13 de FEBRERO de 2023

Cecilia Llamazares López

IES San Andrés del Rabanedo

Villabalter

GREEN ROADS

Página web de rutas

Tabla de contenido

[Introducción 2](#_Toc127811014)

[Lenguajes y recursos 2](#_Toc127811015)

[Aspectos generales de la página 2](#_Toc127811016)

[Paleta de colores 2](#_Toc127811017)

[Función de los elementos 2](#_Toc127811018)

[Landing Page 2](#_Toc127811019)

[Búsqueda de rutas 3](#_Toc127811020)

[Detalles de ruta 5](#_Toc127811021)

[Inicio de sesión 6](#_Toc127811022)

[Registro 7](#_Toc127811023)

[Perfil de usuario 7](#_Toc127811024)

[Edición de perfil 8](#_Toc127811025)

[Subir rutas 9](#_Toc127811026)

[Mis rutas 10](#_Toc127811027)

[Mapa de navegación 11](#_Toc127811028)

[Manual del programador 11](#_Toc127811029)

[JavaScript 11](#_Toc127811030)

[Cabecera y pie de página 11](#_Toc127811031)

[Landing Page 13](#_Toc127811032)

[Búsqueda de rutas 13](#_Toc127811033)

[Detalles de rutas 15](#_Toc127811034)

[Inicio de sesión 17](#_Toc127811035)

[Registro 18](#_Toc127811036)

[Perfil de usuario 20](#_Toc127811037)

[Edición de perfil 22](#_Toc127811038)

[Subir rutas 24](#_Toc127811039)

[Mis rutas 25](#_Toc127811040)

[PHP 26](#_Toc127811041)

[Usuarios 26](#_Toc127811042)

[Editar 28](#_Toc127811043)

[Rutas 30](#_Toc127811044)

[Comentarios 32](#_Toc127811045)

[Bibliografía 35](#_Toc127811046)

# Introducción

Green Roads es una aplicación que permite a los usuarios realizar una búsqueda de rutas., consultar cada uno de sus detalles, comentar acerca de cada ruta y tener a su disposición un mapa con el trayecto marcado de cada ruta.

# Lenguajes y recursos

Se ha hecho uso de los siguientes lenguajes de programación:

* JavaScript
* PHP
* HTML

En cuanto a los recursos utilizados, esta aplicación se ha desarrollado en Visual Studio Code y se ha utilizado una API externa para el uso de los **mapas**: **Leaflet**.

Más adelante se explicará en detalle el funcionamiento de este.

En cuanto a APIS internas se han utilizado cuatro, las cuales están explicadas en el **Manual del programador**.

Algunas de las imágenes utilizadas han sido sacadas de **Pixabay**.

# Aspectos generales de la página

## Paleta de colores

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

# Función de los elementos

Green Roads se divide en varias páginas web, cada una con una función en específico.

Todas las páginas están formadas por la misma cabecera y el mismo pie de página.

## Landing Page

La página principal o “*Landing Page*” es la primera que ve el usuario al entrar en la aplicación.

Está formada por una frase y una imagen. Esta frase incita a los usuarios a pulsar en ella, cuando el usuario coloca el cursor encima, esta aumenta de tamaño.



La parte de debajo de la página de inicio está formada por unas rutas destacadas y otra frase que acentúa el motivo de la página.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

## Búsqueda de rutas

La búsqueda de rutas contiene todas las rutas disponibles en la página web. Tiene dos tipos de vista: vista en lista y en mapa. En el menú superior se puede seleccionar qué tipo de vista desea ver el usuario, así como buscar una ruta en concreto.

También contiene todas aquellas rutas destacadas o cercas del usuario junto con un número indicador de las rutas totales en ese apartado.

Debajo de las rutas cercanas se encuentra una lista con todas las rutas, donde el usuario podrá buscar por nombre de ruta, dificultad o usuario que la publicó.

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Desde esta página se puede acceder a detalles de ruta pulsando en **Ver más** disponible en cada tarjeta de cada ruta.

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

## Detalles de ruta

Aquí el usuario podrá ver todos los detalles de una ruta, ya sea su dificultad, el tipo de recorrido, la distancia, el tiempo total, las coordenadas y su tramo. En la parte de debajo se encuentra la sección de comentarios donde, si se ha iniciado sesión, se podrá dejar un comentario sobre la ruta.

