



13 DE FEBRERO DE 2023

GREEN ROADS

PÁGINA WEB DE RUTAS

CECILIA LLAMAZARES LÓPEZ

IES SAN ANDRÉS DEL RABANEDO
Villabalter



Tabla de contenido

Introducción	2
Lenguajes y recursos	2
Aspectos generales de la página	2
Paleta de colores	2
Función de los elementos	2
Landing Page	2
Búsqueda de rutas.....	3
Detalles de ruta	5
Inicio de sesión	6
Registro	7
Perfil de usuario	7
Edición de perfil.....	8
Subir rutas	9
Mis rutas.....	10
Mapa de navegación	11
Manual del programador	11
JavaScript.....	11
Cabecera y pie de página	11
Landing Page	13
Búsqueda de rutas.....	13
Detalles de rutas.....	15
Inicio de sesión	17
Registro	18
Perfil de usuario	20
Edición de perfil.....	22
Subir rutas	24
Mis rutas.....	25
PHP	26
Usuarios.....	26
Editar	28
Rutas.....	30
Comentarios	32
Bibliografía	35

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**.

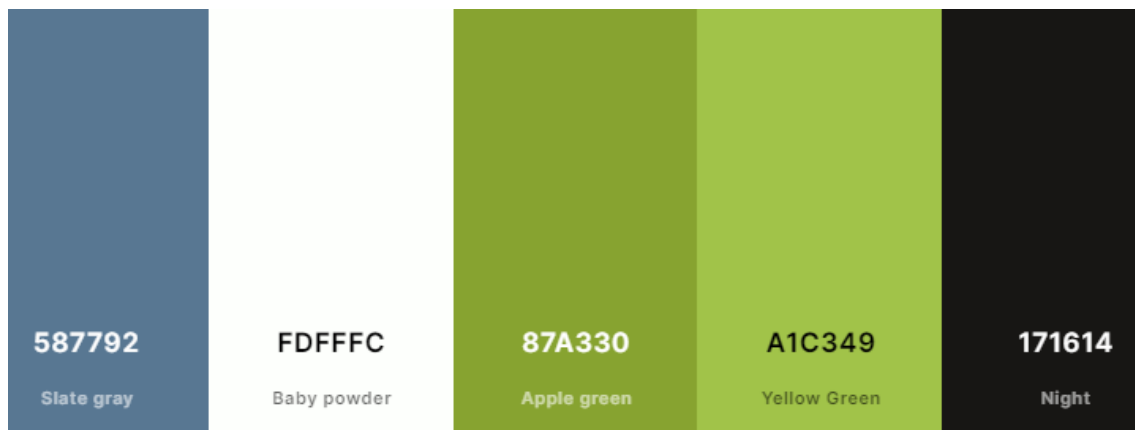
Más adelante se explicará en detalle el funcionamiento de este.

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



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.



GreenRoads

Regístrate

Iniciar Sesión

**¿Quieres
salir a
explorar?**



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.

Dificultad: -
Longitud: -
Tiempo: -

Dificultad: -
Longitud: -
Tiempo: -

Dificultad: -
Longitud: -
Tiempo: -

Dificultad: -
Longitud: -
Tiempo: -

Sal a pasear

Contigo hasta por las montañas más altas
y los caminos más difíciles

Explorar

Rutas

Detalles de las Rutas

Mapa

Sobre GreenRoads

Sobre nosotros

Ayuda

Miembros de la comunidad



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.



GreenRoads

[Registrarse](#)[Iniciar Sesión](#)[Rutas](#) | [Mapa](#)[Subir ruta](#)

Mejores rutas de senderismo

4 rutas

[Ver más](#)

Cerca de ti

4 rutas

[Ver más](#)

Filtros de búsqueda

[Buscar](#)

Ruta de los Calderones

País - Comunidad - Ciudad

**Distancia**
12270 km**Dificultad**
--

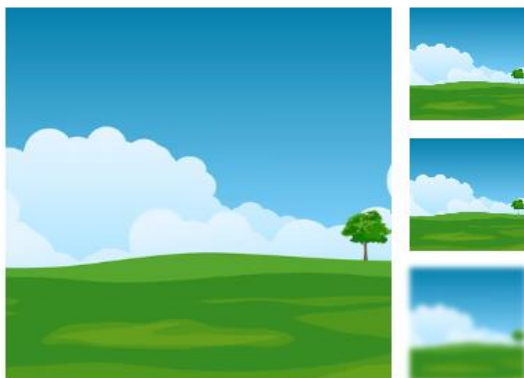
0 | ★ 5

La ruta se inicia en Piedrasecha, para tomar enseguida una vereda casi paralela al río. Destaca una gran roca silícica, muy llamativa por los líquenes amarillentos que la colonizan; es El Serrón. Pron...

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

Ruta de los Calderones

País - Comunidad - Ciudad

**Distancia**
12270 km**Dificultad**
--


0 | ★ 5

La ruta se inicia en Piedrasecha, para tomar enseguida una vereda casi paralela al río. Destaca una gran roca silícica, muy llamativa por los líquenes amarillentos que la colonizan; es El Serrón. Pron...

[Ver más](#)


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.


