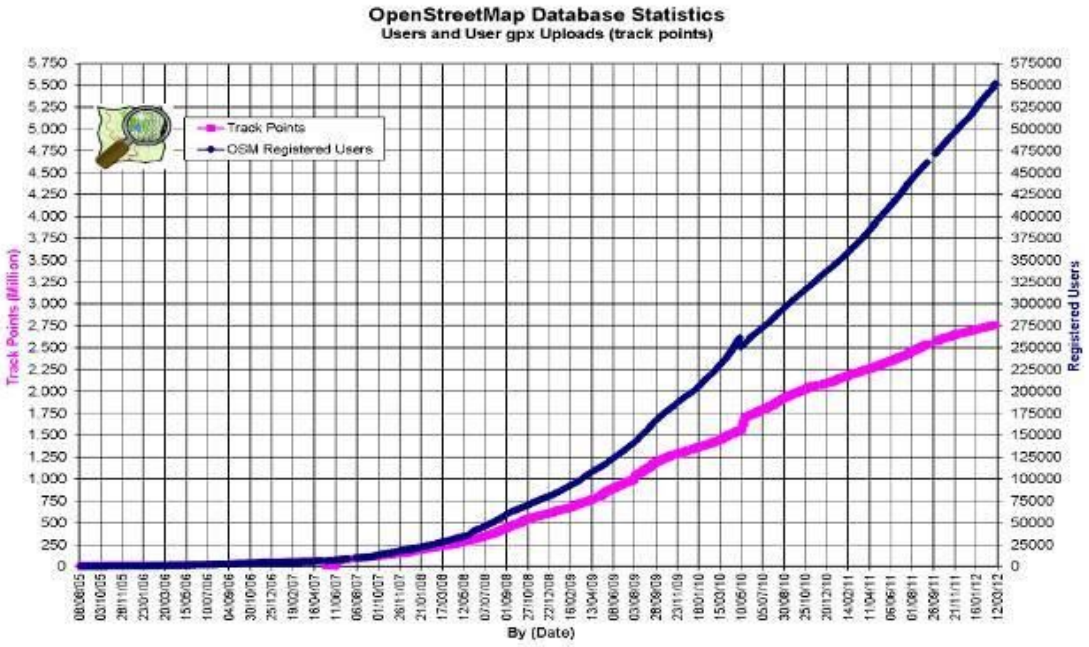


Figura 3: Crecimiento OpenStreet Map [11]

2. Desventajas:
    a. Al igual que hay zonas muy completas, también existen vacíos importantes que se solucionan con Bing o Microsoft basándose en imágenes de satélite.
    b. Dificultad de la API.

Google Maps:

a. Ventajas:

Mapas más concretos.
Mapas con información más allá de únicamente geolocalizar puntos.
Disponen de Api.
Buena documentación.
Opción StreetView

b. Desventajas

API parcialmente de pago.

Finalmente, la forma de trabajar con los mapas ha sido mediante Google Maps, el motivo principal ha sido que nos ofrece información extra más allá de únicamente la geolocalización ya que nos ofrece información detallada sobre bares y puntos de interés, además de poder usar una vista a nivel de calle, en la que podemos ver la zona como si estuviésemos en el sitio permitiéndonos entrar incluso a ciertos lugares cerrados.

A pesar de trabajar con Google Maps, en caso de escalabilidad y hacer más consultas de las que Google Maps nos permite hacer de forma gratuita,

podemos trabajar con OpenStreetMaps y Leaflet.

Leafleft es una librería open-source para crear mapas interactivos que se despliegan en una página web y nos permite trabajar con los mapas de una forma más amigable para el programador ya que ofrece una excelente API bien documentada. Leafleft es usada por plataformas como foursquare, CARTO y Wikimedia.

2.5 Modelo vista controlador

El modelo vista controlador, o comúnmente conocido como MVC es una arquitectura software que separa los datos y lógica de la interfaz de usuario.

El MVC fué introducido por Trygve Reenskaug en los años 70. En esta primera versión, el controlador se definía como "el encargado de la entrada" de tal manera, la vista era "la encargada de la salida". En la actualidad esta definición no tiene cabida ya que en la actualidad, el controlador ejerce un papel de sección intermedia de código que actúa de intermediario entre el modelo y la vista.

El siguiente esquema nos explica cómo se interactúa en el MVC desde el usuario hasta lo que se realiza en el servidor.

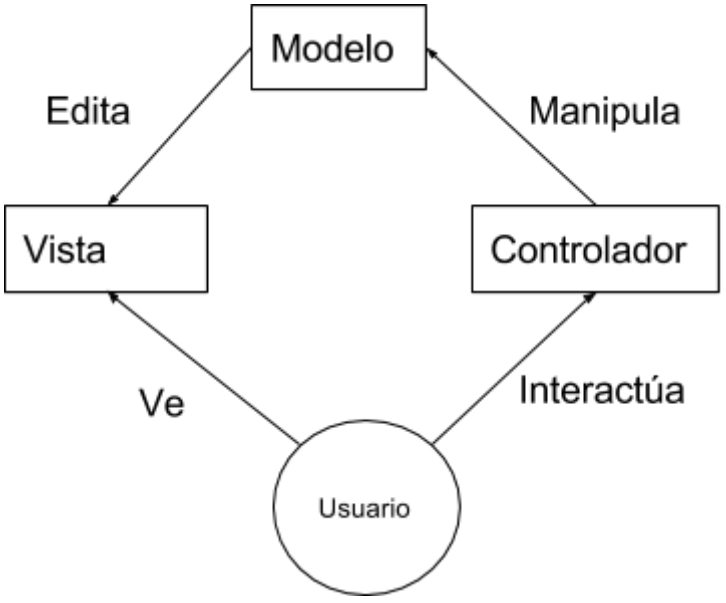


Figura 4: Esquema MVC

Las pautas seguidas en Inforestal, en lo que a estructura de directorio se refiere ha sido bastante similar a la que enuncia el MVC. Disponemos de una estructura de directorios similar al que indican el MVC formado por las siguientes carpetas (figura 5):

- Controllers: En esta carpeta se encontrarán todos los controladores de

nuestra aplicación, existe un fichero por modelo con las acciones necesarias para gestionar la base de datos además de un fichero maps.js que es el encargado de controlar el apartado del mapa en nuestra plataforma.

- Models: No haremos mucho hincapié en este apartado ya que fue expuesto en el apartado 2.2. Existe un fichero por cada uno de los ítems que vamos a registrar en nuestra plataforma (usuarios, senderos, parques recreativos y parques infantiles) además de otros modelos como el que se ha preparado para poder guardar los comentarios.
- Routes: En este directorio se encontrarán todos los ficheros que indican las rutas que vamos a permitir en nuestra aplicación express según la petición que realicemos (get, post, put, etc.).
- Views: En este directorio se encontrarán las vistas de las cuales hace uso nuestra plataforma. Las vistas de las cuales hace uso la plataforma son renderizadas por el motor de vistas EJS.