Mapa

Descripción generada automáticamente

Mapa

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

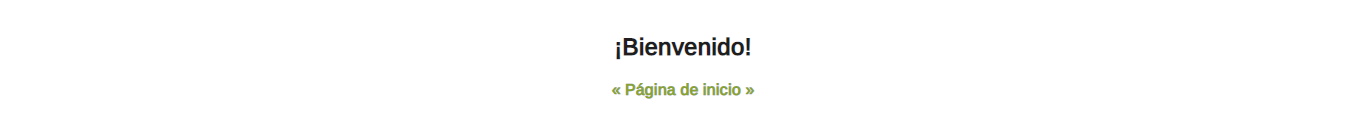
## Inicio de sesión

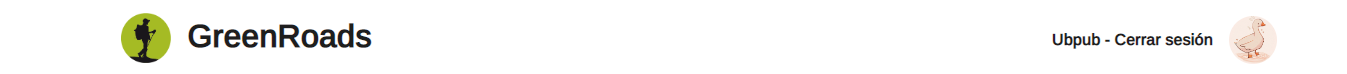
Se accede desde la cabecera, en el botón de **Iniciar sesión**. Aquí el usuario puede acceder a la página a través de su cuenta para poder publicar rutas y dejar comentarios. Es un pequeño formulario donde debe introducir su nombre y su contraseña. Desde esta página también puede acceder a la página de registro desde el enlace **Registrarse** o desde la cabecera como en todas las partes de la página.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una vez iniciada la sesión, la cabecera cambiará de apariencia para mostrar el nombre y la foto del usuario (de normal no es posible tenerla, solo hay tres usuarios que la tienen).







Si el usuario pulsa en la parte roja de la siguiente foto cerrará la sesión. Si pulsa en la parte verde, su foto de usuario, entrará a su página de perfil.

Forma

Descripción generada automáticamente con confianza media

## Registro

Al igual que en el inicio de sesión, se puede acceder a **Iniciar sesión** desde el enlace debajo de **Crear cuenta**.

El usuario tendrá que rellenar un formulario con sus datos personales para poder registrarse y podrá dejar el campo de las actividades favoritas vacío para rellenarlo más tarde.

Interfaz de usuario gráfica

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Perfil de usuario

Pulsando en su foto de perfil se puede acceder a la visualización del perfil del usuario. En esta aparecerá el nombre de usuario y su correo electrónico junto con dos botones: **Mis rutas** y **Subir ruta**.

También un enlace debajo de la foto de perfil para eliminar la cuenta, en el que aparecerá una ventana donde se deberá introducir la contraseña de la cuenta para poder eliminarla.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Edición de perfil

Se accede desde la parte superior de la página del perfil, en un icono de un lápiz.

Imagen que contiene Gráfico

Descripción generada automáticamente

Los campos que rellenar ya aparecerán con la información del usuario. Sólo deberá sobrescribirlos en caso de que quiera cambiar alguno. Para que se guarde la información, el usuario deberá introducir su contraseña en el último campo.

Toda la información deberá ser válida, y si la contraseña no coincide, el usuario no podrá editar su perfil y tendrá que volver a escribir su contraseña.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Subir rutas

En caso de encontrarse en la página anterior, se puede acceder desde la parte de arriba, donde pone **Volver** a la página del usuario.

Forma, Rectángulo

Descripción generada automáticamente

También se puede volver desde estos dos puntos. En el perfil de usuario y búsqueda de rutas:

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Si se accede desde búsqueda de rutas el usuario deberá haber iniciado sesión primero. Si no ha iniciado sesión con ninguna cuenta le redirigirá a la página de inicio de sesión.

En subir ruta el usuario deberá indicar el nombre, la descripción, la dificultad y el tipo de ruta, a su vez deberá incluir un fichero GPX con los datos de la ruta y sus tramos. Con estos campos rellenados podrá subir la ruta.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Mis rutas

El usuario podrá acceder a una página donde aparecerán todas aquellas rutas que él haya subido, aparecerán tanto en modo de lista como en mapa, y se podrá buscar una ruta por su nombre.

