

Planit



Proyecto realizado por Raúl Martín

IES Galileo 2ºDAW 2024-2025

Memoria del proyecto

- **Introducción**
- **Análisis**
 - *Requisitos Funcionales*
 - *Requisitos No Funcionales*
 - *Tipos de Usuarios*
- **Diseño**
 - *Arquitectura del Sistema*
 - *Tecnologías Utilizadas*
- **Implementación**
 - *Estructura del código*
- **Pruebas**
 - *Pruebas Manuales*
- **Conclusiones**
 - *Dificultades encontradas*
 - *Objetivos y Logros*
 - *Posibles implementaciones Futuras*

Manual de Instalación

- **Requisitos previos**
- **Descargar e instalar el proyecto**
- **Preparar php y MySQL**
- **Inicializar base de datos**
- **Iniciar Planit**

Guía de Usuario

- **Registro**
- **Inicio de sesión**
- **Crear un nuevo Plan**
- **Buscar y Filtrar Eventos**
- **Participar en un Evento**
- **Gestionar tus Eventos**
 - *Ver tus eventos creados*
 - *Ver los eventos a los que estás apuntado*
- **Interactuar con comentarios**
- **Perfil de Usuario**
- **Bloquear usuarios y administrar usuarios bloqueados**
- **Interacción y elogio entre usuarios**
- **Funciones de administrador**
- **Cerrar Sesión**



Introducción

Planit nace como respuesta a la limitación de la gente común para organizar eventos de una manera formal y profesional, la cual, hasta el momento ha estado reservada para promotoras y agencias. Hasta ahora, la gente común se ha visto obligada a la difusión por redes sociales para publicitar y dar a conocer los eventos que quiera organizar personalmente. Al no hacerlo desde un servicio especializado en este campo, la difusión no tiene el mismo alcance, y las posibilidades de organización y gestión son mucho más limitadas. Planit viene a solucionar este problema ofreciendo una plataforma pública en la que los usuarios puedan publicar y gestionar eventos organizados por ellos mismos y asistir a ellos ofreciendo un feedback a los creadores e incluso generar ingresos. En definitiva, Planit es una nueva plataforma centrada en ofrecer eficacia, accesibilidad, difusión y autogestión a los usuarios que deseen organizar eventos formales.

Análisis

Requisitos Funcionales

- Registro y Login de usuarios
- Creación, edición, borrado y cancelado de eventos
- Búsqueda y filtros de eventos por su categoría, precio, ciudad y título
- Participación en eventos y capacidad de que el usuario gestione sus inscripciones
- Sistema de comentarios y puntos de experiencia para que los usuarios den feedback a los creadores del evento
- Capacidad de bloquear usuarios y gestionarlos
- Panel de administrador para administrar y moderar eventos, categorías y usuarios

Requisitos No Funcionales

- La plataforma debe ser accesible desde cualquier dispositivo con un navegador web y conexión a internet
- La interfaz debe ser intuitiva, responsive con un diseño moderno y atractivo
- La plataforma debe garantizar la integridad de los datos de los usuarios y su seguridad

Tipos de usuarios

- Usuarios:
 - Crear eventos
 - Crear categorías
 - Editar o cancelar sus eventos
 - Apuntarse a eventos de otros usuarios
 - Darse de baja de los eventos en los que están apuntados
 - Acceso al registro de eventos que han creado y a los que se han apuntado
 - Comentarios en los eventos a los que han asistido
 - Capacidad de bloquear a otros usuarios que no podrán asistir a los eventos del usuario que les ha bloqueado
 - Modificación de los datos de su cuenta de usuario
 - Visualizar el perfil público de otros usuarios y ver todos sus eventos
 - Elogiar a otros usuarios dándoles Planit Points
 - Eliminar su cuenta
- Administrador:
 - Todas las funciones de usuario
 - Editar, cancelar y modificar todos los eventos de la aplicación
 - Modificar los datos de cualquier usuario de la aplicación
 - Eliminar a usuarios de un evento
 - Crear y eliminar categorías