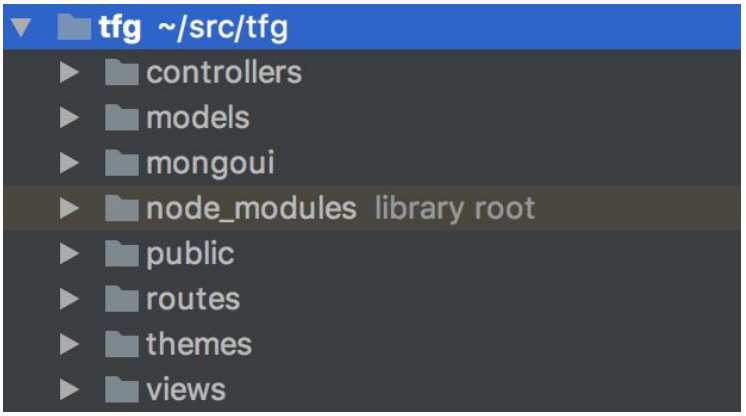


Figura 5: Estructura de directorios

Capítulo 3. Plataforma Inforestal

Para el desarrollo de la plataforma se ha elegido Node.JS gracias a la comparativa realizada previamente se ha llegado a la conclusión de la importancia que puede tener node para nuestro proyecto, para ello es bueno conocer un poco como es su funcionamiento para poder pasar a describir la plataforma .

Node.JS es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.JS usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, lo que lo hace mas liviano y eficiente. El ecosistema de paquetes de Node.JS, NPM, es el sistema de librerías de código abierto más grande del mundo.

Por otro lado, si nos ponemos a analizar qué problemas resuelve Node.JS, podemos partir de que una de las metas que se plantean es "proporcionar una manera fácil para construir programas de red escalables". Actualmente sistemas en lenguajes como PHP o Java, cada conexión genera un hilo que viene acompañado de 2 MB de memoria, esto dá un número máximo teórico de conexiones concurrente de cerca de 4000 usuarios. En caso de que queramos soportar un mayor número de clientes, necesitaremos agregar más servidores.
Node resuelve este problema cambiando la forma en que se realiza una conexión con el servidor.EN lugar de generar un nuevo hilo para cada conexión (con la memoria que le corresponde), cada conexión dispara un evento dentro del motor de Node y esto permite soportar miles de conexiones concurrentes.

Cabe destacar que Node.JS es un programa de servidor, pero es importante destacar que no es un producto como Apache o Tomcat. Esos tipos de servidores está preparados para ser instalados directamente y poder alojar nuestras aplicaciones de forma rápida. Sin embargo a modo de diferencia, en Node, tenemos que configurar y programar en lenguaje Javascript el servidor, las tareas que queremos realizar, acciones API REST (get, post, put…).

3.1 NPM

NPM(Node Package Manager) es un gestor de paquetes con más de 500.000 paquetes cuya misión es facilitar el trabajo al programador ya que gracias a él podemos tener cualquier librería disponible siempre que se encuentre dentro del repositorio de paquetes NPM. NPM también se encargará de ayudarnos a administrar nuestros módulos, distribuir paquetes y agregar dependencias de una manera sencilla.

Cuando instalamos nuevos paquetes lo que hace NPM es instalarlo de manera local en nuestro proyecto dentro de la carpeta node_modules, sin embargo, si queremos darle un carácter más amplio y poder usarlo en cualquier proyecto sin necesidad de instalarlo.

En la tabla 1 expondremos algunos de los paquetes NPM usados en nuestra plataforma.

Nombre	Versión	Descripción
bcrypt-nodejs	0.0.3	Este paquete se encarga de aplicar una función hash sobre nuestra contraseña.
express	4.15.2	Infraestructura de aplicaciones web que proporciona un conjunto sólido de características para las aplicaciones web.
express-session	1.15.3	Nos permite controlar las sesiones en express.
mongodb	2.2.26	Drivers oficiales de MongoDB para Node.js, ofrece una API para su manejo
mongoose	4.9.5	Mongoose nos permite trabajar con Mongo de una forma más sencilla para el usuario, nos ofrece validación, construcción de consultas, etc.
nodemon	1.11.0	Nodemon nos permite mantener el servidor continuamente funcionando sin tener que relanzar cada vez que hagamos un cambio en el servidor.

Tabla 1: Paquetes NPM instalados

3.2 Express.JS

Uno de los frameworks más utilizados en el framework Node.js es Express.JS, este está inspirado en los frameworks Connect y Sinatra, se caracteriza por ser simple, rápido y robusto.

Las principales tareas por las que Express.JS es tan importante en Node.JS es:

- Session: Gran manejo de la sesiones de los usuarios dentro de nuestra web o portal, las sesiones nos permiten mantener la información de un usuario de una página a otra.

- Middleware: Característica que nos permite realizar un enlace entre

aplicaciones software independientes, es decir que conecta dos aplicaciones y pasa los datos entre ellas.

- cookieParser:Esta característica nos permite trabajar con las cookies de una manera cómoda.

- bodyParser: Nos permite leer los objeto tipo "request" de una forma más amigable para el desarrollador.

Sin embargo, la característica insignia de express es el manejo de las rutas. Sin estas generadas, no podremos generar la interfaz para el usuario.

A continuación vamos a definir unos ejemplos de rutas empleados en el proyecto:

- **Peticiones GET**

    - Por ejemplo, en el fichero /routes/map.js podemos encontrar lo siguiente:

      api.get('/index', map.index);

      Luego si accedemos a la variable **map.index** que se encuentra en el fichero /controllers/maps.js podemos encontrar lo siguiente:

      exports.index = function (req, res) {
      res.render('map.ejs')
      }
      Lo que conseguimos con el código mencionado antes es que cuando un usuario acceda a la ruta "/index" el servidor reciba la petición y sea capaz de renderizar la vista "maps.ejs"

- **Peticiones POST**

    - Este tipo de ejemplo lo podemos encontrar en el fichero ubicado en /routes/user.js, el ejemplo es el siguiente:

        - **api.post**('/login',users.login);

      Luego si accedemos a la ruta users.login que se encuentra en el fichero /controllers/users.js podemos encontrar lo siguiente:

        - **exports.login** = function(req,res){
          Users.findOne({ 'nick': req.body.name},function(err,obj) {
          console.log("Ha encontrado un user");
          console.log(obj);

```
            if(bcrypt.compareSync(req.body.password,obj.password)){
                console.log("User y password correctos")
                res.redirect("/map/index")
            }
            else{
                res.render('bad_credentials.ejs');
                console.log("User y password incorrectos")
            }
        });


};
```

Con el código anterior conseguimos verificar si los datos enviados desde el formulario de login son ciertos, en caso de serlo, damos acceso al resto del contenido.

3.3 Vistas de la plataforma

Inforestal es una aplicación que nace de la falta de información y la validez que tiene esta información ya que mucha o no existe o está desactualizada. Para ello se le da un papel importante al usuario, el cual es el agente encargado de ir mejorando día a día estos datos y tenerlos lo más actualizados posible.

Inforestal está compuesto por las siguientes funcionalidades y páginas destacables que nos permitirán hacer un uso de la gestión del conocimiento.

- Login de usuarios: Los usuarios dispondrán de un registro compuesto por nombre de usuario, contraseña y email. Solo podrán acceder a la plataforma aquellos usuarios que estén previamente registrados y autenticados en la misma, guardaremos mediante sesiones que ese usuario se ha registrado correctamente y coincide con los datos que tenemos en la base de datos. En caso forzar la entrada en la plataforma sin estar logueado el sistema mediante un middleware en express se encargará de pedirle al usuario que se verifique su identidad. Esto se realiza de la siguiente manera:

```
let auth = function(req, res, next) {
                let aux = 0;

                Users.findOne({
                    'nick': req.session.user
                    }, function(err, obj) {
                        if ((obj == null) && (req.user == null)) {
                    console.log("No hay user en session");
                    aux = 0;
                        } else {
```

```
                    console.log("Hay user en session");
                    aux = 1;
                    }
                    if (aux == 0) {
                    return res.redirect('/');

                    } else {
                    return next();
                    }

                });

    };
```

- Página de inicio: Una vez autenticados en el servidor, accederemos directamente a la página de inicio en la cual encontraremos una vista general con todos los senderos, áreas recreativas y parques que se encuentran dentro de nuestra aplicación. Junto a estos iconos, una vez hacemos click, se muestra una información básica a modo resumen desde la cual podemos acceder a una información más detallada. En esta vista podemos acceder en el índice a una vista más filtrada mostrando solo lo que queremos ver.