Imagen que contiene Código QR

Descripción generada automáticamente

# Mapa de navegación

Diagrama

Descripción generada automáticamente

# Manual del programador

## JavaScript

### Cabecera y pie de página

Se utiliza JavaScript en la página inicial como en todas las demás para obtener la cabecera y el pie de página. En el HTML hay dos contenedores para incluirlos:



En este JavaScript comprueba primero si hay una sección activa para cambiar entre cabeceras. En caso de que haya una sesión activa seleccionará la cabecera que accederá a la información del usuario.

En caso contrario, seleccionará como cabecera aquella que contiene los botones de registro e inicio de sesión.

Esta sesión se comprueba si en ***LocalStorage*** se encuentra almacenado un token.

Texto

Descripción generada automáticamente

Obtiene la cabecera con los id y las vistas correspondientes y las crea.

Texto

Descripción generada automáticamente

Dentro del método **obtenerHeaderFooter** hace una petición a la vista correspondiente pasada por parámetro.

Dentro de la petición comprueba si se le ha pasado una cabecera o un pie de página gracias a un booleano que se le pasó por parámetro en su llamada. Si es una cabecera comprueba si se ha iniciado sesión y le asigna una cabecera dependiendo de la respuesta.

Texto

Descripción generada automáticamente

Finalmente añade un evento al botón de cerrar sesión en la cabecera 2 para eliminar todos los ítems del LocalStorage.

Texto

Descripción generada automáticamente

### Landing Page

La página inicial utiliza JavaScript para obtener las rutas destacadas, que en este caso son las cuatro primeras. Las variables importantes que no aparecen en la captura son **numRutas** que vale 0 y **maxRutas** que vale 4. Manda una petición a la API de las rutas y por cada elemento obtenido va sumando uno al número de Rutas y creando un contenedor con su información. Si el número de rutas llega a lo que vale el máximo de rutas deja de crear contenedores.

Texto

Descripción generada automáticamente

### Búsqueda de rutas

La página de búsqueda de rutas está diseñada dinámicamente con JavaScript, por lo que este documento es más largo que los otros.

Texto

Descripción generada automáticamente

El evento de la ventana sirve para cambiar el contenido de un botón cuando la ventana sea pequeña. Esto también se puede realizar mediante CSS en una media-query cambiando el atributo *content*.

Texto

Descripción generada automáticamente

Obtiene todas las rutas mediante una petición al API de rutas y por cada elemento crea una ruta.

Texto

Descripción generada automáticamente

Una función similar está en los filtros de estas rutas, en la que al pulsar el botón de búsqueda se vacían las rutas anteriores y se obtienen las rutas que coincidan con aquellos campos.

Comprueba si están vacíos los campos y va añadiendo a la URL de la petición aquellos que no estén vacíos. Vacía el contenedor con todas las rutas y de nuevo hace una petición al API de rutas para hacer lo mismo que en la imagen anterior.

Texto

Descripción generada automáticamente

Para acceder a la vista de detalles de rutas mediante un enlace, añade un evento en el que impide que ese enlace le lleve a donde tiene su href sin antes obtener el id de la ruta. Finalmente, envía al usuario a la ventana de detalles pasando como parámetro en la URL el id de la ruta seleccionada.

Texto

Descripción generada automáticamente

### Detalles de rutas

En los detalles se obtiene la información de la ruta mediante PHP, pero se obtiene el mapa y los comentarios mediante JavaScript. Se ha utilizado Leaflet como mapa.

Con PHP obtenemos las variables de la **longitud de comiento** y la **latitud de comienzo** para colocar el marcador en el mapa. Si no tienen coordenadas de comienzo se le asigna las coordenadas de la ciudad de León.

Se crea el mapa con la vista centrada en las coordenadas de comienzo con un zoom de 15. Se dibuja una línea con los puntos obtenidos de la ruta en caso de que tenga de un color verde y se añade al mapa.

Texto

Descripción generada automáticamente

Finalmente hacemos un control de escala y creamos un marcador personalizado para añadirlo en las coordenadas de comienzo en el mapa.

Texto

Descripción generada automáticamente

En la parte de comentarios primero se comprueba si se ha iniciado la sesión para poder mostrar el cuadro de escribir un nuevo comentario. En caso de que no se haya iniciado sesión se moctrará un enlace para iniciar sesión.