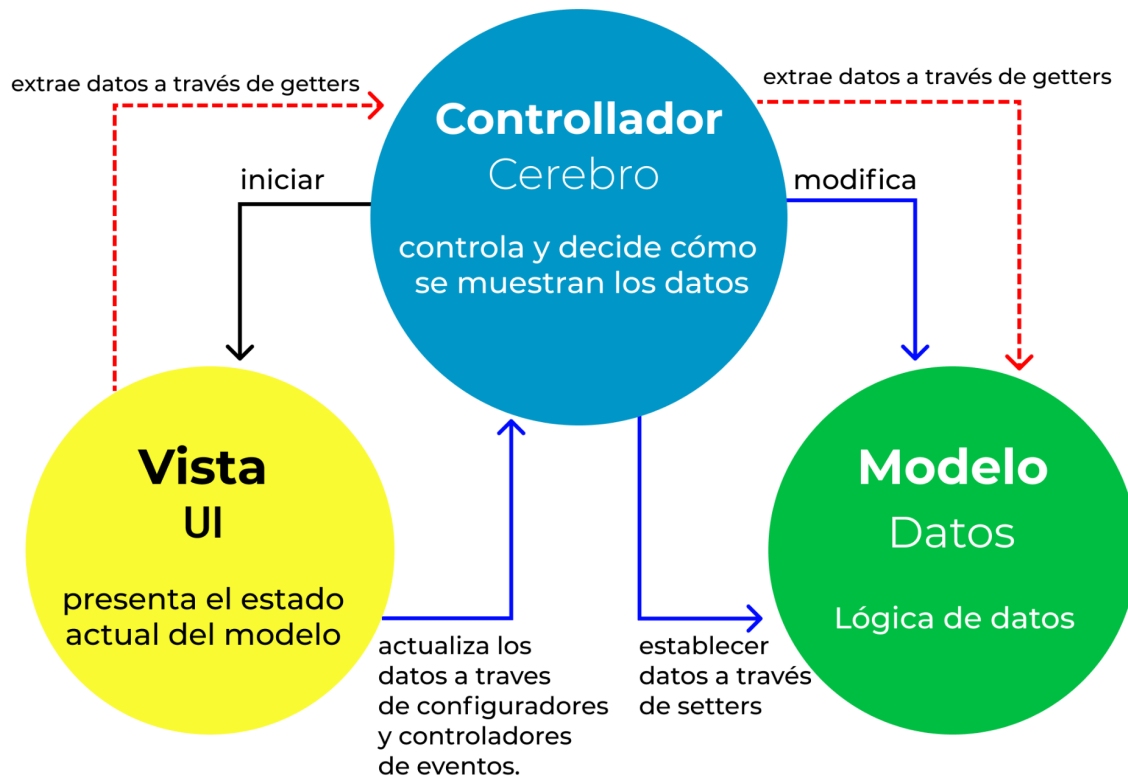
Diseño

Arquitectura del Sistema

La arquitectura elegida sigue el patrón Modelo-Vista-Controlador (MVC), implementado en el framework CodeIgniter para el Backend, HTML 5, JavaScript nativo, JQuery y React para el frontend y MySQL como base de datos.

El patrón Modelo Vista Controlador separa la lógica de la aplicación en tres componentes principales conectados entre sí, favoreciendo la modularización de las estructuras y facilitando la escalabilidad.

Patrones de Arquitectura MVC



fuelle: freeCodeCamp

Modelo

El Modelo representa el almacenamiento de los datos en la aplicación. Interactúa directamente con la base de datos para añadir, modificar y borrar los registros de la base de datos (CRUD). En CodeIgniter 4, los modelos extienden de la clase `CodeIgniter\Model`, la cual engloba las funciones que implementa el framework para agilizar el CRUD con la base de datos. Cada tabla de la base de datos será un modelo en el sistema. Por ejemplo, `EventModel` maneja las operaciones relacionadas con los eventos en la aplicación (relacionado con la tabla `events` en base de datos)

Vista

La vista es la capa visual que muestra al usuario los datos que necesita a través de una interfaz responsive. No contiene ninguna lógica de negocio. Solo renderiza la información que se acumula en el controlador correspondiente.

En este proyecto, las vistas son archivos `php` que generan un `html` como por ejemplo en `catalog.php` o `create.php`. Dichas vistas obtienen datos que se le envían desde el controlador para renderizar el contenido de forma dinámica.