GreenRoads

[Registrarse](#)
[Iniciar Sesión](#)

[Página de inicio](#)



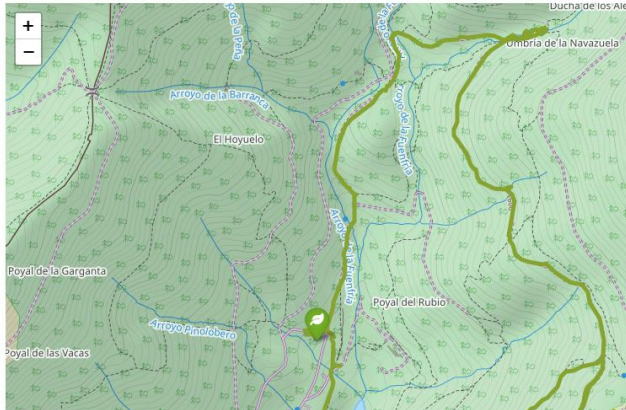
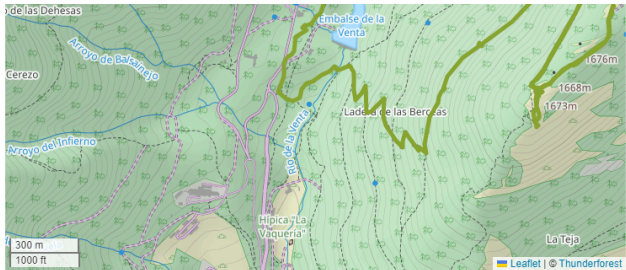
Mirador de los poetas
 He realizado esta ruta

0 | ★ 5

Distancia	Dificultad	Tipo de ruta
11419	Normal	Recta

Tiempo	Coordenadas
11419	40.7707

Desde Madrid por la A-6 salida 47 Dirección Guadarrama cogemos una gran recta pasamos una rotonda y en la siguiente rotonda tercera salida dirección a el Puerto del León cruzamos el Pueblo y después de salir de una curva a izquierdas la segunda calle a la derecha que es la Calle Doctor Gómez Ruiz la M-622 que no la dejamos hasta llegar a Cercadilla cruzamos el túnel del tren y justo en la segunda curva tenemos un Stop. Cogemos la carretera de las dehesas M-966 todo recto hasta un bifurcación que nosotros tomaremos a la derecha a la izquierda va a El hospital de La Fuenfría. Tenemos dos Parking a la derecha ó el ultimo que es muy Pequeñín justo en el último giro a la izquierda enfrente de la fuente de Majavilán; siempre tenemos un buen chorro que podemos aprovechar para llenar nuestras cantimploras.

Comentarios

Ubpub
 Mi comentario en la ruta mirador de los poetas

Comentarios

Ubpub
 Mi comentario en la ruta mirador de los poetas


Admin
 El administrador aprueba esta ruta con buena puntuación.

[» Inicia sesión para escribir comentarios «](#)

[Explorar](#)

[Rutas](#)
[Detalles de las Rutas](#)

[Mapa](#)

[Sobre GreenRoads](#)

[Sobre nosotros](#)
[Ayuda](#)

[Miembros de la comunidad](#)





Comentarios



Ubpub

Mi comentario en la ruta mirador de los poetas



Admin

El administrador aprueba esta ruta con buena puntuación.



Añadir comentario como (Ubpub)

[Enviar comentario](#)

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.



GreenRoads

[Registrarse](#)[Iniciar Sesión](#)

Iniciar sesión

¿Todavía no tienes una cuenta? [Regístrate](#)☐ Recordarme[Iniciar sesión](#)

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).

¡Bienvenido!

[« Página de inicio »](#)

GreenRoads

Ubpub - Cerrar sesión

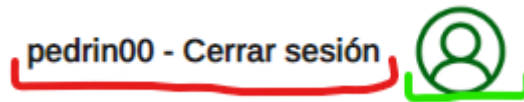


GreenRoads

pedrin00 - Cerrar sesión




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.



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.

 **GreenRoads** Registrarse Iniciar Sesión

Crear cuenta

¿Ya tienes una cuenta? [Iniciar sesión](#)

Actividades favoritas

Correr: ☐

Senderismo: ☐

Ciclismo: ☐

Alpinismo: ☐

Paseo: ☐

Kayak: ☐

Registrarme

Al seguir utilizando GreenRoads, aceptas nuestras [Condiciones de Uso](#) y [Política de Privacidad](#).