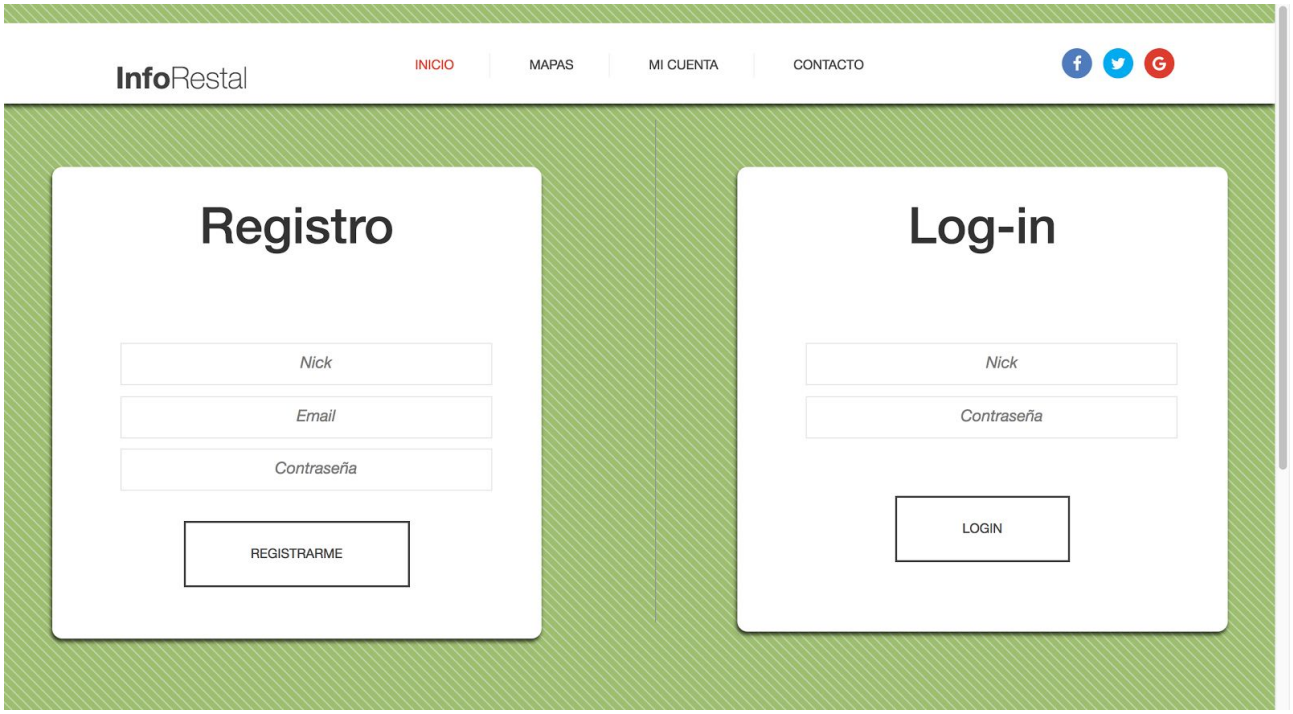


Figura 6: Vista para el acceso de los usuarios

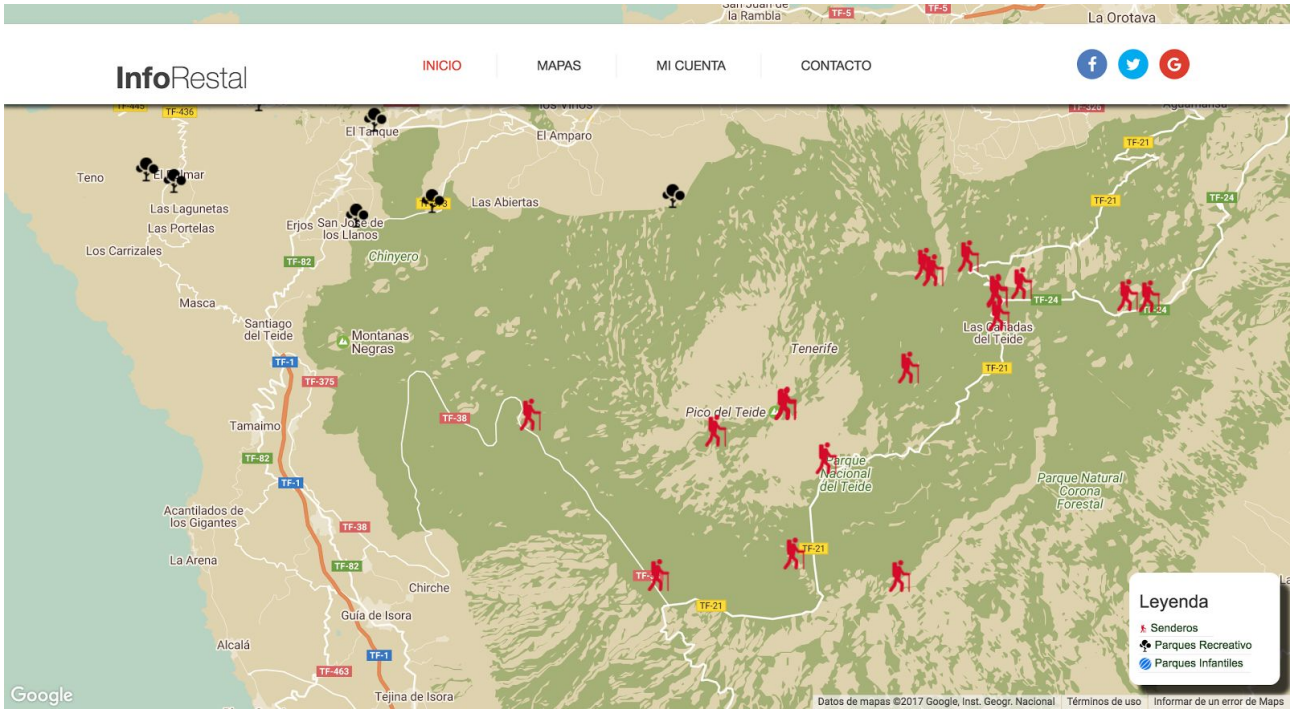


Figura 7: Vista para la página que muestra todos los items

- Página de item: Se encontrará toda la información referente a el item seleccionado, permitiéndonos la posibilidad de interactuar, ya sea valorando o dejando un comentario.

