Desarrollo de una aplicación para el turismo en Madrid



Trabajo Fin de Grado

Curso 2018-2019

Autor

Eduardo Vela Galindo

Director

Antonio Sarasa Cabezuelo

Departamento de Ingeniería del Software e Inteligencia Artificial

Universidad Complutense de Madrid

Madrid, 2018

# AGRADECIMIENTOS

# RESUMEN

# ABSTRACT

# PALABRAS CLAVE

Arquitectura multicapa, Node.js, Aplicación Web, Gestor de Contenido Web, APIS

# ACRÓNIMOS

API: Application Programming Interface

EJS: (Embedded JavaScript templates)

BD: Base de datos

CSS: Cascading Sytle Sheets

HTML: HyperText Markup Lenguage

JS: Javascript

JSON: Javascript Object Notation

NPM: Node Package Manager

TFG: Trabajo de Fin de Grado

UCM: Universidad Complutense de Madrid

ÍNDICE

[AGRADECIMIENTOS 3](#_Toc534213399)

[RESUMEN 4](#_Toc534213400)

[ABSTRACT 5](#_Toc534213401)

[PALABRAS CLAVE 6](#_Toc534213402)

[ACRÓNIMOS 7](#_Toc534213403)

[1. INTRODUCCIÓN 1](#_Toc534213404)

[1.1 Motivación 2](#_Toc534213405)

[1.2 Objetivos 2](#_Toc534213406)

[2. ESTADO DEL ARTE 4](#_Toc534213407)

[3. TECNOLOGÍA EMPLEADA 6](#_Toc534213408)

[3.1. NODE.JS 7](#_Toc534213409)

[3.1.1. NODE PACKAGE MANAGER 7](#_Toc534213410)

[3.1.2. EXPRESS.JS 8](#_Toc534213411)

[3.1.3. MYSQL 9](#_Toc534213412)

[3.1.4. OTROS MÓDULOS 10](#_Toc534213413)

[3.2. TECNOLOGÍAS DEL CLIENTE: HTML Y CSS (BOOTSTRAP) 10](#_Toc534213414)

[3.3. MYSQL (PHPMYADMIN) 10](#_Toc534213415)

[3.4. APIS 10](#_Toc534213416)

[4. ESPECIFICACIÓN 11](#_Toc534213417)

[5. ARQUITECTURA DE LA APLICACIÓN 13](#_Toc534213418)

[6. DISEÑO DE LA APLICACIÓN 15](#_Toc534213419)

[7. CONCLUSIONES Y TRABAJO FUTURO 17](#_Toc534213420)

[ANEXO 19](#_Toc534213421)

[BIBLIOGRAFÍA 21](#_Toc534213422)

# INTRODUCCIÓN

## 1.1 Motivación

En la era actual las aplicaciones de escritorio tradicionales están cada vez más en desuso en comparación a las aplicaciones web debido a nuestros nuevos hábitos y costumbres.

En general cuando queremos acceder a un servicio o aplicación, queremos hacerlo de forma inmediata, sin necesidad de descargar y/o instalar nada previamente, y las aplicaciones de escritorio van a necesitar de como mínimo uno de los pasos mencionados. Sin embargo, las aplicaciones web permiten acceder a ellas de forma inmediata, con el simple uso de un explorador, disponible en prácticamente cualquier plataforma actual, ya sea móvil, portátil, sobremesa o incluso consolas modernas.

Es por eso que en este TFG se plantea una recopilación del conocimiento adquirido a lo largo del grado, como bases de datos, arquitecturas o algoritmia, haciéndolo culminar en una aplicación web que amplíe y asiente las bases del conocimiento adquirido en la carrera de esta materia, que no es todo lo extenso que podría esperarse de un estudiante de ingeniería del software.

Y para lograr ese objetivo una de las opciones más atractivas era el desarrollo de un gestor de contenido web personalizado que necesita un conjunto muy amplio de tecnologías.

## 1.2 Objetivos

Por lo mencionado anteriormente, los objetivos del presente TFG son principalmente los siguientes:

* Desarrollo de una aplicación web dentro un entorno de ejecución dinámico y moderno como es Node.js y que está dirigido principalmente al desarrollo de aplicaciones web en Javascript y cuyo futuro está claramente en alza.
* Aprendizaje y familiarización con las APIS de acceso a datos, que permiten utilizar una gran cantidad de información sin necesidad de mantenerla actualizada y/o ampliarla. Como por ejemplo la API de datos del ayuntamiento de Madrid, de la que hablaremos más adelante.
* Implementación de un gestor de contenido web que permita dar una funcionalidad nueva en base a información proporcionada por terceos, constituyendo así un servicio completamente nuevo.

# ESTADO DEL ARTE

Vamos a hacer un breve repaso a lo largo de la historia del desarrollo web, y posteriormente incidiremos un poco más en cómo y cuando surge Node y por qué se ha decidido utilizarlo para el desarrollo de esta aplicación web en particular.