Explorar

Rutas

Mapa

Detalles de las Rutas

Sobre nosotros

Miembros de la comunidad

Sobre GreenRoads

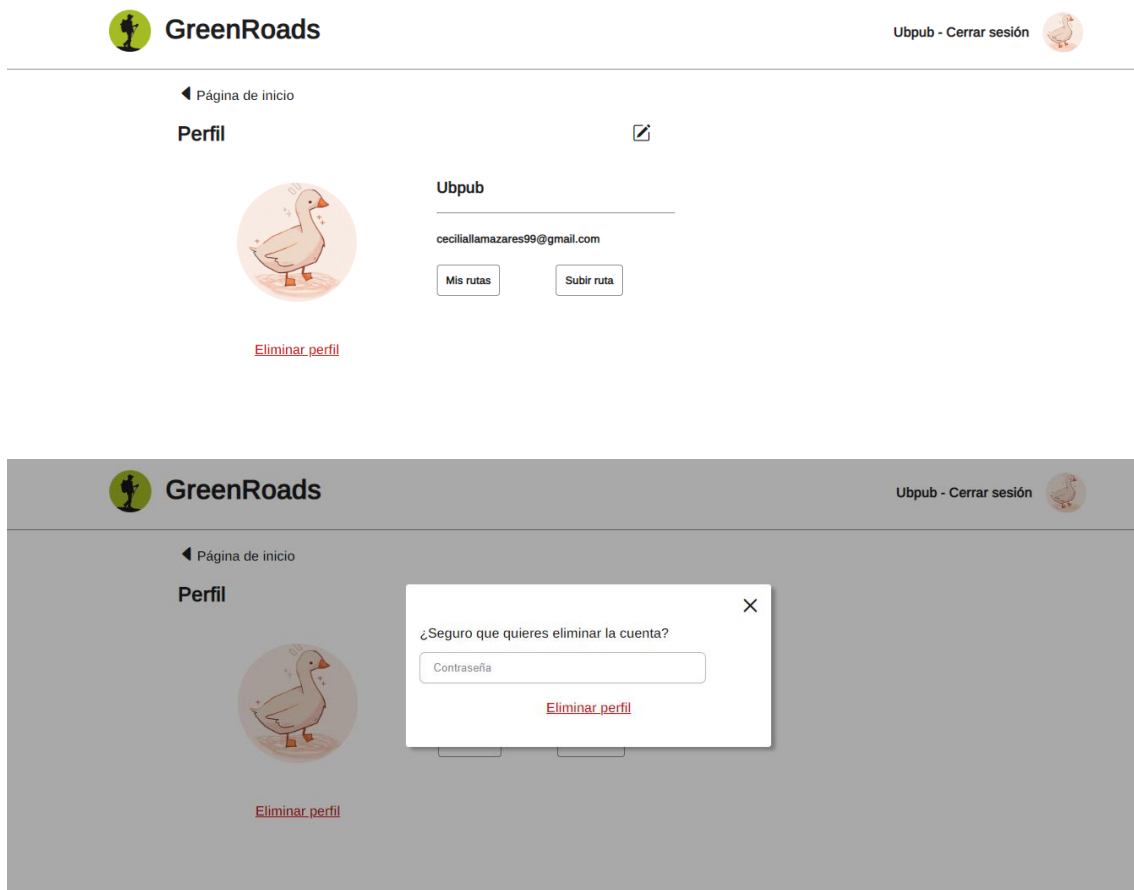
Ayuda



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.



Edición de perfil

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

Perfil



Ubpub

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.

[◀ Volver](#)

Información del perfil



Nombre completo:

Cecilia Llamazares López

Nombre de usuario:

Ubpub

Correo electrónico:

ceciliaallamazares99@gmail.com

Estatura (cm):

160

Peso (kg):

52

Fecha de nacimiento:

2002-03-21

Actividades favoritas:

Correr: ☒ Senderismo: ☒ Ciclismo: ☐Alpinismo: ☐ Paseo: ☒ Kayak: ☐

Introduce tu contraseña:

Contraseña

Guardar cambios

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.

[◀ Volver](#)

Información del perfil

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

[◀ Página de inicio](#)

Perfil



Ubpub

ceciliaallamazares99@gmail.com

Mis rutas

Subir ruta

[Eliminar perfil](#)

[Rutas](#) | [Mapa](#)

Buscar rutas...

 Subir ruta

Mejores rutas de senderismo

4 rutas



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.

[◀ Volver](#)**Subir ruta**

Nombre:

Descripción:

Dificultad:

Fácil	Normal	Difícil	Muy difícil
-------	--------	---------	-------------

Tipo de ruta:

Fichero GPX:

Subir ruta

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.

[◀ Página de inicio](#)**Mis rutas**[Rutas](#) | [Mapa](#)

Buscar rutas...

Subida al Vizcodillo

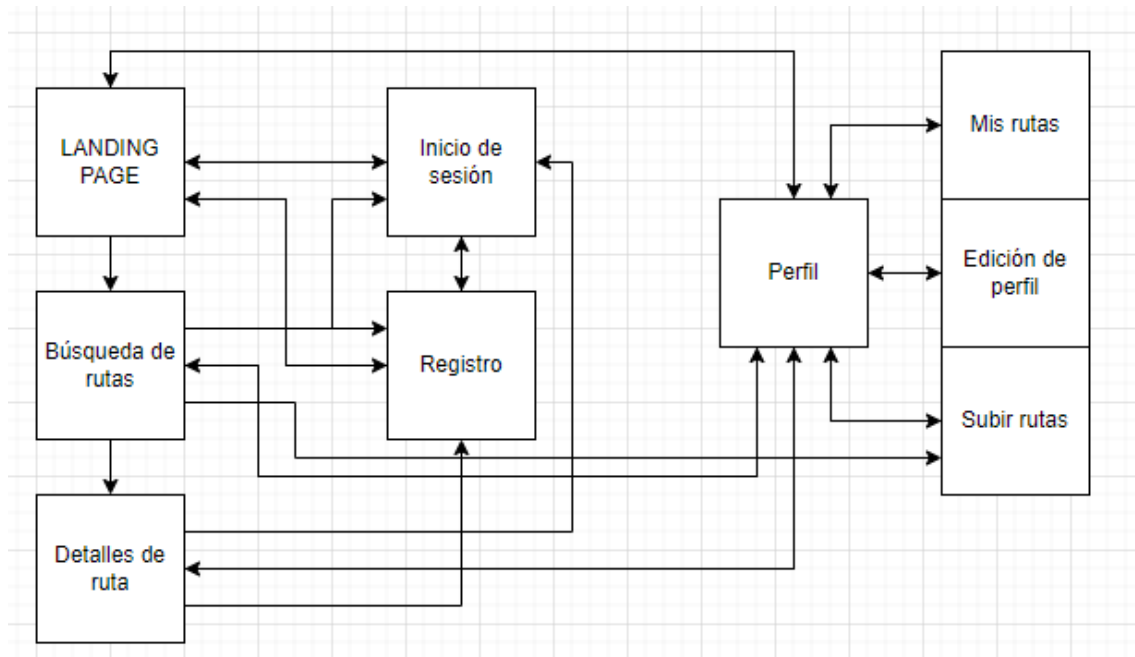
País - Comunidad - Ciudad



A 1,5km pasando Truchillas desde Truchas hay un parking con un panel indicativo para subir al lago Truchillas, hasta el cual llego y atraveso por la salida del agua que forma el río Truchillas, para su...