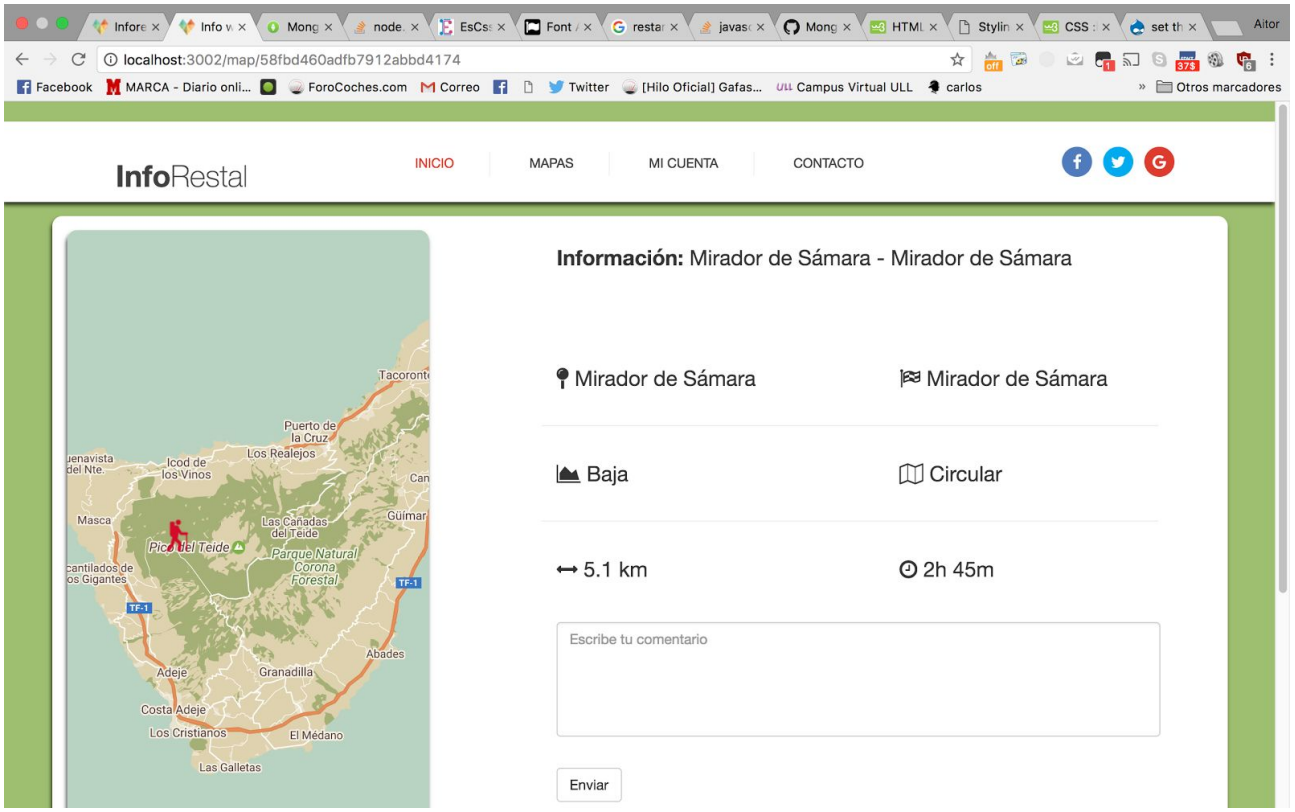


Figura 8: Vista que muestra la información del ítem senderos.

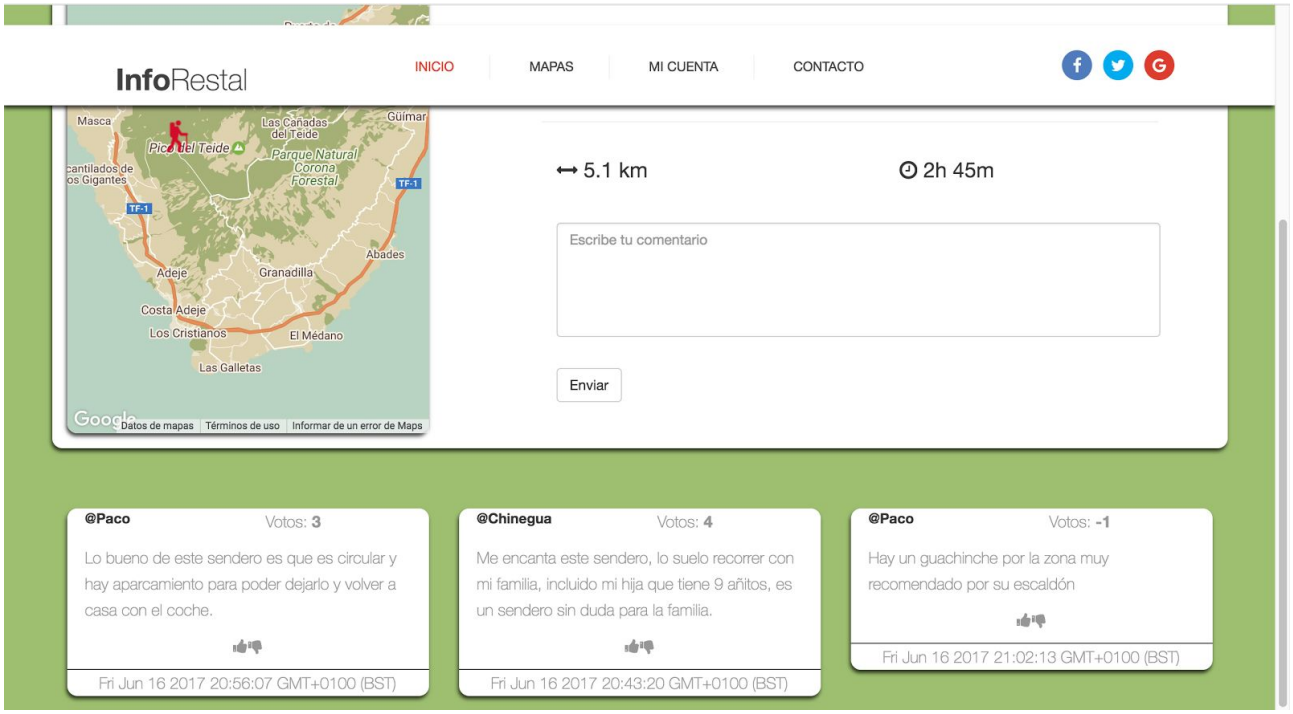


Figura 9: Vista que muestra la información de cada ítem con sus comentarios.

- Pestaña "Mi cuenta": Desde esta pestaña podremos acceder a la información vital del usuario, este puede editar campos vitales como son su nick, email o contraseña y también en esta pestaña se mostrará un historial de los cambios propuestos.

3.4 Base de datos empleadas

En el desarrollo de la plataforma Inforestal se ha usado la base de datos MongoDB que es de tipo NoSQL.

Para el desarrollo de la plataforma se ha hecho uso de una serie de modelos para permitir un correcto almacenamiento de la información necesaria para un correcto funcionamiento, estos modelos son los siguientes:

- Usuarios: Para el almacenamiento de los usuarios registrados en la plataforma se hará uso de un esquema con un nick, mail, password y status. En el campo status almacenaremos si es administrador o usuario normal. Podemos encontrar el código del modelo en el apéndice A1.

- Senderos: Por otro lado, también se ha creado un modelo para los senderos que tenemos almacenados en nuestra base de datos. Los campos de los que dispone son: {'partida', 'fin', 'dificultad', 'mapa',

‘distancia’, ‘duracion’, ‘latitud’, ‘longitud’}. Podemos encontrar el código del modelo en el apéndice A2.