Texto

Descripción generada automáticamente

Vuelve a comprobar si se ha iniciado la sesión y recoge el valor del texto para añadirlo a un objeto que tiene el nombre de usuario, el comentario y el id de la ruta. El nombre del usuario se obtiene el *LocalStorage*.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Continua realizando una petición al API de los comentarios pasando el objeto en el cuerpo, y, si todo sale bien, el comentario se habría publicado correctamente y volvería a la página de inicio.

Texto

Descripción generada automáticamente

### Inicio de sesión

Cuando se pulsa el botón de iniciar sesión recoge los valores del formulario: el usuario y la contraseña. Realiza una petición al API de usuarios y si la respuesta sale bien, la devuelve como JSON. En caso contrario aparecerá una alerta en la página con el error.

Texto

Descripción generada automáticamente

Ahora se comprueba que haya datos, y si los hay añade algún ítem al *LocalStorage*. También comprueba si el usuario tiene una imagen almacenada en la base de datos para asignársela.

Con esto el usuario ya habría iniciado la sesión, mostrando una nueva ventana de **Inicio de sesión correcto**.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

### Registro

El registro tiene varios patrones creados, para la fecha, el email y la contraseña. Comprueba primero de todo que no se haya iniciado sesión para que pueda aparecer la página de registro.

Texto

Descripción generada automáticamente

Comprueba si los campos están vacíos con el siguiente código:

Texto

Descripción generada automáticamente

Llamando a la función **emptyFields** va comprobando campo a campo si está vacío y, en caso de que esté, le cambia el estilo a rojo para que el usuario sepa que hay que rellenarlo. Finalmente devuelve la variable **vacío**, que tiene almacenado el nombre del campo vacío.

Texto

Descripción generada automáticamente

Después de esto, comprueba si los campos están vacíos, y si no lo están recoge todos los valores del formulario, así como las actividades, en la que hace falta llamar a la función **obtenerActividades**. En esta función comprueba los campos seleccionados y los añade a un array. Finalmente lo devuelve como una cadena separada por espacios.

Texto

Descripción generada automáticamente

Con los valores recogidos va comprobando con ayuda de los patrones si están bien escritos los campos de **email**, **contraseña** y **fecha de nacimiento**.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Si va todo correcto genera un JSON con los datos introducidos gracias al método **gererateJSON** y lo devuelve.

Texto

Descripción generada automáticamente

Finalmente hace una petición al API de usuarios pasando el JSON generado en el cuerpo, y, si se registra correctamente aparecerá una nueva ventana de **Registrado correctamente** y volverá al inicio para poder iniciar sesión.

Texto

Descripción generada automáticamente

### Perfil de usuario

Crea un objeto para almacenar le¡¡el usuario obtenido mediante *LocalStorage* y la función **obtenerUsuario**.

Texto

Descripción generada automáticamente

Haciendo una petición al API de usuarios pasando el nombre de usuario como parámetro en la URL de la petición, almacena los datos del usuario obtenido en el objeto creado anteriormente. Finalmente escribe el nombre de usuario y el correo en los campos del HTML. En caso de que no se hayan encontrado datos escribirá que no se ha encontrado un usuario.

Texto

Descripción generada automáticamente

Se puede eliminar la cuenta del usuario pulsando en el enlace de eliminar. A este enlace se le ha añadido un evento en el que al pulsar aparece una ventana emergente con un campo para poner la contraseña y un botón de eliminar. Cuando se pulsa eliminar se obtienen los valores del formulario: la contraseña; y el id del usuario.

Texto

Descripción generada automáticamente

Comprueba que la contraseña no se ha dejado en blanco para seguir adelante y procede a hacer una petición al API de usuarios para eliminar el usuario, pasando sus datos en el cuerpo de la petición.

Texto

Descripción generada automáticamente

Con la respuesta, si todo ha ido bien, elimina los ítems del *LocalStorage* y vuelve a la página de inicio con una alerta diciendo **Cuenta eliminada correctamente**. Si ha ido mal puede ser porque la contraseña está mal escrita o porque ha ocurrido algún error en la petición.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

El método **writeAlert** simplemente crea una alerta para mostrar en la pantalla en caso de que algo haya ido mal.