[Ver ruta](#)[Editar ruta](#)

Mapa de navegación



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:

```

<!-- HEADER --> <!-- FOOTER -->
<div id="index-header"></div> <div id="index-footer"></div>
  
```

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.

```
// Comprueba con los items de localStorage si hay una sesión activa
function checkLocalStorage() {
  let header = "nulo";
  if (localStorage.getItem('webToken') != null) {
    header = "../views/included-views/headerlog.html";
  } else {
    header = "../views/included-views/header.html";
  }
  return header;
}
```

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

```
let header_id = '#index-header';
const nav_bar = "../views/included-views/nav-bar.html";
let nav_bar_id = '#nav-bar';
const footer_view = "../views/included-views/footer.html";
let footer_id = '#index-footer';

obtenerHeaderFooter(header_view, header_id, true);
// obtenerHeaderFooter(nav_bar, nav_bar_id);
obtenerHeaderFooter(footer_view, footer_id, false);
```

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.

```
function obtenerHeaderFooter(view, id, isHeader) {
  // Obtiene la vista de la cabecera / pie de página
  fetch(view)
    .then( (response) => response.text())
    .then(data => {
      document.querySelector(id).innerHTML = data;

      // Comprueba si es una cabecera o un footer
      if (isHeader) {
        if (localStorage.getItem('img') == 'userImagen') document.querySelector('.image-user-hd').style.backgroundImage = `url('../imgs/icons/user-no-image.png')`;
        else {
          document.querySelector('.image-user-hd').style.backgroundImage = `url('${localStorage.getItem('img')}')`;
          document.querySelector('.image-user-hd').style.borderRadius = "50%";
        }
        document.querySelector('#nom-usu').textContent = `${localStorage.getItem('usuario')} - Cerrar sesión`;
      }
      cerrarSesion();
    });
}
```

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

```
function cerrarSesion() {
  if (header_view == '../views/included-views/headerlog.html') {
    document.querySelector('#nom-usu').addEventListener('click', (e) => {
      localStorage.removeItem('usuario');
      localStorage.removeItem('webToken');
      localStorage.removeItem('id');
      localStorage.removeItem('img');
    });
  }
}
```

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.

```
const response = await fetch('http://localhost/GreenRoads/api/rutas.php')
  .catch(error => console.error(error));
const data = await response.json();

// Recorro las rutas obtenidas y creo un item con cada una
data.forEach(item => {
  numRutas++;
  if (numRutas <= maxRutas) {
    let ruta = document.createElement('div');
    ruta.classList.add('w-ruta-img');
    ruta.innerHTML = `
      <div class="ruta-img"></div>
      <div class="ruta-texto">
        <div>${ item.nombre_ruta }</div>
        <div class="texto-ruta">Dificultad: <span class="span-dificultad">${ item.dificultad }</span></div>
        <div class="texto-ruta">Distancia: <span class="span-longitud">${ item.distancia } km</span></div>
        <div class="texto-ruta">Tiempo: <span class="span-tiempo">${ item.tiempo }</span></div>
      </div>`;
    rutas.append(ruta);
  }
});
```

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.

```
let contenido = document.querySelector('#wrapper');
let all_rutas = document.createElement('div');
let btn_ver1 = "";
let btn_ver2 = "";
let rutas = [];
window.addEventListener('resize', changeText);
```

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*.

```
// Cambia el texto de unos botones dependiendo del tamaño de la página
function changeText() {
    if(document.documentElement.clientWidth < 420) {
        btn_ver1.innerHTML = `<i class="bi bi-plus-lg"></i>`;
        btn_ver2.innerHTML = `<i class="bi bi-plus-lg"></i>`;
    } else {
        btn_ver1.textContent = `Ver más`;
        btn_ver2.textContent = `Ver más`;
    }
}
```

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

```
// Realizo una petición a rutas.php para obtener todas las rutas
const response = await fetch('http://localhost/GreenRoads/api/rutas.php')
    .catch(error => console.error(error));
const data = await response.json();

// Recorro las rutas obtenidas y creo un item con cada una
data.forEach(item => {
    crearRutas(item);
})
contenido.append(all_rutas);
```

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.

```

let url = `http://localhost/GreenRoads/api/rutas.php`;
if (!(iNombre.value == "" && iDificultad.value == "" && iUsuario.value == "")) {
  rutas = [];
  let aux = '?';
  if (iNombre.value != "") {
    let nombre = iNombre.value.replace(/ /g, '+');
    url = `${url }${ aux }nombre_ruta=${ nombre }`;
    if (aux == '?') {
      aux = '&';
    }
  }
  if (iDificultad.value != "") {
    url = `${url }${ aux }dificultad=${ iDificultad.value }`;
    if (aux == '?') {
      aux = '&';
    }
  }
  if (iUsuario.value != "") {
    url = `${url }${ aux }usuario=${ iUsuario.value }`;
    if (aux == '?') {
      aux = '&';
    }
  }
}

```

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.

```

verMas.addEventListener('click', (e) => {
  e.preventDefault();
  let id = e.target.dataset.id;
  console.log(id);
  window.location.href = `detalles.php?id=${ id }`;
});

```

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.


```
// Con ayuda de PHP obtenemos la latitud y longitud del punto de partida de la ruta
let start_lat = <?php echo $ruta[0]->start_lat; ?>;
let start_lon = <?php echo $ruta[0]->start_lon; ?>;

// Si no tiene punto de partida le ponemos un valor predeterminado
if (start_lat == "" || start_lat == null) start_lat = 42.60003;
if (start_lon == "" || start_lon == null) start_lon = -5.57032;

// L.map es la clase central de la API. Creamos y manipulamos el mapa
let map = L.map('map').setView([start_lat, start_lon], 15);

// Mapa base con máximo nivel de zoom
L.tileLayer('https://tile.thunderforest.com/landscape/{z}/{x}/{y}.png?apikey=b8b39a61c93e4e49ac1dab84527bedff', {
  maxZoom: 18,
  attribution: '&copy; <a href="https://www.thunderforest.com/terms/">Thunderforest</a>'
}).addTo(map);

// Dibujar el trazo de la ruta
L.polyline([<?php echo $ruta[0]->puntos ?> ], {
  color: '#87A330',
  weight: 5,
}).addTo(map);
```