- Parques recreativos: Para los parques recreativos, al igual que ocurre para el almacenamiento de los usuarios y los senderos, debemos de crear un modelo para poder almacenarlos en la base de datos. Dicho modelo dispone de los siguientes campos: {‘nombre’, ‘isla’, ‘municipio’, ‘acceso’, ‘transporte’, ‘contacto’, ‘permiso’, ‘superficie’, ‘minusvalido’, ‘agua’, ‘electricidad’, ‘duchas’, ‘bar’, ‘comedor’, ‘fogones’, ‘bancos’, ‘aparcamiento’, ‘latitud’, ‘longitud’}. Podemos encontrar el código del modelo en el apéndice A3.
- Parques infantiles: Los parques infantiles son otros de los items que vamos a poder encontrar en nuestra plataforma por lo tanto, debe tener un modelo para poder ser registrado en nuestra base de datos. El modelo presenta los siguientes campos: {‘suelo’, ‘situacion’,’nombre’, ‘superficie’, ‘latitud’, ‘longitud’}.
- Comentarios: Dentro de cada página de información de cada item los usuarios pueden comentar sobre este mismo, pudiendo además añadir un ‘Me gusta’ a los comentarios, para ello se ha creado un modelo con los siguientes atributos: {‘autor’, ‘sendero’, ‘comentario’, ‘fecha’, ‘votos’}. Este modelo tiene una peculiaridad ya que como podremos comprobar en el apéndice A4, hacemos referencia a otros modelos. En autor referenciamos a el modelo de usuario y en sendero referenciamos al sendero que se está comentando.

En el esquema presentado en la figura 9, se representa como relacionamos la colección Comentario con las colecciones de Usuario e Item añadiendo también otros valores como son ‘comentario’, ‘fecha’ y votos.

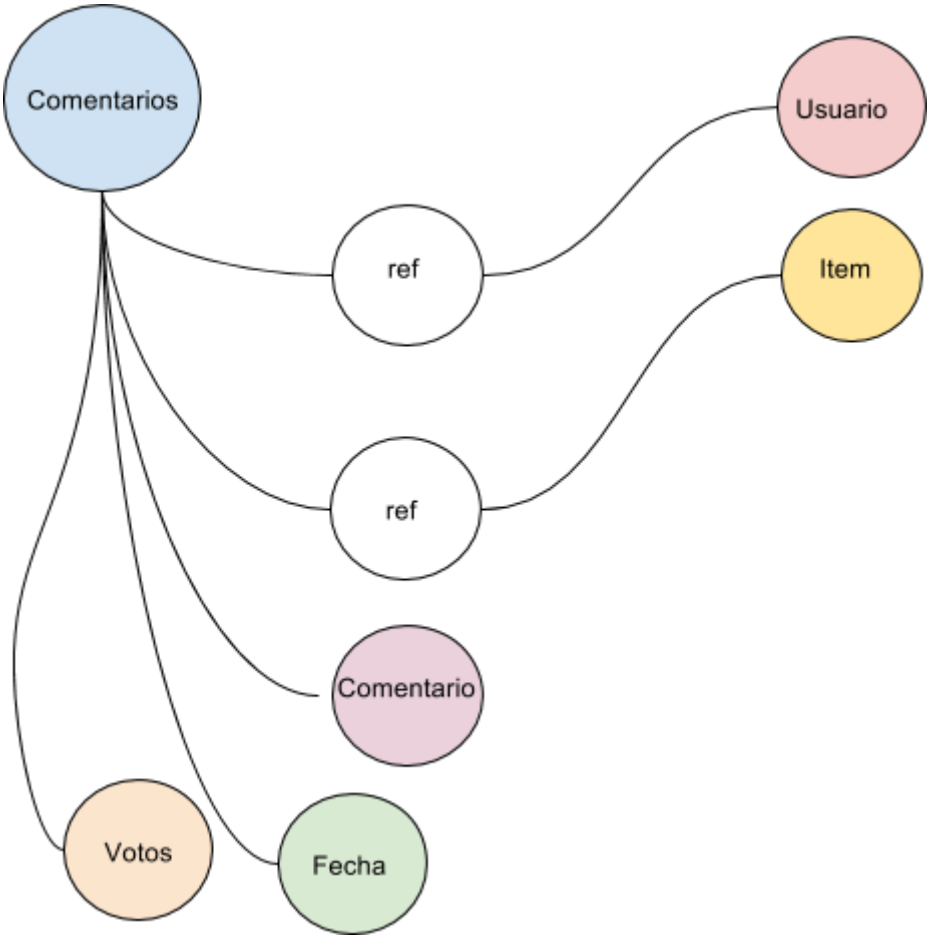


Figura 10: Esquema visual de la relación de los comentarios

Con el fin de fin de facilitar la gestión de nuestra base de datos (creación, modificación y eliminación) se ha hecho uso de Mongoui [12].

Mongoui es una interfaz web alojada en nuestro servidor, Mongoui se ejecuta en tiempo real y ha sido creado por Azat Mardan exclusivamente para ser usado con Node.JS y MongoDB y escrito en Derby.JS. Mongoui aún está en versión beta pero nos ha sido bastante útil ya que nos ha facilitado la gestión de la base de datos sin tener que estar entrando en la terminal de mongo para hacer ediciones pequeñas.

Capítulo 4. Datos adquiridos y caso de uso

4.1 Origen de los datos

No debemos olvidarnos de un pilar importante de esta plataforma como son los datos y otras informaciones extraídas de otras web que serán los encargados de nutrir nuestra plataforma. Dichos datos permiten al usuario interactuar con ellos para mantenerlos lo más actualizado posible. La práctica que siguen las plataformas Open Data es la ofrecer los datos del sector público y otro tipo de recursos para su posible explotación. De esta forma, tanto empresas como la propia ciudadanía puede informarse o crear nuevos servicios aumentando su valor
Las plataformas desde las que se han extraído los datos han sido las siguientes:

- Open Data Canarias: Esta plataforma nace de una colaboración entre el Área de Educación, Juventud e Igualdad del Cabildo de Tenerife, la Universidad de La Laguna y la Fundación General de la Universidad de La Laguna. Dicho proyecto se encarga de fomentar y desarrollar la estrategia Open Data de publicación y reutilización de la información del sector público [3].

- Senderos de Tenerife (Webtenerife): Dicha web ha sido creada para ofrecer información sobre el turismo en la isla de Tenerife. La parte en concreto que nos interesa de esta web es el apartado que tiene en el que recoge los senderos de las islas y una información básica como el inicio, fin, duración, etc. Sin embargo no poseen una API ni de Open Data, por lo tanto no podremos acceder a sus datos de una forma sencilla ya que solo podemos encontrarlos en su web de forma estática. Para ello se ha creado un script en Python que obtiene estos datos aprovechando que esta web tiene una estructura bien definida. A continuación vamos a describir el script que nos ha generado el fichero .csv con los datos que nos interesan desde dicha web:

    - Obtenemos el HTML con el listado de todos los senderos:
    - Eliminamos información innecesaria y guardamos en un array todos los senderos que existen.
    - Una vez tenemos todos los senderos, accedemos a cada uno y obtenemos los datos que nos interesan. (Visualizar apéndice A1)

En la figura 10 se muestra como están dispuestos los datos en WebTenerife.