# TECNOLOGÍA EMPLEADA

## TECNOLOGÍA DEL SERVIDOR: NODE.JS

Node es un entorno de programación en tiempo de ejecución multiplataforma principalmente para la capa del servidor basado en el lenguaje de programación ECMAScript y en el motor V8 de Google (entorno de ejecución JavaScript de Google Chrome[[1]](#footnote-1)). Una de las principales cualidades de Node es su funcionamiento concurrente asíncrono, que utiliza un hilo de ejecución principal para la aplicación, del que se generan el resto de las llamadas concurrentes de entrada/salida[[2]](#footnote-2).

Otra de las cualidades más llamativas y apreciadas de Node que importa del ya mencionado motor V8, es el uso de módulos que extienden enormemente la funcionalidad de Node y le convierten en un entorno escalable muy configurable. Estos módulos pueden ser enormes y añadir una gran funcionalidad como Express[[3]](#footnote-3) o añadir una funcionalidad más concreta y acotada puede ser el módulo MySQL que permite lanzar y recibir consultas de una base de datos SQL, aunque ya hablaremos más delante de todos los módulos utilizados.

El carácter asíncrono y “no bloqueante” de las llamadas de datos recibidas sumado a una alta tolerancia de concurrencia le convierten en un lenguaje perfecto para aplicaciones web ligeras y es ese el motivo por el que ha sido elegido para sostener la aplicación web. Además, se encuentra bajo licencia MIT de código abierto, por lo que puede ser utilizada sin problemas de derechos.

A continuación, vamos a hablar un poco más en detalle de la tecnología Node empleada, en concreto haremos un repaso por algunos de los módulos más destacados que hemos utilizado.

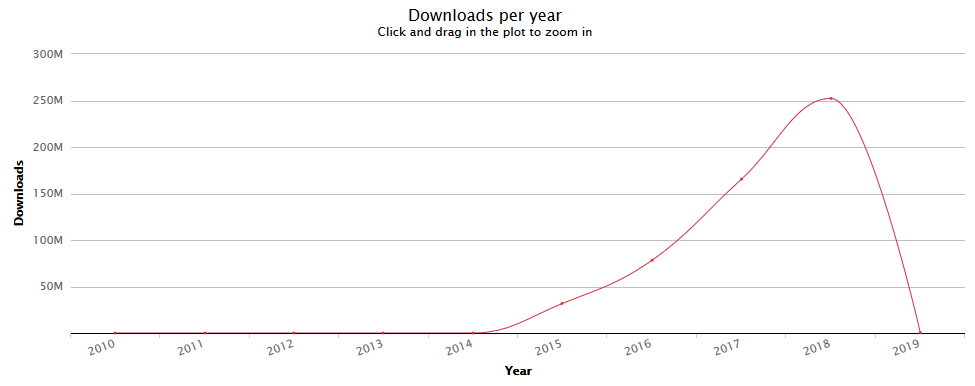
## NODE PACKAGE MANAGER

Node Package Manager (de ahora en adelante npm) es el manejador de paquetes de Node, el cual se usa para instalar y mantener actualizados todos los módulos y requisitos de una aplicación Node. Este tiene un funcionamiento similar al de otros manejadores de paquetes tradicionales como puede ser Yum. Aunque también se puede utilizar para instalar y/o actualizar aplicaciones Node enteras.[[4]](#footnote-4)

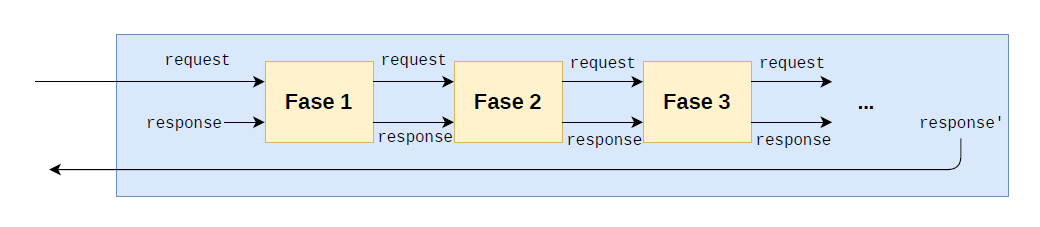
Npm se instala automáticamente junto con Node desde la versión 0.6.3 y se accede a él mediante el comando “npm” lanzado desde una terminal con acceso a Node. Algunos ejemplos significativos de npm, y que serán necesarios más adelante durante la guía de instalación son npm install o npm update.

## EXPRESS.JS

Express es un framework[[5]](#footnote-5) diseñado y enfocado para desarrollar aplicaciones web y APIS en Node. Su uso se ha popularizado mucho (solo en github podemos encontrar más de 120.000 proyectos que lo utilizan y que suponen un aproximadamente 30% del total de proyectos Node de Github[[6]](#footnote-6) y además según npm-stats acumula más de medio millón de descargas[[7]](#footnote-7) como muestra la figura 3.1), tanto que es considerado como el “framwork” por defecto si vas a desarollar para web en Node.