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

```
// Control de escala
L.control.scale().addTo(map);

// Estilos personalizados del marcador con iconos de font-awesome
let marcador = L.AwesomeMarkers.icon({
  icon: 'leaf',
  prefix: 'fa',
  markerColor: 'green',
  iconColor: 'white',
});

// Colocar el marcador en las posiciones iniciales de la ruta
L.marker([start_lat, start_lon], {icon:marcador}).addTo(map);
```

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 mostrará un enlace para iniciar sesión.

```
if (localStorage.getItem('usuario') != null) {
  document.querySelector('#iniciar-sesion').style.display = 'none';
  document.querySelector('.username').textContent = `${ localStorage.getItem('usuario') }`;
} else {
  document.querySelector('#add-comment').style.display = 'none';
  document.querySelector('#iniciar-sesion').style.display = 'block';
}
```

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*.

```

if (localStorage.getItem('usuario') !== null) {
  document.querySelector('.com-enlace').addEventListener('click', (e) => {
    e.preventDefault();
    let texto = document.querySelector('#comentario-txt').value;
    if (texto !== "") {
      let comentario = {
        'usuario': localStorage.usuario,
        'comentario': texto,
        'id_ruta': <?php echo $id; ?>
      }
    }
  });
}

```

Continúa 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.

```

fetch('http://localhost/GreenRoads/api/comentario.php', {
  method: 'POST',
  headers: {
    'Content-type': 'application/json,charset=utf-8'
  },
  body: JSON.stringify(comentario)
})
.then((response) => {
  switch (response.status) {
    case 200:
      window.location.href = 'inicio.html';
      return response.json();
    case 400:
      console.log(response);
  }
});

```

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.

```

// Petición pasando el usuario y la contraseña
const url = (`http://localhost/GreenRoads/api/usuario.php?usuario=${ usuario }&pass=${ pass }`);
fetch( url )
  .then(response => {
    switch (response.status) {
      case 200:
        return response.json();
      case 404:
        document.querySelector('#alerta').style.display = 'block';
      case 409:
        document.querySelector('#alerta').style.display = 'block';
    }
  })

```

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**.

```
.then( data => {
  if (data[0]) {
    // Añade los datos a localStorage para mantener la sesión activa
    localStorage.setItem('webToken', data[0].webToken);
    localStorage.setItem('usuario', data[0].usuario);
    localStorage.setItem('id', data[0].id);

    // Comprueba si tiene una imagen
    if(data[0].imagen != null) localStorage.setItem('img', data[0].imagen);
    else localStorage.setItem('img', 'userImagen');

    document.querySelector('#login').style.display = 'none';
    document.querySelector('#logueado').style.display = 'block';
  } else {
    console.log("ERROR");
  }
})
```

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.

```
const patrones = {
  'email': /^(\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,10})+)/,
  'fecha': /^(([1-2][0-9]{3})\-(0[1-9]|1[0-2])\-(0[1-9]|1\d|2\d|3[0-1]))$/,
  'contrasena': /^(?=.*)(?=.*[A-Z])(?=.*\d)[A-Za-z\d]{8,}$/,
};

if (localStorage.getItem('webToken') == null) renderPage();
```

Comprueba si los campos están vacíos con el siguiente código:

```
// Comprueba los campos vacíos
let vacio = emptyFields();
let alerta = document.querySelector('#alerta');
let alerta2 = document.querySelector('#alerta2');
```

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 **vacio**, que tiene almacenado el nombre del campo vacío.

```
// Comprueba campos vacíos y les da estilos
function emptyFields() {
  let inputs = document.querySelectorAll('input');
  Array.from(inputs).forEach(item => {
    item.style.border = '1px solid #B9B9B9';
  })
  let vacio = 0;
  if (document.querySelector('#nombreApellido').value == "") {
    vacio = "Nombre y apellidos";
    changeInputStyle('#nombreApellido', 'red');
  }
}
```

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.

```
// Obtiene las actividades seleccionadas del formulario
function obtenerActividades() {
  let actividades = [];
  if (document.querySelector('#jogging').checked) actividades.push('jogging');
  if (document.querySelector('#trekking').checked) actividades.push('trekking');
  if (document.querySelector('#cycling').checked) actividades.push('cycling');
  if (document.querySelector('#mountaineering').checked) actividades.push('mountaineering');
  if (document.querySelector('#walk').checked) actividades.push('walk');
  if (document.querySelector('#kayak').checked) actividades.push('kayak');

  // Lo devuelve como una cadena separada por espacios
  return (actividades.join(' '));
}
```

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

```
if (!(comprobarCampos(correo, 'email'))) {
  writeAlert('Correo no válido');
  changeInputStyle('#correo', 'red');
} else if (!(comprobarCampos(contrasena, 'contrasena'))) {
  writeAlert('La contraseña debe contener al menos 8 caracteres, mayúsculas, minúsculas y números');
  changeInputStyle('#contrasena', 'red');
} else if (!(comprobarCampos(nacimiento, 'fecha'))) {
  writeAlert('Fecha no válida');
  changeInputStyle('#fechanac', 'red');
}
```

```
// Comprueba los campos con los patrones
function comprobarCampos(campo, patron) {
  return patrones[patron].test(campo);
}
```