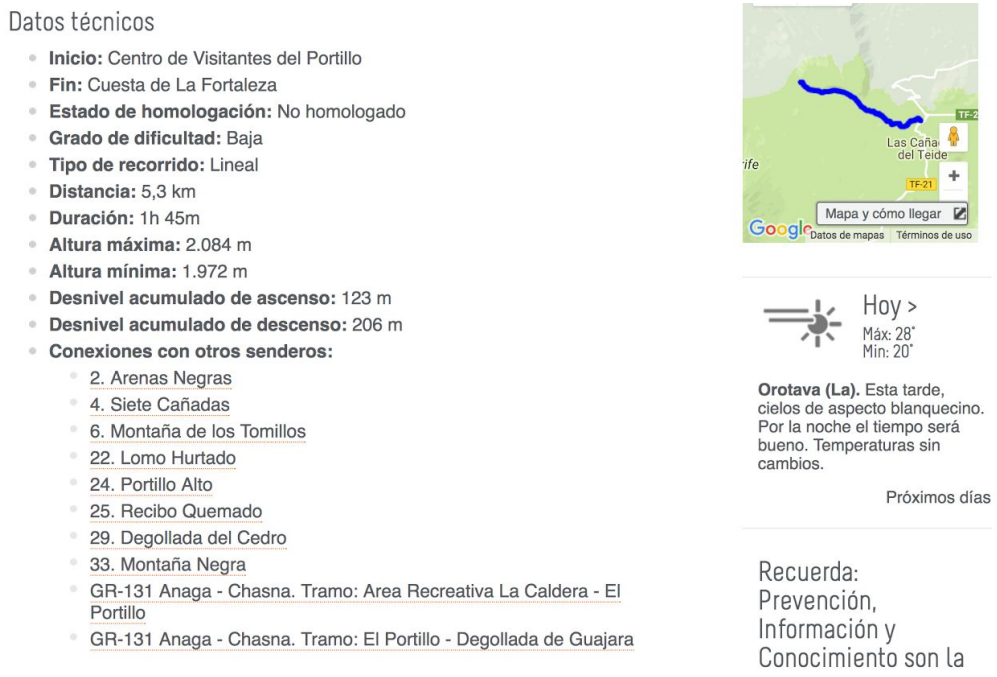


Figura 11: Datos en WebTenerife.

- Gobierno de Canarias: El gobierno de Canarias se ha encargado de recoger los datos y conocimiento de áreas recreativas. Dicha información se encuentra recogida de manera estática en su página web. En esta situación se ha proseguido de manera similar a como se trabajó para la obtención de los datos referentes a los senderos. A diferencia del caso anterior de los senderos, para obtener los datos de parques recreativos tuvimos que adaptarnos a una estructura HTML distinta, con lo que ello conlleva, que es una estructura final del script. Podemos encontrar el código en el apéndice A4 [2].

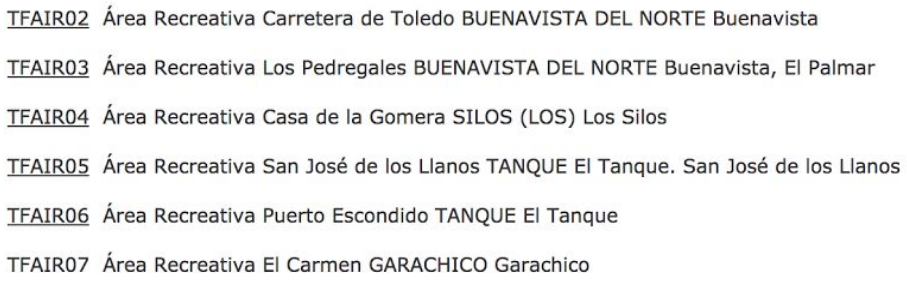


Figura 12: Forma de los datos en la web del Gobierno de Canarias

4.2 Caso de uso de la plataforma Inforestal

En esta sección procederemos a presentar un caso concreto en el que un usuario puede utilizar nuestra plataforma satisfaciendo su necesidad de información actualizada.

Pongamos el supuesto de la familia 'Familia1' la cual quiere realizar el

próximo Domingo una excursión pero sin destino definido, lo único que tienen claro es que tienen 2 hijos uno de ellos tiene 7 años de edad y otro hijo tiene 12 años de edad, por lo tanto buscan un sendero adaptado a sus necesidades.

En primer lugar la familia accede a la aplicación a través de su navegador en cualquier dispositivo. Una vez dentro de la plataforma deciden registrarse para poder acceder a la información rellenando un formulario como el que se encuentra a continuación.

Figura 13: Formulario de registro para la "Família1"

Una vez completado el registro se accede a la página de inicio donde se encuentran todos los items. Existe información que no les resulta relevante por lo tanto deciden filtrar únicamente por los senderos mediante el botón que se encuentra en la leyenda.

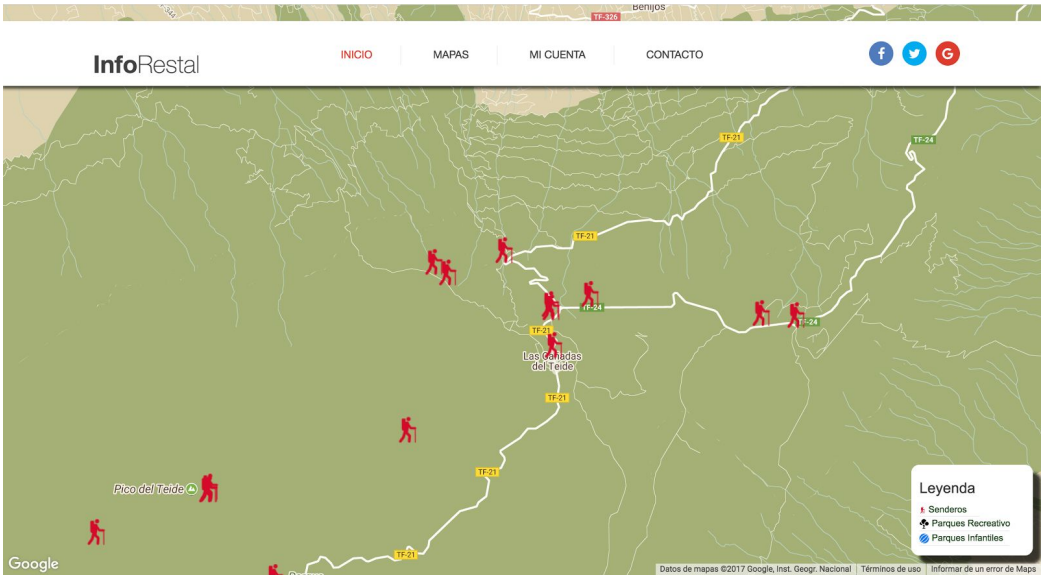


Figura 14: Listado de senderos.

Una vez puede visualizar todos los senderos la familia decidirá gracias a la información disponible cuál es el sendero que más se adapta a su situación, como se puede ver en la imagen a continuación.

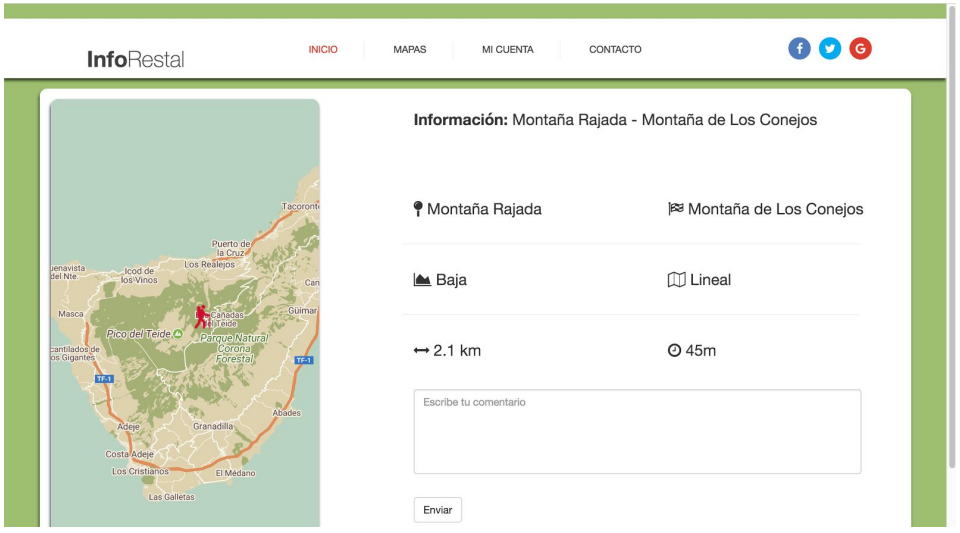


Figura 15: Información detallada de un item.