Texto

Descripción generada automáticamente

### Edición de perfil

Aquí realiza lo mismo que al registrar el usuario. Crea unos patrones para el **email** y la **fecha de nacimiento**. Obtiene el usuario con el *LocalStorage* y con estos datos obtenidos de la petición los escribe en el formulario.

Añade un evento al botón de editar en el que comprueba que la contraseña no se ha dejado vacía. Si se ha dejado vacía saltará una alerta que indicará al usuario que la contraseña es necesaria para poder editar el perfil.

Si no está vacía, recoge los valores del formulario de la misma manera que en el registro de usuario y comprueba que no hay ningún campo vacío. En caso de que no haya ningún campo vacío comprobará con los patrones al igual que en el registro los campos de **email** y **fecha de nacimiento** y, si todo está correcto, añade las actividades de la misma manera.

Crea un JSON con los valores del formulario y las actividades recogidas.

Texto

Descripción generada automáticamente

Realiza una petición al API de editar pasándole en el cuerpo el JSON del usuario obtenido.

Texto

Descripción generada automáticamente

Dependiendo de los códigos que haya devuelto, indicará si ha salido bien o mal. Si ha salido bien, aparecerá otra ventana indicando que se ha editado correctamente y volverá a la página de edición de perfil con los campos ya actualizados. Si no, la contraseña estará mal escrita, por lo que se mostrará una alerta indicándolo.

Texto

Descripción generada automáticamente

### Subir rutas

Para subir las rutas es necesario un fichero JavaScript adicional: GXSparser.js que se puede sacar de GitHub.

En la página de subir rutas hay varios botones para marcar la dificultad, y cuando seleccionas uno, todos cambian de estilo para indicar cuál se ha seleccionado.

Esto se hace recogiendo los botones y con un bucle añadirle a cada uno un evento onclick. Esta quita a todos los botones la clase “selected” y añade al botón seleccionado esta misma clase.

Texto

Descripción generada automáticamente

Añade un evento “**submit**” al formulario en el que primero se previene de que realice su función principal y se rediriga a la página que hay en el *action*. Primero recoge el fichero gpx subido y lo lee cargándolo con un FileReader. Con ayuda del gpxParser lo parsea a GPX y lee los puntos de este.

Texto

Descripción generada automáticamente

Recorre cada ítem dentro de puntos del JSON obtenido y va añadiendo a un array la latitud y la longitud.

Texto

Descripción generada automáticamente

Finalmente crea un array con la siguiente estructura. La distancia, como la devuelve como un número entero en metros y lo queremos mostrar en kilómetros, lo dividimos entre 1000 y lo redondeamos a dos decimales para que no sea una cifra muy larga.

Para guardar los puntos hace falta pasarlos como una cadena.

Captura de pantalla de computadora

Descripción generada automáticamente

Hace una petición al API de rutas pasando en el cuerpo de esta el JSON creado anteriormente. Si sale bien la vista de la página cambia a una de **Subido correctamente**.

Texto

Descripción generada automáticamente

Finalmente, fuera del onload del reader, añadimos lo siguiente para que lo lea como texto y parsee las etiquetas XML y GPX.

Texto

Descripción generada automáticamente

### Mis rutas

Mis rutas tienen la misma función que la búsqueda de rutas, salvo que esta vez se obtienen las rutas por el nombre de usuario.

Texto

Descripción generada automáticamente

Todo lo demás es igual a la búsqueda de rutas, aunque tenga un diseño un tanto diferente, el JavaScript no cambia mucho más.

## PHP

### Usuarios

La API de usuarios consta de dos peticiones, una siendo tipo “POST” y otra de tipo “GET”.

http://localhost/GreenRoads/api/usuarios.php

#### POST

Esta petición tiene que recibir en el cuerpo un JSON con los datos del usuario que se desea insertar. El JSON que se envía tiene la siguiente estructura:

return user = {

        'nombre': nombre,

        'usuario': usuario,

        'correo': correo,

        'contrasena': contrasena,

        'nacimiento': nacimiento,

        'estatura': estatura,

        'peso': peso,

        'actividades': actividades,

    };

Esta petición procesa el JSON enviado y cifra la contraseña pasada. Crea una consulta donde selecciona todos los usuarios de la taba “usuarios” que coincidan con el nombre de usuario.