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

```
// Genera un JSON con los datos obtenidos del usuario
function generateJSON(nombre, usuario, correo, contrasena, nacimiento, estatura, peso, actividades) {
    return user = {
        'nombre': nombre,
        'usuario': usuario,
        'correo': correo,
        'contrasena': contrasena,
        'nacimiento': nacimiento,
        'estatura': estatura,
        'peso': peso,
        'actividades': actividades,
    };
}
```

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.

```
// Petición para registrarse
fetch('http://localhost/GreenRoads/api/usuario.php', {
    method: 'POST',
    headers: {
        'Content-type': 'application/json,charset=utf-8'
    },
    body: JSON.stringify(user),
}).then(response => {
    switch(response.status) {
        case 200:
            return response.text();
        case 404:
            writeAlert('Ha ocurrido un error');
        case 409:
            writeAlert('El nombre de usuario ya está en uso');
            changeInputStyle('#usuario', 'red');
    }
})
.then(data => {
    document.querySelector('#registro').style.display = 'none';
    document.querySelector('#registrado').style.display = 'block';
})
```

Perfil de usuario

Crea un objeto para almacenar lejjel usuario obtenido mediante *LocalStorage* y la función **obtenerUsuario**.

```
let user = {};

renderPage();

function renderPage() {
  obtenerUsuario(localStorage.getItem('usuario'));
  document.querySelector('#eliminar').addEventListener('click', clickEliminar);
}
```

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.

```
let url = (`http://localhost/GreenRoads/api/usuario.php?usuario=${ usuario }`);
fetch( url )
  .then(response => {
    switch (response.status) {
      case 200:
        return response.json();
      case 404:
        console.log("ERROR");
    }
  })
  .then(data => {
    if (data[0]) {
      if (data[0] != null) {
        // Objeto con información del usuario y mostrar en la página
        user = {
          'id': data[0]['id'],
          'usuario': data[0]['usuario'],
        };
        document.querySelector('#nom-usuario').textContent = `${ data[0]['usuario'] }`;
        document.querySelector('#email').textContent = `${ data[0]['correo'] }`;
      }
    }
  })
```

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.

```
// Obtiene los valores del formulario
let input_ctr = document.querySelector('#contrasena');
let contrasena = input_ctr.value;
let usuario = {
  'id': user.id,
  'pass': contrasena,
}
```

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.

```
// Comprueba que la contraseña no esté en blanco
if (contrasena != "") {

    // Petición DELETE hacia editar.php
    fetch(`http://localhost/GreenRoads/api/editar.php`,{
        method: 'DELETE',
        headers: {
            'Content-type': 'application/json,charset=utf-8'
        },
        body: JSON.stringify(usuario),
    })
}
```

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.

```
.then(response => {
    switch (response.status) {
        case 200:
            // Todo correcto, se eliminan los items de localStorage
            localStorage.removeItem('usuario');
            localStorage.removeItem('webToken');
            localStorage.removeItem('id');
            localStorage.removeItem('img');
            document.querySelector('#info').style.display = 'none';
            document.querySelector('#delete-form').style.display = 'block';
            return response.text();
        case 400:
            console.log("No se ha podido borrar el usuario");
        case 404:
            writeAlert("La contraseña es incorrecta");
            input_ctr.style.border = '1px solid red';
    }
})
```

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

```
// Escribe una alerta con el texto pasado por parámetro
function writeAlert(texto) {
    document.querySelector('.alerta').textContent = texto;
    document.querySelector('.alerta').style.display = 'block';
}
```

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.

```
// Creación del JSON del usuario
let user = {
  'nombre': document.querySelector('#nombreApellido').value,
  'usuario': document.querySelector('#usuario').value,
  'correo': document.querySelector('#correo').value,
  'contrasena': document.querySelector('#contrasena').value,
  'nacimiento': document.querySelector('#fechanac').value,
  'estatura': document.querySelector('#estatura').value,
  'peso': document.querySelector('#peso').value,
  'actividades': actividadesArr,
};
```

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

```
// Fetch a la API editar.php
const url2 = (`http://localhost/GreenRoads/api/editar.php`);
fetch( url2, {
  method: 'POST',
  headers: {
    'Content-type': 'application/json;charset=utf-8'
  },
  body: JSON.stringify(user),
})
.then(response => {
  switch (response.status) {
    case 200:
      return response.json();
    case 404:
      return 404;
    case 409:
      return 409;
  }
})
```

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.

```
.then(data => {
  // Comprueba los resultados que ha devuelto
  if (data != 409 && data != 400){
    document.querySelector('#informacion').style.display = 'none';
    document.querySelector('#editado').style.display = 'block';
  } else {
    document.querySelector('#alerta2').textContent = 'Contraseña incorrecta'
    document.querySelector('#alerta2').style.display = 'block';
  }
})
```


Subir rutas

Para subir las rutas es necesario un fichero JavaScript adicional: GXSparger.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.

```
let botones = document.querySelectorAll('.btn-dif');
Array.from(botones).forEach(item => {
  item.addEventListener('click', () => {
    Array.from(botones).forEach(bot => {
      bot.classList.remove('selected');
    })
    item.classList.add('selected');
  })
});
```

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.

```
// Cargar el fichero
let fileInput = document.getElementById("gpx");
let file = fileInput.files[0];
let reader = new FileReader();

reader.onload = function() {
  let gpx = reader.result;
  let parser = new gpxParser();
  parser.parse(gpx);
  let json = parser.tracks[0];
```

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

```
// Array para los puntos de la ruta
let puntos = [];
json.points.forEach(item => {
  puntos.push([item.lat, item.lon]);
})
```