El día de realizar el sendero la familia se dispone a acceder al sitio pero está un poco perdida así que decide sacar su dispositivo móvil y acceder a la web para ver cuál es concretamente la ubicación, la apariencia de este caso lo podemos ver en la figura 15.

Figura 16: Vista móvil Inforestal.

Finalmente tras hacer el sendero la familia desea dejar un comentario en el item del sendero poniendo lo bien que le ha parecido y añadiendo información que le puede ser útil para otros usuarios.

Figura 17: Comentario realizado sobre un sendero.

No obstante, la familia1 después de añadir su aportación, puede valorar opiniones de otros usuarios.

Figura 18: Botones de valoración de un comentario.

Una vez votado, el sistema se encargará de filtrar ordenando los comentarios. Se parte de los comentarios mas votados hasta los menos votados.

Capítulo 5. Conclusiones

En los siguiente apartados procederemos a presentar las conclusiones de este proyecto y las líneas futuras de este trabajo.

5.1 Conclusiones

- El estudio del estado del arte ha permitido identificar las plataformas más relevantes de los sistemas de gestión del conocimiento orientados a senderos, parques y áreas recreativas.

- Se han empleado fuentes de datos abiertos que han permitido construir una aplicación que gestione conocimiento para el dominio en cuestión.

- Se ha logrado implementar e integrar los módulos de cartografía y de red social que permite al usuario a la vez que hace uso de la información poder generar más conocimiento en función de su experiencia. Por otro lado y con una visión general cabe destacar el uso de datos abiertos (Open Data) y NOSQL, recursos relativamente modernos y hacia los que apuntan las plataformas más innovadoras.

5.2 Líneas futuras

A continuación vamos a definir una serie de líneas que podrían estudiarse o implementarse para hacer una plataforma más eficaz y robusta.

- Crear algoritmos de auto reportes y aceptación de cambios. Capaz de aceptar automáticamente sugerencias de cambio según el status de los usuarios.
- Obtener datos mediante estaciones meteorológicas privadas o públicas. Dichas estaciones se pueden hacer en una arquitectura arduino y que envíen datos a los servidores.
- Funcionalidad de registro y actualización de senderos. Con esta funcionalidad el usuario no solo verá el punto de partida, será capaz de ver la ruta completa que ha sido registrada por otro usuario.
- Nuevas características a sugerencia por el usuario. El usuario puede sugerir campos que pueden parecer importante en determinados items
- Soporte multilingue. Esta funcionalidad permitirá el uso internacional.
- Adaptar disponibilidad para toda canarias. Esto se podría expandir a otros destinos.

Capítulo 6. Summary and Conclusions

The following chapter presents a general definition and conclusions regarding the objectives defined in the first chapter. First we will proceed to analyze the objectives one by one:

- The documentation that has been carried out on similar platforms has been correct. This process has been very useful and we have learned weaknesses, strengths and way of working from other platforms similar to what serves as extra experience.

- The data search was correct. It has occurred with the open data that we were looking for or has used other means but it has been completed successfully.

- It has been possible to implement the geographical module in a satisfactory way, providing an intuitive and easy to use design for the user, which without prior knowledge in the platform is able to understand its operation.

- The social module and the geographical module work in harmony within the application. We can therefore say that the objective of integrating both parties has been achieved

- Inforestal fits well on both mobile devices and desks. This allows users to be able to use the app regardless of the platform they are on.

6.1 Líneas futuras

The objective for which we declare the future lines is to be able to define a series of tasks that to be carried out later by other works of end of degree. These lines could be studied or implemented to make a platform more efficient and robust.

- Create algorithm of auto reports and acceptance of changes. Capable of automatically accepting change suggestions based on user status.
- Implement own weather stations. These stations can be made with arduino and that send data to the servers.

- Tracks logging functionality. With this functionality the user will not only see the starting point, he will be able to see the full path that has been registered by another user.
- New fields suggested by the user. The user can suggest fields that may seem important in certain items.
- Add more languages. This functionality will allow use for a more foreign user type, using i18-n.
- Adapt availability for all Canary Islands. Currently the application is only prepared for Tenerife. This could be expanded to other islands.

Capítulo 7. Presupuesto

7.1 Presupuesto respecto a horas

En la siguiente tabla vamos a plasmar el presupuesto del proyecto basándonos solo en horas empleadas en el.

Actividad	**Duración (horas)**	**Precio por hora (€)**	**Coste (€)**
Obtención de datos	80	15	1200
Diseño	30	20	600
Desarrollo	150	20	3000
Pruebas	20	15	300
Despliegue	20	15	300
TOTAL:	300	-	5400

Tabla 2: Presupuesto horas de trabajo.

7.2 Presupuesto hardware y software y despliegue.

El hardware utilizado para la construcción de dicha plataforma es el siguiente:

Dispositivo	**Descripción**	**Coste (€)**
Ordenador	Dispositivo necesario para poder trabajar	450€

Tabla 3: Presupuesto Hardware empleado

Por otro lado también se ha hecho uso de software de pago pero existen alternativas gratuitas con las que poder trabajar.

Software	**Descripción**	**Coste(€)**
WebStorm	Software de desarrollo web.	129€ /año

Tabla 4: Presupuesto Software empleado

Por último, hay que tener en cuenta el hospedaje de nuestra aplicación web que se desglosa en el siguiente presupuesto:

Recurso	Descripción	Coste
Dominio	Dominio en el cual se alojará la plataforma	5€ /año
Servidor dedicado (Digital ocean)	● 1GB de memoria ● 1 Core ● 30 GB de SSD ● 2 TB de transferencia	10€ /mes
TOTAL:	-	10,45€ mes

Tabla 5: Presupuesto despliegue

Referencias

[1] Webtenerife.com. (2017). Senderos - Tenerife. [online] Available at: http://www.webtenerife.com/que-hacer/naturaleza/senderismo/senderos/ [Accessed 21 Jun. 2017].

[2] Gobiernodecanarias.org. (2017). GOBIERNO DE CANARIAS. [online] Available at: http://www.gobiernodecanarias.org/cmayot/centrodocumentacion/recursoseducativos/guia_equipamientos_naturaleza/guiaequip.jsp?id_isla=40&id_tipo=3 [Accessed 21 Jun. 2017].

[3] Opendatacanarias.es. (2017). Inicio | Open Data Canarias. [online] Available at: http://opendatacanarias.es/ [Accessed 21 Jun. 2017].

[4] Guachapp.com. (2017). Guachapp | Encuentra Vino y Comida Típica Canaria con tu móvil. [online] Available at: http://www.guachapp.com/ [Accessed 21 Jun. 2017].

[5] Shorcial.wordpress.com. (2017). (no title). [online] Available at: https://shorcial.wordpress.com/ [Accessed 21 Jun. 2017].

[6] Fg.ull.es. (2017). Via ULL | Fundación General de la Universidad de La Laguna. [online] Available at: https://www.fg.ull.es/es/tag/via-ull/ [Accessed 21 Jun. 2017].

[7] Airbnb. (2017). Alquileres vacacionales, casas, experiencias y lugares - Airbnb. [online] Available at: https://www.airbnb.es/ [Accessed 21 Jun. 2017].

[8] Smartcitizen.me. (2017). Smart Citizen : Citizen Science Platform for participatory processes of the people in the cities. [online] Available at: https://smartcitizen.me/kits/ [Accessed 21 Jun. 2017].