Figura 3.1. Descargas por año del framework Express.js

A pesar de que su gran popularidad es ya en sí un factor determinante para elegir Express como framework principal (a mayor popularidad, mayor será la documentación y funcionalidades existentes, entre otras ventajas) este no es el único factor. Es un framework muy versátil, simple en su versión vanilla pero muy potente gracias a su sistema de “middelwares” que permite extender una gran cantidad de funcionalidades ya creadas de manera muy simple o crear las nuestras propias. A continuación podemos ver un diagrama sencillo del funcionamiento de los middelware en Express.

Figura 3.2.Funcionamiento de los middlewares en Express.js

Como muestra la figura 3.2, cuando Express recibe una petición del cliente en el servidor, este va pasando por distintas fases (cada una de estas fases es un middelware) y al concluir dicha fase se encadena con la siguiente hasta terminar.

## MYSQL

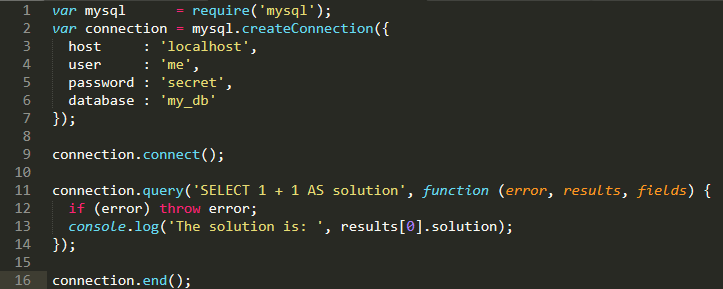
Este módulo actúa como driver de Node para la utilización de MySQL. Se encuentra bajo licencia MIT y está escrito íntegramente en JavaScript. En nuestro proyecto lo utilizamos combinado con otros módulos menores que extienden la funcionalidad de este como MySQLStore y sessionStore.Este módulo permite establecer una conexión con una base de datos MySQL siguiendo el siguiente formato:

Figura 3.3.Ejemplo de uso del módulo MySQL[[8]](#footnote-8)

## OTROS MÓDULOS

Aunque los dos módulos citados anteriormente son los más representativos del proyecto, la lista de módulos usada es muy larga y no tiene sentido explicar en detalle el funcionamiento de todos ellos, por lo que a continuación proporcionamos una lista con una breve descripción del resto de módulos:

* Request: Simplifica enormente el manejo de rutas http y https.
* CreateError: Permite generar errores personalizados.
* ExpressSession: Permite generar y gestionar sesiones de usuarios .
* ExpressMysqlSession: Permite insertar y mantener las sesiones en BD.
* Path: simplifica el uso de rutas internas para establecer la localiación de archivos locales.
* CookieParser: Permite parsear el contenido de datos, por ejemplo JSON cuando estos están contenidos en una cookie.
* Async: Añade una gran cantidad de funcionalidades para lidiar con la problemática de trabajar en un entorno asíncrono.
* Wdk: Simplica la conexión con la API de datos Wikidata.

## TECNOLOGÍAS DEL CLIENTE

Para la tecnología del cliente, al tratarse de una aplicación web, lo más idóneo era un portal web basado en HTML5 y CSS3. Debido a que ambos son lenguajes tradicionales de desarrollo web no vamos a estendernos en la explicación de ambas tecnologías más allá de mencionar que HTML es el lenguaje utilizado para la estructuración de la vista y CSS para la correcta maquetación de la misma.

En nuestra aplicación web hemos utilizado como plantillas EJS(Embedded JavaScript templates)

## BOOTSTRAP

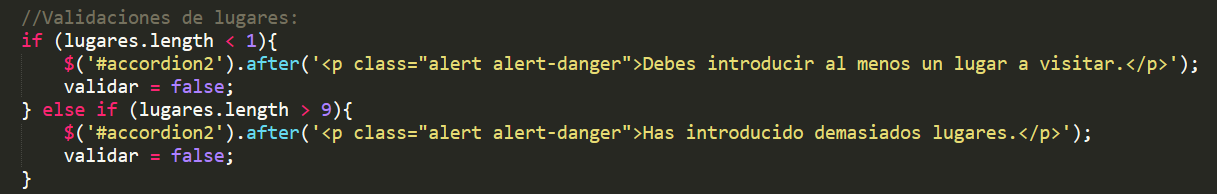
No obstante, y debido que la maquetación de una página web si se hace siguiendo una hoja de estilos adecuada, puede demorarse meses de grandes esfuerzos (y al fin y al cabo la vista de la aplicación no es el objetivo de este tfg) se incluyó BOOTSTRAP como tecnología para agilizar las tareas de maquetación. Aunque puede llevar algo de tiempo aprender a utilizarlo, su curva de aprendizaje es tan rápida y su potencial tan alto que compensa claramente su uso.

## JQUERY Y AJAX

Otra de las tecnologías relevantes utilizadas en el cliente ha sido la incorporación de JavaScript en el cliente. La potencia de funcionamiento que aporta esta tecnología en el cliente es altísima y su uso está altamente generalizado. Aunque es cierto que puede suponer algunos riesgos para la seguridad del cliente (necesaria referencia) sus capacidades claramente valen la pena, ya que nos van a permitir relegar una gran carga de operaciones del servidor al cliente, además de agilizar muchos procesos y de dar una sensación de mayor velocidad y rendimiento de la aplicación web.