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.

```
let ruta = {
  'nombre_ruta': document.querySelector('#nombre').value,
  'distancia': ((json.distance.total)/1000).toFixed(2),
  'max_height': Math.round(json.elevation.max),
  'min_height': Math.round(json.elevation.min),
  'dificultad': document.querySelector('.selected').textContent,
  'pos_slope': Math.round(json.elevation.neg),
  'neg_slope': Math.round(json.elevation.pos),
  'start_lat': json.points[0].lat,
  'start_lon': json.points[0].lon,
  'usuario': localStorage.getItem('usuario'),
  'fecha': json.points[0].time,
  'descripcion': document.querySelector('#descripcion').value,
  'puntos': JSON.stringify(puntos),
};
```

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**.

```
fetch('http://localhost/GreenRoads/api/rutas.php', {
  method: 'POST',
  headers: {
    'Content-type': 'application/json;charset=utf-8'
  },
  body: JSON.stringify(ruta)
})
.then((response) => {
  switch (response.status) {
    case 201:
      document.querySelector('#exito').style.display = 'block';
      document.querySelector('#div-form').style.display = 'none';
      return response.json();
    case 400:
      console.log(response);
  }
});
```

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

```
reader.readAsText(file);
let gpx = new gpxParser();
gpx.parse("<?xml><gpx></gpx></xml>");
```

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.

```
async function renderPage() {
  const response = await fetch('http://localhost/GreenRoads/api/rutas.php?usuario=${ localStorage.getItem('usuario') }')
  .catch(error => console.error(error));
  const data = await response.json();
  data.forEach(item => {
    crearRutas(item);
  })
}
```

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 tabla “usuarios” que coincidan con el nombre de usuario.

```
if($_SERVER['REQUEST_METHOD'] == 'POST') {  
  
    // Obtener el fichero enviado en el cuerpo  
    $json = file_get_contents('php://input');  
  
    // Comprueba si se ha obtenido el fichero  
    if (isset($json)) {  
        try {  
            //Almacenar el json en usuario  
            $usuario = json_decode($json);  
  
            // Cifrado de contraseña  
            $passHash = hash("sha512", $usuario->contrasena);  
  
            // Selecciona todos los usuarios que coincidan con el nombre de usuario pasado  
            $sqlNombre = "SELECT * FROM usuarios WHERE usuario = '{$usuario->usuario}'";
```

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 *Conflict*, 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*.

```

// Ejecuta la consulta
$result = $con->query($sqlNombre);

// El usuario se inserta si NO se ha encontrado ningún usuario con ese nombre
if ($result->num_rows == 0) {

    // Creación de sentencia sql para insertar nuevo usuario
    $sql = "INSERT INTO usuarios (nom_ape, usuario, correo, pass, fecnac, estatura, peso, activ_fav, rol)
    VALUES ('{$usuario->nombre}', '{$usuario->usuario}', '{$usuario->correo}', '{$passHash}',
    '{$usuario->nacimiento}', '{$usuario->estatura}', '{$usuario->peso}', '{$usuario->actividades}', 'user')";

    $con->query($sql);
    header("HTTP/1.1 201 Created");
    echo json_encode($con->insert_id);
} else {
    header("HTTP/1.1 409 Conflict");
}
} catch (mysqli_sql_exception) {
    header("HTTP/1.1 400 Bad Request");
}
}
exit;

```

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

Comprobar si el login es correcto

[http://localhost/GreenRoads/api/usuario.php?usuario=\\$usuario&pass=\\$pass](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.

```

if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    try {
        $sql = "SELECT * FROM usuarios WHERE 1";

        // Comprobar si se ha enviado el id como parámetro
        if (isset($_GET['id'])) {
            $id = $_GET['id'];
            $sql .= " AND id = '$id'";
        }
        if (isset($_GET['usuario']) && !isset($_GET['pass'])) {
            $usuario = $_GET['usuario'];
            $sql .= " AND usuario = '$usuario'";
        } else if (isset($_GET['usuario']) && isset($_GET['pass'])) {
            $usuario = $_GET['usuario'];
            $contrasena = $_GET['pass'];
            $hashPass = hash("sha512", $contrasena);
            $sql .= " AND usuario = '$usuario' AND pass = '$hashPass'";
        }

        $result = $con->query($sql);
        if ($result->num_rows > 0) {
            $usuarios = $result->fetch_all(MYSQLI_ASSOC); // Obtiene lo usuarios
        }
    } catch (Exception $e) {
        // Manejo de errores
    }
}

```

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.

```
$key = 'example_key';
```

```

// Genera un payload con el nombre de usuario (para el webToken)
$payload = [
    'iss' => $usuarios[0]['usuario'],
];

$jwt = JWT::encode($payload, $key, 'HS256');

$usuarios[0]['webToken'] = $jwt;

HEADER("HTTP/1.1 200 OK");
echo json_encode($usuarios);

```

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.

```
if($_SERVER['REQUEST_METHOD'] == 'POST') {

    // Obtener el fichero enviado en el cuerpo
    $json = file_get_contents('php://input');

    // Comprueba si se ha obtenido el fichero
    if (isset($json)) {
        try {
            // Almacenar el json en usuario
            $usuario = json_decode($json);

            // Cifrado de contraseña
            $passHash = hash("sha512", $usuario->contrasena);

            // Comprueba si el usuario existe con la contraseña pasada
            $comprobacion = "SELECT * FROM usuarios WHERE usuario = '{$usuario->usuario}' AND pass = '$passHash'";
            $result = $con->query($comprobacion);

            // Se ejecuta si obtiene un usuario
            if ($result->num_rows == 1){
```