Texto

Descripción generada automáticamente

Con esta consulta comprueba si el usuario existe. En caso de que no exista crea una sentencia SQL para insertarlo con todos los campos y la ejecuta, devolviendo un 201 *Created* en la cabecera.

Si hubiese encontrado un usuario en la primera consulta pasaría a devolver directamente el código de error 409 *Confilct*, por lo que no podría insertarlo.

Si hubiese ocurrido que no llegase a ejecutarse la petición, directamente devolvería el código de error 400 *Bad Request*.

Texto

Descripción generada automáticamente

#### GET

Esta petición puede no recibir nada, que devolvería todos los usuarios existentes, o puede recibir tanto el id como el nombre de usuario o usuario y contraseña.

Obtener todos los usuarios

http://localhost/GreenRoads/api/usuario.php

Obtener un usuario por id

http://localhost/GreenRoads/api/usuario.php?id=$id

Buscar usuarios por su id y su nombre de usuario

http://localhost/GreenRoads/api/usuario.php?id=$id&usuario=$usuario

Comprobar si existe usuario en la base de datos (registro)

http://localhost/GreenRoads/api/usuario.php?usuario=$usuario

Comprobar si el login es correcto

http://localhost/GreenRoads/api/usuario.php?usuario=$usuario&pass=$pass

Genera una sentencia SQL y hace una consulta en la tabla “usuarios” dependiendo de los datos enviados.

La contraseña, como en todas las peticiones y subida de contraseñas a una base de datos, se cifra con sha512 y se añade a la consulta.

No se puede enviar la contraseña sola en la petición ya que no sabría que buscar. Esta petición se utiliza sobre todo para el inicio de sesión y para comprobar si existe el usuario en la base de datos a la hora de registrarse.

Captura de pantalla de computadora

Descripción generada automáticamente

En el caso de inicio de sesión, éste genera un payload con una key declarada fuera de las peticiones. Este payload sirve para generar un web token, con el nombre del usuario en este caso, para que mantenga la sesión iniciada aun recargando la página.

Una vez generado el web token lo añade al usuario obtenido y lo devuelve, enviando a la cabecera el código 200 *OK* si todo ha salido bien.



Texto

Descripción generada automáticamente

Si al ejecutar la consulta no consigue encontrar ningún usuario que coincida con los criterios de búsqueda, enviará el código de error 409 *Conflict*, y en el mismo caso que el anterior, si hubo algún problema con la petición o el servidor, enviará un 400*Bad Request*.

### Editar

A diferencia del anterior, esta API tiene un método “POST” y otro método “DELETE”.

http://localhost/GreenRoads/api/editar.php

#### POST

Este método se utiliza para editar el perfil, actualizar sus campos en la tabla.

Recibe un JSON al igual que el anterior, con los mismos datos, de la misma estructura. Cifra la contraseña y realiza una consulta donde selecciona todos aquellos usuarios que coincidan con el nombre de usuario y la contraseña enviadas. Si el resultado no devuelve nada devolverá el código de error 404 *Not Found*, no encontrando el usuario que se ha enviado.

Texto

Descripción generada automáticamente

En caso de que sí se encuentre el usuario significa que los campos están bien y genera una consulta SQL de UPDATE con los campos enviados en el JSON.

Si todo ha ido bien, los campos del usuario se habrán actualizado correctamente y nos devolverá un 201 *Created*. Devuelve el JSON del usuario actualizado.

Texto

Descripción generada automáticamente

Si algo ha ido mal devolverá un 400 *Bad Request*.

#### DELETE

Esta petición se utiliza para borrar la cuenta del usuario, en la que deberá indicar su contraseña.

Se obtiene un JSON de la petición y comprueba que no esté vacío o que se haya recibido correctamente.

Cifra la contraseña y crea una consulta SQL donde elimina al usuario que coincida con su id y la contraseña cifrada. Si se ejecuta sin problemas devuelve un 200 *OK*.

Si no ha podido ejecutar la consulta devuelve un 404 *Not Found* porque no se ha encontrado al usuario que desea eliminar, por lo que alguna de las credenciales que ha tenido que rellenar no coinciden con las suyas.