No obstante, para incrementar las funcionalidades de Javascript se ha recurrido a dos tecnologías complementarias como son JQuery y Ajax:

* JQuery es una librería de JavaScript que nos permite interactuar de manera muy simple y potente con el árbol DOM de las plantillas hmtl asociadas a dicho script. Además, también permite modificar dinámicamente elementos tanto del DOM como sus estilos asociados (CSS). En la siguiente figura (figura 3.4) podemos ver como se inserta dinámicamente código en el DOM utilizando en este caso el método de JQuery after(), que inserta a continuación del lugar especificado el código indicado entre los paréntesis.

Figura 3.4. Ejemplo de inserción de código en el DOM de forma dinámica

* AJAX (*Asynchronous JavaScript And XML*) es una tecnología web que permite a las aplicaciones interactuar de forma interactiva y dinámica con el servidor mediante una conexión asíncrona permitiéndonos así poder realizar peticiones con el servidor sin necesidad de recargar la página del navegador.

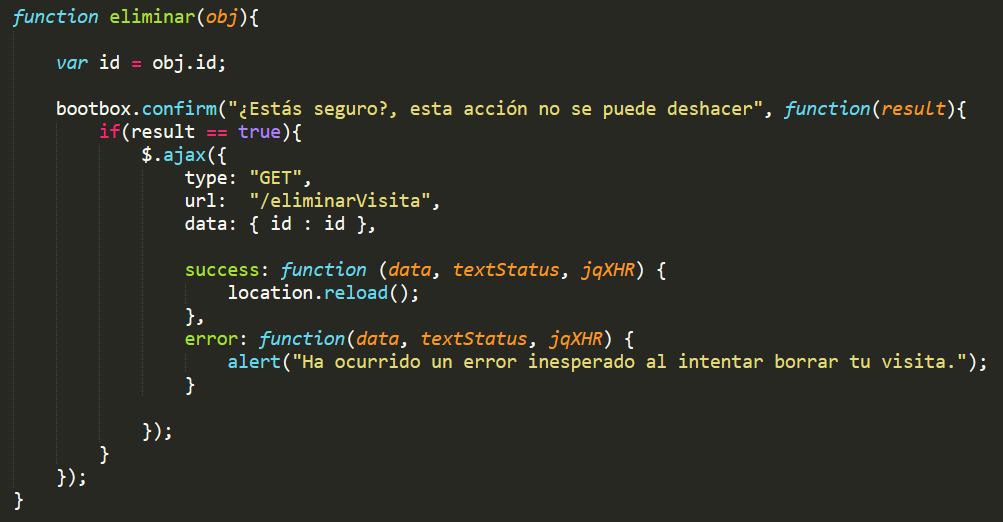
Lo cual mejora enormemente la velocidad y usabilidad de las aplicaciones, además de darle al usuario una mayor sensación de control de la aplicación. A continuación mostramos un ejemplo de llamada al servidor mediante AJAX:

Figura 3.5. Ejemplo de llamada al servidor usando AJAX

## TECNOLOGÍAS DE LA BD: MYSQL (PHPMYADMIN)

## MARÍADB

MySQL es un gestor de bases de datos relacionales que se encuentra bajo una licencia dual (código libre y otra comercial bajo Oracle). Aunque en concreto se utiliza MariaDB, un sistema derivado de MySQL bajo licencia GPL.

Aunque en la práctica MySQL y MariaDB son prácticamente iguales, este último incorpora algunas diferencias en los mecanismos de almacenamiento y en prestaciones, que la hacen más atractiva al uso.

## PHPMYADMIN

PhpMyAdmin es un programa utilizado para manejar y administrar bases de datos de tipo MySQL, en concreto de tipo MariaDB, de una forma sencilla a través de un navegador Web. Este programa permite un manejo total de las funcionalidades típicas de una base de datos (modificar campos, eliminar, crear o modificar tablas, eliminar o insertar tuplas entre otras muchas opciones) de forma sencilla e interactiva sin necesidad de introducir manualmente los comandos SQL.

El proyecto PhpMyAdmin se encuentra licenciado bajo GPL v2 y ha sido desarrollado bajo un modelo de software libre en php[[9]](#footnote-9).

## APACHE

Apache es un servidor que implementa el protocolo HTTP y la noción de un sitio virtual[[10]](#footnote-10). Se utiliza principalmente para el envío de páginas web a través de la WWW además de para servir de contenedor para, por ejemplo, el servidor de MySQL. Se encuentra bajo licencia GPL v2.

## XAMPP

Se trata de una herramienta de software libre que se utiliza para facilitar la gestión de servidores, concretamente MySQL y Apache. El uso de esta herramienta facilita enormemente la gestión de ambos servidores ya inlcuye en un solo paquete los servidores ya preparados y configurados para su ejecución y uso bajo diversos sistemas operativos.