[9] Álvarez, C. (2017). ¿Cómo funciona Node.js?. [online] Genbetadev.com. Available at: https://www.genbetadev.com/frameworks/como-funciona-node-js [Accessed 21 Jun. 2017].

[10] dzone.com. (2017). A Comparison of SQL and NoSQL to Simplify Your Database Decision - DZone Database. [online] Available at: https://dzone.com/articles/a-comparison-of-sql-and-nosql-to-simplify-your-dat [Accessed 22 Jun. 2017].

[11] Wiki.openstreetmap.org. (2017). ES:Dosier de prensa - OpenStreetMap Wiki. [online] Available at: http://wiki.openstreetmap.org/wiki/ES:Dosier_de_prensa [Accessed 21

Jun. 2017].

[12] webapplog: [programming weblog]. (2017). MongoUI: Real-Time MongoDB Admin Web Interface (a la phpMyAdmin in Node.js). [online] Available at: https://webapplog.com/mongoui/ [Accessed 30 Jun. 2017].

Apéndice.

A.1. Modelo de usuario

```
/*********************************************************************************
*
* user.js
*
*********************************************************************************
*let mongoose = require('mongoose');
*let Schema = require('mongoose').Schema
*
*let User = new Schema({
*    nick            :   String,
*    mail            :   String,
*    password        :   String,
*    status          :   { type: String, default: '0' }
*
*});
*module.exports = mongoose.model('User', User);
*
*
*********************************************************************************/
```

A.2. Modelo para los senderos

```
/*********************************************************************************
*
* sendero.js
*
*********************************************************************************
*
*
*let mongoose = require('mongoose');
*let Schema = require('mongoose').Schema
```

```
*let Sendero = new Schema({
*    partida            :   String,
*    fin                :   String,
*    dificultad         :   String,
*    mapa               :   String,
*    distancia          :   String,
*    duracion           :   String,
*    latitud            :   String,
*    longitud           :   String
*});
*module.exports = mongoose.model('Sendero', Sendero);
*
*
*********************************************************************************/
```

A.3. Modelo para los parques recreativos

```
/*********************************************************************************
*
* parques_recreativos.js
*
*********************************************************************************
*
*
* let mongoose = require('mongoose');
* let Schema = require('mongoose').Schema
*
* let Parques = new Schema({
*    nombre           :   String,
*    isla             :   String,
*    municipio        :   String,
*    acceso           :   String,
*    transporte       :   String,
*    contacto         :   String,
*    permiso          :   String,
 *   superficie       :   String,
 *   minusvalido      :   String,
```

```
 *   agua             :   String,
 *   electricidad     :   String,
 *   duchas           :   String,
 *   bar              :   String,
 *   comedor          :   String,
 *   fogones          :   String,
 *   bancos           :   String,
 *   aparcamientos    :   String,
 *   latitud          :   String,
 *   longitud         :   String
 * });
 * module.exports = mongoose.model('Parques', Parques);
 *
 *
*********************************************************************************/
```

A.4. Script para la obtención de datos desde webs estáticas.

```
/*********************************************************************************
*
*
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# -*- encoding: utf-8 -*-
# encoding: latin1
import re
import sys

import urllib.request

print (sys.stdout.encoding)



def cargar_links():
    expr = "map-data"

                                        wp          =          urllib.request.urlopen
("http://www.webtenerife.com/que-hacer/naturaleza/senderismo/senderos/?gclid=CMfX7qWrot
ICFYc_GwodZi4LyQ&page-index=1&tab-view-mode=listado" )
```

```
        pw = wp.read ( ).decode ( wp.headers.get_content_charset ( ) )

        res = pw.split ( )
        indx = [ ]
        res2 = [ ]
        for i in range ( 0 , len ( res ) ):
            if re.search ( expr , str ( res[ i ] ) ):
                indx += [ i ]
        res2 = [ ]
        for i in range ( 0 , len ( indx ) ):
            aux = str ( res[ indx[ i ] ] ).split ( "/" )
            tam = len ( aux )
            res[ indx[ i ] ] = aux[ tam - 1 ]
            aux = str ( res[ indx[ i ] ] ).split ( "\"" )
            res[ indx[ i ] ] = aux[ 0 ]
            res[ indx[ i ] ] = res[ indx[ i ] ].replace ( "aspx" , "htm" )
                                                                    res2        +=        [
"http://www.webtenerife.com/que-hacer/naturaleza/senderismo/senderos/" + res[ indx[ i ]
] ]

        return res2

        print("Done")



    links1 = cargar_links ( )

    resultado = [ ]
    for i in range ( 0 , len ( links1 ) ):
        vec_ub = urllib.request.urlopen ( links1[ i ] )
        vec_ub = vec_ub.read ( ).decode ( vec_ub.headers.get_content_charset ( ) )
        vec_ub = vec_ub.split ( "<" )

        vec_lin = urllib.request.urlopen ( links1[ i ] )
        vec_lin = vec_lin.read ( ).decode ( vec_lin.headers.get_content_charset ( ) )
        vec_lin = vec_lin.split ( "<strong>" )
        for i in range ( 0 , len ( vec_lin ) ):
            if re.search ( "Inicio:" , str ( vec_lin[ i ] ) ):
                aux = vec_lin[ i ].split ( "</strong>" )
                aux = aux[ 1 ].split ( "</li>" )
```

```
        resultado += [ aux[ 0 ].lstrip ( ) ]
for i in range ( 0 , len ( vec_lin ) ):
    if re.search ( "Fin:" , str ( vec_lin[ i ] ) ):
        aux = vec_lin[ i ].split ( "</strong>" )
        aux = aux[ 1 ].split ( "</li>" )
        resultado += [ aux[ 0 ].lstrip ( ) ]
for i in range ( 0 , len ( vec_lin ) ):
    if re.search ( "Grado de dificultad:" , str ( vec_lin[ i ] ) ):
        aux = vec_lin[ i ].split ( "</strong>" )
        aux = aux[ 1 ].split ( "</li>" )
        resultado += [ aux[ 0 ].lstrip ( ) ]
for i in range ( 0 , len ( vec_lin ) ):
    if re.search ( "Tipo de recorrido:" , str ( vec_lin[ i ] ) ):
        aux = vec_lin[ i ].split ( "</strong>" )
        aux = aux[ 1 ].split ( "</li>" )
        resultado += [ aux[ 0 ].lstrip ( ) ]
for i in range ( 0 , len ( vec_lin ) ):
    if re.search ( "Distancia:" , str ( vec_lin[ i ] ) ):
        aux = vec_lin[ i ].split ( "</strong>" )
        aux = aux[ 1 ].split ( "</li>" )
        resultado += [ aux[ 0 ].lstrip ( ) ]
for i in range ( 0 , len ( vec_lin ) ):
    if re.search ( "Duraci" , str ( vec_lin[ i ] ) ):
        aux = vec_lin[ i ].split ( "</strong>" )
        aux = aux[ 1 ].split ( "</li>" )
        resultado += [ aux[ 0 ].lstrip ( ) ]
for i in range ( 0 , len ( vec_ub ) ):
    if re.search ( "meta itemprop=" , str ( vec_ub[ i ] ) ):
        aux = vec_ub[ i ].split ( "content=\"" )
        aux = aux[ 1 ].split ( "\"" )
        resultado += [ aux[ 0 ].lstrip ( ) ]
# print(resultado)
with open ( 'lib/output.csv' , 'a' ) as f:
    print (len ( resultado ))
    f.write ( ((str ( resultado )).replace ( "[" , "" )).replace ( "]" , "" ) )
    f.write ( "\n" )
    f.close ( )
    resultado = [ ]
```

```
*
**********************************************************************************/
```