Captura de pantalla con la imagen de una pantalla

Descripción generada automáticamente con confianza media

### Rutas

Esta API tiene dos métodos, también de “POST” y “GET”.

http://localhost/GreenRoads/api/rutas.php

#### POST

Recibe un JSON con los datos de la ruta de la siguiente estructura:

Texto

Descripción generada automáticamente

Crea una sentencia SQL de insertar con cada campo pasado en el JSON. Si se ejecuta sin ningún problema devuelve un 200 *OK*. En caso contrario devuelve el código de error 400 *Bad Request*, por algún fallo a la hora de hacer la petición o ejecutar la sentencia.

Este método se utiliza para subir una ruta a la página web en el formulario de **subir rutas**.

Pantalla de computadora

Descripción generada automáticamente con confianza media

#### GET

Se le puede pasar la id de la ruta, el usuario, el nombre de la ruta o la dificultad de la ruta por parámetro.

Obtener todas las rutas

http://localhost/GreenRoads/api/rutas.php

Obtener una ruta por id

http://localhost/GreenRoads/api/rutas.php?id=$id

Obtener una ruta en específico de un usuario

http://localhost/GreenRoads/api/rutas.php?id=$id&usuario=$usuario

Obtener las rutas de un usuario

http://localhost/GreenRoads/api/rutas.php?usuario=$usuario

Estas combinaciones pueden ir por separado o juntas y en cualquier orden.

Obtener las rutas por su nombre

http://localhost/GreenRoads/api/rutas.php?nombre\_ruta=$nombre

Obtener las rutas por su dificultad

http://localhost/GreenRoads/api/rutas.php?dificultad=$dificultad

En el caso de buscar por el nombre de la ruta, buscará aquellas que contengan la cadena pasada en la ruta. Por ejemplo, si escribe “Ruta” en el nombre\_ruta, buscará a todas aquellas rutas que en su nombre contenga la palabra “Ruta”. No distingue entre mayúsculas ni minúsculas.

Crea una sentencia SQL donde selecciona todas las rutas, dependiendo de lo que se le envíe en la petición va añadiendo más cosas a la consulta o no.

Si no se envía nada más en la URL obtendrá todas las rutas.

Si se envía el ID de la ruta obtendrá una ruta en específico. Esta se utiliza para mostrar el detalle de rutas, donde se le pasa un ID a la petición dependiendo de la ruta que se quiera mostrar.

Si se envía el usuario obtendrá todas las rutas de ese usuario, el usuario es el **nombre de usuario**, no el ID.

Si se envía el id y el usuario obtendrá una ruta en específico de un usuario en específico.

Texto

Descripción generada automáticamente

Si se ejecuta bien devuelve un 200 *OK*, si no, devuelve un 404 *Not Found*.

### Comentarios

Los comentarios, al igual que los anteriores, tienen dos métodos. GET para obtenerlas y POST para publicarlos.

#### GET

Obtiene todos los comentarios según los parámetros pasados en la URL.

Obtener el comentario por su id de ruta

http://localhost/GreenRoads/api/comentario.php?id\_ruta=$id

Obtener el comentario por su id

http://localhost/GreenRoads/api/comentario.php?id=$id

Obtener el comentario por usuario

http://localhost/GreenRoads/api/comentario.php?usuario=$usuario

Para cada ruta corresponde una serie de comentarios, que viene indicado en su id\_ruta.

Realiza una consulta de selección de todos los comentarios y va añadiendo campos a esta.

Devuelve un 200 *OK* si todo sale bien y un 404 *Not Found* si no ha encontrado ningún comentario que coincida con el criterio de búsqueda.

Termina devolviendo el/los comentario/s obtenido/s en caso de que todo haya salido bien.

Texto

Descripción generada automáticamente

#### POST

El método post recibe un JSON de la siguiente estructura:

Texto

Descripción generada automáticamente

Obtiene el JSON enviado y crea una consulta de inserción con los tres valores. Si sale bien devuelve como los anteriores, un 200 *OK*, y si sale mal un 400 *Bad Request*. Al final del todo devuelve el comentario subido a la base de datos si se ha realizado con éxito.

Texto

Descripción generada automáticamente

# Bibliografía

* [Página de imágenes](https://pixabay.com/es/)
* [Documentación de Leaflet](https://leafletjs.com/examples.html)
* [Iconos de Bootstrap](https://icons.getbootstrap.com/)