# ESPECIFICACIÓN

## DIAGRAMA DE CASOS DE USO

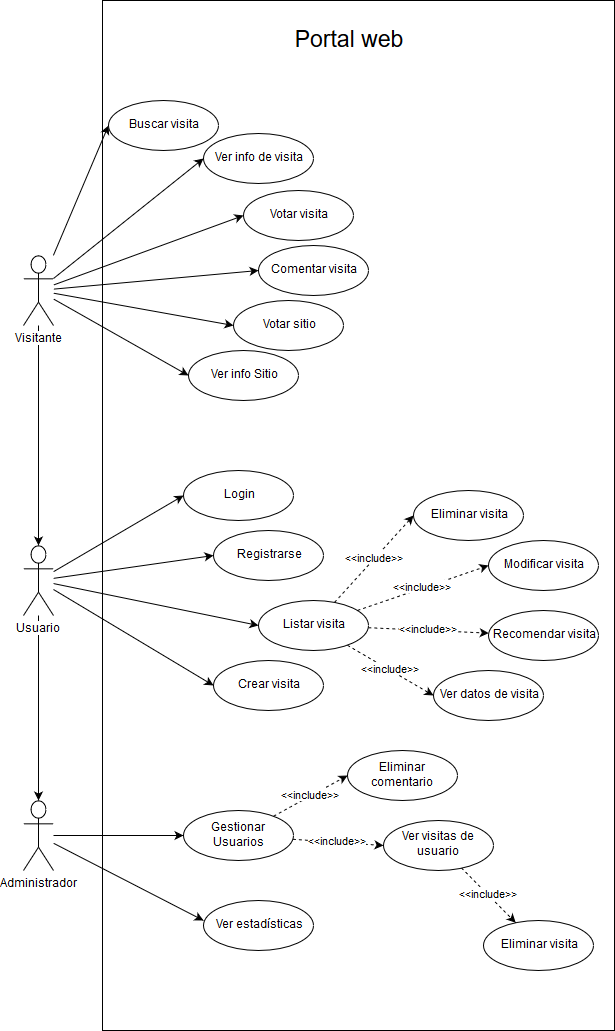
A continuación presentamos el diagrama de casos de uso bajo el que se ha desarrollado la aplicación. Este diagrama pretende dar una idea aproximada de una forma gráfica de la naturaleza de la aplicación y su alcance.

Figura 4.1. Diagrama de casos de uso

## CASOS DE USO

Ahora que ya hemos descrito el diagrama de casos de uso, vamos a hacer una descripción más detallada de cada una de las funciones que hemos implementado. Los casos de uso se han clasificado según el rol del usuario que puede ser Visitante, Usuario registrado o Administrador. Previamente vamos a empezar a describir los casos de uso vamos a definir una serie de conceptos que serán útiles para la correcta comprensión de los casos de uso:

* Visitante: Accede al portal sin identificarse.
* Usuario registrado: Accede al portal habiéndose registrado previamente y se identifica satisfactoriamente. Puede llevar a cabo todas las acciones del visitante, además de las suyas propias
* Administrador: Accede al portal y se identifica satisfactoriamente. Los administradores no pueden ser estableces vía aplicación, sino que deben ser registrados manualmente en la BD. No puede llevar a cabo todas las opciones de los usuarios registrados, aunque sí de los visitantes, además de las suyas.
* Visita: Se trata del objeto principal de esta aplicación. Está compuesta por un título, una descripción, los lugares que se desean visitar y tiene siempre asociado un usuario creador.
* Sitio: Es el componente básico de la visita, y se trata de un lugar concreto que de sea visitar, por ejemplo, el Museo del Prado.
* Sistema: Siempre que se haga referencia al sistema nos estamos refiriendo a la aplicación en sí misma, ya sea en el lado del servidor o en el lado del cliente.

## CASOS DE USO DEL VISITANTE

|  |  |
| --- | --- |
| Buscar una visita | |
| Entrada | Texto que contiene la palabra clave que se quiera buscar. |
| Flujo del evento | 1. El visitante introduce la cadena de texto que desea buscar en un cuadro de texto habilitado para ello. 2. El sistema busca en la BD la palabra clave introducida y devuelve aquellas visitas que contienen en su título, descripción o en su lista de palabras claves una coincidencia. Puede haber dos posibles resultados:    1. Hay coincidencias: en cuyo caso se listan las 5 primeras visitas para las que se encontró una coincidencia.    2. No hay coincidencias: en cuyo caso se informa al usuario y no se lista nada. |

|  |  |
| --- | --- |
| Ver información de una visita | |
| Entrada | Visita seleccionada. |
| Flujo del evento | 1. El visitante selecciona el incono de ver información asociada a una visita, representado de forma universal por un  en toda la aplicación. 2. El sistema busca la información correspondiente a esa visita y abre un panel sobre el que se muestra la información que ha encontrado. |