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.

```
// Sentencia para actualizar los campos editados del usuario
$sql = "UPDATE usuarios SET nom_ape = '{$usuario->nombre}', usuario = '{$usuario->usuario}', correo = '{$usuario->correo}',
    fecnac = '{$usuario->nacimiento}', estatura = '{$usuario->estatura}', peso = '{$usuario->peso}',
    activ_fav = '{$usuario->actividades}' WHERE usuario = '{$usuario->usuario}' AND pass = '$passHash'";

// Ejecución de la sentencia
$con->query($sql);

// Devuelve 201 si se ha creado exitosamente
header("HTTP/1.1 201 Created");
echo json_encode($usuario);
```

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.

```
// Almacenar el json en usuario
$usuario = json_decode($json);

// Cifrado de contraseña
$hashPass = hash("sha512", $usuario->pass);

// Sentencia para eliminar el usuario que coincida en contraseña e id
$sql = "DELETE FROM usuarios WHERE id = '$usuario->id' AND pass = '$hashPass'";
try {
    $con->query($sql);
    header("HTTP/1.1 200 OK");
} catch(mysqli_sql_exception $e) {
    header("HTTP/1.1 404 Not Found");
}
```

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:

```
let ruta = {
  'nombre_ruta': document.querySelector('#nombre').value,
  'distancia': Math.round(json.distance.total),
  'max_height': Math.round(json.elevation.max),
  'min_height': Math.round(json.elevation.min),
  'dificultad': document.querySelector('.selected').textContent,
  'pos_slope': Math.round(json.elevation.neg),
  'neg_slope': Math.round(json.elevation.pos),
  'start_lat': json.points[0].lat,
  'start_lon': json.points[0].lon,
  'usuario': localStorage.getItem('usuario'),
  'fecha': json.points[0].time,
  'descripcion': document.querySelector('#descripcion').value,
  'puntos': JSON.stringify(puntos),
};
```

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**.

```
try {
    $ruta = json_decode($json);

    // Sentencia para insertar la ruta en la tabla ruta
    $sql = "INSERT INTO rutas (nombre_ruta, distancia, max_height, min_height, dificultad, pos_slope, neg_slope, start_lat, start_lon, usuario, fecha, descripcion, puntos)
    VALUES ('{$ruta->nombre_ruta}', '{$ruta->distancia}', '{$ruta->max_height}', '{$ruta->min_height}', '{$ruta->dificultad}', '{$ruta->pos_slope}', '{$ruta->neg_slope}',
    '{$ruta->start_lat}', '{$ruta->start_lon}', '{$ruta->usuario}', '{$ruta->fecha}', '{$ruta->descripcion}', '{$ruta->puntos}')";

    $con->query($sql);
    header("HTTP/1.1 201 OK");
    echo json_encode($ruta);
} catch (mysqli_sql_exception $e) {
    header("HTTP/1.1 400 Bad Request");
}
```

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.


```
// Selecciona todas las rutas
$sql = "SELECT * FROM rutas WHERE 1";

// Comprueba si se le ha pasado el id por parámetro y lo añade a la consulta
// Sirve para los detalles de rutas, donde se le pasa una id para obtenerlos
if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $sql .= " AND id = '$id'";
}

// Comprueba si se ha pasado un usuario por parámetro y lo añade a la consulta
// Sirve para obtener las rutas de un determinado usuario
if (isset($_GET['usuario'])) {
    $usuario = $_GET['usuario'];
    $sql .= " AND usuario LIKE '%$usuario%'";
}
if (isset($_GET['nombre_ruta'])) {
    $nombre_ruta = $_GET['nombre_ruta'];
    $sql .= " AND nombre_ruta LIKE '%$nombre_ruta%'";
}
if (isset($_GET['dificultad'])) {
    $dificultad = $_GET['dificultad'];
    $sql .= " AND dificultad = '$dificultad'";
}

$result = $con->query($sql);

// Obtener las rutas
$rutas = $result->fetch_all(MYSQLI_ASSOC);
header("HTTP/1.1 200 OK");
```

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.

```
// Obtener comentarios
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    try {
        // Selecciona todos los comentarios
        $sql = "SELECT * FROM comentarios WHERE 1";

        if (isset($_GET['id'])) {
            $id = urlencode($_GET['id']);
            $sql .= " AND id = '$id'";
        }

        if (isset($_GET['usuario'])) {
            $usuario = $_GET['usuario'];
            $sql .= " AND usuario = '$usuario'";
        }

        if (isset($_GET['id_ruta'])) {
            $id_ruta = $_GET['id_ruta'];
            $sql .= " AND id_ruta = '$id_ruta'";
        }

        $result = $con->query($sql);

        // Obtener los comentarios
        $comentarios = $result->fetch_all(MYSQLI_ASSOC);
        header("HTTP/1.1 200 OK");
        echo json_encode($comentarios);
    } catch (mysqli_sql_exception $e) {
        header("HTTP/1.1 404 Not Found");
    }
}
```

POST

El método post recibe un JSON de la siguiente estructura:

```
let comentario = {
    'usuario': localStorage.usuario,
    'comentario': texto,
    'id_ruta': <?php echo $id; ?>
}
```

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.

```
// Subir comentario
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Obtiene toda la información del comentario pasado en la cabecera de la petición
    $json = file_get_contents('php://input');

    // Comprueba si se ha obtenido un fichero
    if (isset($json)) {
        try {
            $comentario = json_decode($json);

            // Sentencia para insertar la ruta en la tabla ruta
            $sql = "INSERT INTO comentarios (usuario, comentario, id_ruta)
                VALUES ('{$comentario->usuario}', '{$comentario->comentario}', '{$comentario->id_ruta}')";

            $con->query($sql);
            header("HTTP/1.1 200 OK");
            echo json_encode($comentario);
        } catch(mysqli_sql_exception $e) {
            header("HTTP/1.1 400 Bad Request");
        }
    }
    exit;
}
```

Bibliografía

- [Página de imágenes](#)
- [Documentación de Leaflet](#)
- [Iconos de Bootstrap](#)