|  |  |
| --- | --- |
| Votar una visita | |
| Entrada | Visita seleccionada.  Voto numérico (entre 1 y 5). |
| Flujo del evento | 1. El visitante selecciona la visita a la que desea calificar. 2. Se abre un panel con la información asociada a la visita, incluyendo un apartado indicando la puntuación actual de la visita y la posibilidad para votar la visita. 3. El visitante establece una puntuación entre 1 y 5 y selecciona la opción de enviar. 4. El sistema contabiliza el voto e informa de ello al visitante. |

|  |  |
| --- | --- |
| Comentar una visita | |
| Entrada | Visita seleccionada.  Texto con el comentario. |
| Flujo del evento | 1. El visitante selecciona la visita sobre la que desea comentar. 2. Se abre un panel con la información asociada a la visita, incluyendo un apartado indicando los comentarios asociados y la posibilidad de escribir un nuevo comentario. 3. El visitante escribe en un cuadro habilitado para ello el comentario deseado y selecciona la opción de enviar comentario. Puede haber dos posibles resultados:    1. Se contabiliza el comentario correctamente y se informa al visitante.    2. Se encuentra un error en el comentario (por ejemplo que esté vacío) y se informa de ello al visitante. |

|  |  |
| --- | --- |
| Listar Sitios | |
| Entrada | N/a. |
| Flujo del evento | 1. El visitante selecciona la opción de Sitios en el menú de navegación del portal. 2. Se redirecciona a una nueva página del portal, donde se listan todos los sitios asociados al sistema. |

|  |  |
| --- | --- |
| Buscar Sitio | |
| Entrada | Texto con la clave a buscar y/u otras opciones. |
| Flujo del evento | 1. El visitante introduce la cadena de texto que desea buscar en un cuadro de texto habilitado para ello. También puede introducir otras opciones de filtrado. 2. El sistema busca entre todos los sitios con dos posibles resultados:    1. Hay coincidencias: se muestran los sitios con coincidencias encontradas.    2. No hay coincidencias: no se muestra nada. |

|  |  |
| --- | --- |
| Ver información de un sitio | |
| Entrada | Sitio seleccionado. |
| Flujo del evento | 1. El visitante selecciona el sitio del cual quiere ver su información asociada. 2. El sistema busca la información correspondiente a ese sitio y abre un panel sobre el que se muestra la información que ha encontrado. |

## CASOS DE USO DEL USUARIO REGISTRADO

|  |  |
| --- | --- |
| Login o autentificación | |
| Entrada | Email del usuario que se desea autentificar.  Contraseña del usuario que se desea autentificar. |
| Flujo del evento | 1. El visitante selecciona la opción de “Login” en el menú de navegación del portal. 2. Se redirecciona a una nueva página del portal, donde se muestra un panel dando al visitante la opción de introducir su email, su contraseña y enviar datos. 3. El usuario introduce su email y su contraseña en los campos correspondientes habilitados para ello y selecciona la opción de enviar datos, con dos posibles resultados:    1. El email y su contraseña asociado son correctos y por tanto el usuario queda autenticado correctamente con su rol correspondiente.    2. El email o su contraseña no son correctos, por lo que se informa al visitante y este no queda autenticado. |

|  |  |
| --- | --- |
| Registro | |
| Entrada | Email.  Nick.  Nombre y apellidos.  Contraseña.  Otros datos. |
| Flujo del evento | 1. El visitante selecciona la opción de “Registro” en el menú de navegación del portal. 2. Se redirecciona a una nueva página del portal, donde se muestra un panel dando al visitante la opción de introducir los datos con los que se desea registrar en la aplicación. 3. El usuario introduce sus datos y una vez ha concluido selecciona la opción de enviar con dos posibles resultados:    1. Los datos introducidos por el usuario son correctos y por tanto queda registrado, desde ese preciso momento ya puede autenticarse en el portal mediante la opción de Login.    2. Alguno de los datos no es correcto, el sistema informa al usuario de qué datos no son correctos y el motivo de ello. Hasta que todos los datos no sean correctos el visitante no podrá registrarse. |

|  |  |
| --- | --- |
| Listar visitas asociadas a usuario | |
| Entrada | N/a. |
| Flujo del evento | 1. El usuario registrado selecciona la opción de “Mis Visitas” en el menú de navegación del portal. 2. El sistema comprueba los datos de sesión del usuario y si son correctos lista todas las visitas asociadas a ese usuario en una nueva página. 3. El usuario puede llevar a cabo diversas funciones sobre cada una de las visitas entre las que se incluyen:    1. Eliminar visita.    2. Modificar visita.    3. Recomendar visita.    4. Ver información de visita.   A continuación se detallan estos casos de uso, para identificar que son dependientes del caso de uso de listar visita, se identificarán con un “Listar:” antes del nombre. Por lo que para todos los casos de uso que incluyan “Listar” antes supondremos que ya se ha llevado a cabo dicha opción antes. |

|  |  |
| --- | --- |
| Listar : Eliminar visita | |
| Entrada | Visita seleccionada. |
| Flujo del evento | 1. El usuario registrado selecciona la opción de eliminar, representada por un  de forma universal en toda la aplicación. 2. El sistema pide confirmación al usuario antes de eliminar la visita con dos posibles resultados:    1. El usuario confirma la operación, en cuyo caso la visita se elimina, y ya no estará disponible.    2. El usuario cancela la operación en cuyo caso no se verá afectada la visita. |

|  |  |
| --- | --- |
| Listar : Recomendar visita | |
| Entrada | Visita seleccionada. |
| Flujo del evento | 1. El usuario registrado selecciona la opción de recomendar, representada por un  de forma universal en toda la aplicación. 2. Se abre un panel donde se da opción de especificar el email del usuario al que se quiere recomendar la visita. También ofrece la posibilidad de introducir un texto asociado a dicha recomendación. 3. El usuario introduce el email del destinatario de la recomendación con dos posibles resultados:    1. El email existe y se corresponde con el de un usuario registrado de la aplicación.       1. En este caso se registra la nueva visita dentro de la tabla de visitas recomendadas del usuario especificado.    2. El email no existe, en cuyo caso se informa al usuario de que no hay coincidencias encontradas y no se recomendará la visita. |

|  |  |
| --- | --- |
| Listar : Ver información de visita | |
| Entrada | Visita seleccionada. |
| Flujo del evento | 1. El usuario registrado selecciona la opción de ver información de visita. 2. Se abre un panel donde con toda la información asociada a dicha visita encontrada. |

|  |  |
| --- | --- |
| Listar : Modificar visita | |
| Entrada | Visita seleccionada. |
| Flujo del evento | 1. El usuario registrado selecciona la opción de ver información de visita. 2. Se abre un panel donde con toda la información asociada a dicha visita encontrada. 3. El usuario puede entonces modificar ciertos aspectos de la visita, aunque no todos. El usuario, concretamente, puede:    1. Eliminar un lugar asociado a dicha visita, siempre que no se quede sin ningún lugar.    2. Añadir un nuevo lugar, aunque una visita no puede contener más de 10 lugares. |

|  |  |
| --- | --- |
| Crear una visita | |
| Entrada | Datos asociados a la visita:   * Título de la visita * Lugares que visitar * Descripción de la visita * Etiquetas asociadas a la visita * Fecha de la visita |
| Flujo del evento | 1. El usuario registrado selecciona la opción crear una nueva visita, en la ventana de “Mis visitas” del portal. 2. Se abre un panel donde se solicita toda la información asociada a la visita que es necesaria para crearla. 3. El usuario introduce todos los datos asociados a la nueva visita, con dos posibles resultados:    1. Los datos introducidos son todos correctos, con lo que la visita queda guardada y se recargará la página, donde ya aparecerá la nueva visita.    2. Alguno o varios de los datos introducidos no son correctos, por lo que se indicará cuál o cuáles de los mismos no son correctos, y se dará opción de reintentar. |

## CASOS DE USO DEL ADMINISTRADOR

|  |  |
| --- | --- |
| Buscar usuario | |
| Entrada | Texto con la clave a buscar y/u otras opciones. |
| Flujo del evento | 1. El administrador accede a la ventana de “Administración” del menú de navegación del portal. 2. El administrador introduce la cadena de texto que desea buscar en un cuadro de texto habilitado para ello. 3. El sistema busca entre todos los usuarios con dos posibles resultados:    1. Hay coincidencias: se muestran los usuarios con coincidencias encontradas.    2. No hay coincidencias: no se muestra nada y se informa de ello. |

|  |  |
| --- | --- |
| Listar usuarios | |
| Entrada | N/a. |
| Flujo del evento | 1. El administrador accede a la ventana de “Administración” del menú de navegación del portal. 2. El administrador selecciona la opción de listar usuarios. 3. El sistema entonces listará en pantalla todos los usuarios asociados al mismo. |

|  |  |
| --- | --- |
| Ver información de un usuario | |
| Entrada | Usuario seleccionado |
| Flujo del evento | 1. El administrador accede a la ventana de “Administración” del menú de navegación del portal. 2. El administrador busca un usuario manualmente o bien lista todos los usuarios. 3. El administrador selecciona entonces el usuario que desea que se rehabilite y lo selecciona. 4. Se abre un panel con información del usuario, incluyendo toda la información relevante relativa a dicho usuario. |

|  |  |
| --- | --- |
| Deshabilitar usuarios | |
| Entrada | Usuario seleccionado |
| Flujo del evento | 1. El administrador accede a la ventana de “Administración” del menú de navegación del portal. 2. El administrador busca un usuario manualmente o bien lista todos los usuarios. 3. El administrador selecciona entonces el usuario que desea que se elimine y lo selecciona. 4. Se abre un panel con información del usuario, incluyendo una opción para eliminar/deshabilitar usuarios. 5. El administrador pulsa la opción eliminar/deshabilitar usuario. 6. El sistema solicita entonces confirmación al administrador con dos posibles resultados:    1. El administrador confirma la eliminación, con lo que se establece como “deshabilitado” al usuario seleccionado, y a todos los efectos quedará excluido del sistema, aunque permanecerá en él.   (en la lista de usuarios, este aparecerá diferenciado de los demás).   * 1. El administrador cancela la operación, sin verse el sistema ni el usuario afectados. |

|  |  |
| --- | --- |
| Rehabilitar usuarios | |
| Entrada | Usuario seleccionado |
| Flujo del evento | 1. El administrador accede a la ventana de “Administración” del menú de navegación del portal. 2. El administrador busca un usuario manualmente o bien lista todos los usuarios. 3. El administrador selecciona entonces el usuario que desea que se rehabilite y lo selecciona. 4. Se abre un panel con información del usuario, incluyendo una opción para rehabilitar usuarios. 5. El administrador pulsa la opción eliminar/deshabilitar usuario. 6. El sistema solicita entonces confirmación al administrador con dos posibles resultados:    1. El administrador confirma la rehabilitación, con lo que se establece como “habilitado” al usuario seleccionado, y a todos los efectos quedará incluido en el sistema, como si nunca hubiera sido eliminado.    2. El administrador cancela la operación, sin verse el sistema ni el usuario afectados. |

|  |  |
| --- | --- |
| Eliminar comentario de visita | |
| Entrada | Usuario seleccionado |
| Flujo del evento | 1. El administrador accede a la ventana de “Administración” del menú de navegación del portal. 2. El administrador busca un usuario manualmente o bien lista todos los usuarios. 3. El administrador selecciona entonces el usuario que desea que se rehabilite y lo selecciona. 4. Se abre un panel con información del usuario, incluyendo una opción para rehabilitar usuarios. 5. El administrador pulsa la opción eliminar/deshabilitar usuario. 6. El sistema solicita entonces confirmación al administrador con dos posibles resultados:    1. El administrador confirma la rehabilitación, con lo que se establece como “habilitado” al usuario seleccionado, y a todos los efectos quedará incluido en el sistema, como si nunca hubiera sido eliminado.    2. El administrador cancela la operación, sin verse el sistema ni el usuario afectados. |

|  |  |
| --- | --- |
| Eliminar visita de usuario | |
| Entrada | Usuario seleccionado |
| Flujo del evento | 1. El administrador accede a la ventana de “Administración” del menú de navegación del portal. 2. El administrador busca un usuario manualmente o bien lista todos los usuarios. 3. El administrador selecciona entonces el usuario que desea que se rehabilite y lo selecciona. 4. Se abre un panel con información del usuario, incluyendo una opción para rehabilitar usuarios. 5. El administrador pulsa la opción eliminar/deshabilitar usuario. 6. El sistema solicita entonces confirmación al administrador con dos posibles resultados:    1. El administrador confirma la rehabilitación, con lo que se establece como “habilitado” al usuario seleccionado, y a todos los efectos quedará incluido en el sistema, como si nunca hubiera sido eliminado.    2. El administrador cancela la operación, sin verse el sistema ni el usuario afectados. |

|  |  |
| --- | --- |
| Ver estadísticas | |
| Entrada | Usuario seleccionado |
| Flujo del evento | 1. El administrador accede a la ventana de “Administración” del menú de navegación del portal. 2. El administrador busca un usuario manualmente o bien lista todos los usuarios. 3. El administrador selecciona entonces el usuario que desea que se rehabilite y lo selecciona. 4. Se abre un panel con información del usuario, incluyendo una opción para rehabilitar usuarios. 5. El administrador pulsa la opción eliminar/deshabilitar usuario. 6. El sistema solicita entonces confirmación al administrador con dos posibles resultados:    1. El administrador confirma la rehabilitación, con lo que se establece como “habilitado” al usuario seleccionado, y a todos los efectos quedará incluido en el sistema, como si nunca hubiera sido eliminado.    2. El administrador cancela la operación, sin verse el sistema ni el usuario afectados. |

## DIAGRAMAS DE ACTIVIDAD

# ARQUITECTURA DE LA APLICACIÓN

# DISEÑO DE LA APLICACIÓN

# CONCLUSIONES Y TRABAJO FUTURO

# ANEXO

# BIBLIOGRAFÍA

1. Proyecto V8 de Google <https://v8.dev> [↑](#footnote-ref-1)
2. Página web oficial de Node <https://nodejs.org/en/about/> [↑](#footnote-ref-2)
3. Página oficial de Express <https://expressjs.com/es/> [↑](#footnote-ref-3)
4. Enlace al proyecto npm <https://www.npmjs.com/> [↑](#footnote-ref-4)
5. Un framework es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia para enfrentar y resolver problemas de índole similar. [↑](#footnote-ref-5)
6. Datos obtenidos directamente de Github <https://github.com> [↑](#footnote-ref-6)
7. Datos obtenidos de NpmStats https://npm-stat.com/charts.html?package=express&from=2010-05-01&to=2019-01-01 [↑](#footnote-ref-7)
8. Página web del módulo MySQL <https://www.npmjs.com/package/mysql> [↑](#footnote-ref-8)
9. Página web oficial de phpMyAdmin <https://www.phpmyadmin.net/news/> [↑](#footnote-ref-9)
10. Página web oficial de Apache <https://www.apache.org/> [↑](#footnote-ref-10)