Esta modularización facilita el desarrollo al poder realizar modificaciones sin miedo en la vista sin tener que preocuparnos por la integridad de la lógica de negocio de la aplicación

Controlador

El controlador actúa como intermediario entre el modelo y la vista. Recibe las solicitudes del usuario (como crear un nuevo evento), recupera los datos necesarios del modelo, los procesa según la lógica de negocio y los envía a la vista correspondiente para que se rendericen

Siguiendo el ejemplo de crear evento, desde la vista se llama a la función `store()` de `EventController`, se procesa la información de la vista, se envía la información al modelo (CRUD) y se vuelve a la vista mostrando un mensaje de éxito.

El controlador centraliza toda la lógica de negocio de cada modelo, lo que ayuda a la organización del código

Beneficios

Como resumen, el patrón Modelo Vista Controlador con el framework `CodeIgniter` nos ofrece

- Modularidad
- Reusabilidad
- Escalabilidad
- Legibilidad

Tecnologías Utilizadas

PHP 8 con CodeIgniter 4.6.0

Propósito: PHP es el lenguaje utilizado en el backend de la aplicación utilizado para manejar la lógica de negocio, es decir, gestión de sesiones, procesamiento de formularios, conexión a base de datos, cookies... `CodeIgniter` es el framework de PHP elegido para el desarrollo de la aplicación ya que es ideal para un proyecto de estas características, debido a su una curva de aprendizaje suave y una estructura que no difiere demasiado del php nativo. Además, es ligero, rápido y está diseñado para seguir el patrón MVC trabajado durante el curso. Además, entre sus ventajas técnicas está la gran variedad de métodos integrados que permiten optimizar y agilizar considerablemente el desarrollo.

HTML, CSS y JavaScript

Se ha utilizado HTML, CSS y JavaScript ya que son tecnologías estandarizadas en el desarrollo web y compatibles con todos los navegadores web modernos. HTML se utiliza para el renderizado de los datos en la página, CSS para aplicar estilos personalizados en la interfaz y añadir elementos responsivos y JavaScript para agregar interactividad en el navegador ya que permite mayor dinamismo y procesamiento de datos en tiempo real sin necesidad de recargar la página o cargar el servidor. En este proyecto, por ejemplo, se usa para la validación de formularios o filtro de los eventos en el catálogo principal

React

Se ha seleccionado React, una biblioteca de JavaScript que facilita el desarrollo de componentes JavaScript dinámicos e interactivos, por su capacidad y facilidad de crear interfaces activas y dinámicas a través de los estados. En este proyecto se utiliza para el menú desplegable que se muestra al hacer click sobre la foto de perfil del usuario en la parte superior derecha de la pantalla, y en la muestra dinámica de la foto de perfil o la carátula del evento cuando se añade una nueva o se modifica

Bootstrap

Con el propósito de acelerar el desarrollo, se ha elegido incorporar Bootstrap, un framework de CSS que ofrece estilos predefinidos y componentes css que permiten integrar elementos y estilos complejos (botones, cuadros, formularios, tarjetas, modales..) de manera sencilla. Así mismo, permite también que los estilos de la aplicación sean compatibles con dispositivos móviles sin realizar excesivas modificaciones.

JQuery

Con el propósito también de acelerar el desarrollo, se ha utilizado JQuery, es una librería de JavaScript que tiene una gran comunidad e incluye una gran cantidad de plugins públicos que permiten acelerar el desarrollo y que está centrado en la manipulación del DOM en tiempo real sin recarga de la página. JQuery simplifica el Javascript nativo sin modificar demasiado su sintaxis. Se especializa en el manejo de eventos y manipulación del DOM, por lo que junto a su plugin SELECT2, es lo ideal para los filtros de eventos. Es en virtud de todo ello que en este proyecto, se utiliza, por ejemplo, en numerosos filtros (categoría, precio, ciudad...).

MySQL

Para el sistema de gestión de base de datos se ha elegido MySQL, el sistema más extendido en desarrollo web, y que se encargará de almacenar los datos de los usuarios en la aplicación. Se ha escogido MySQL puesto que es muy fiable, fácilmente escalable y, sobre todo, CodeIgniter está diseñado para esta tecnología.

OpenStreetMaps:

OpenStreetMap permite seleccionar la ubicación de los eventos en un mapa interactivo, así como visualizar donde se llevará a cabo un evento y de esta manera elegir y visualizar fácilmente la ubicación del evento ya sea arrastrando la chincheta en el mapa o introduciendo manualmente la dirección. Se ha elegido esta tecnología por delante de otras ya que es la mejor alternativa pública y de código abierto que incluye las funcionalidades que necesita el proyecto.

Herramientas de Desarrollo

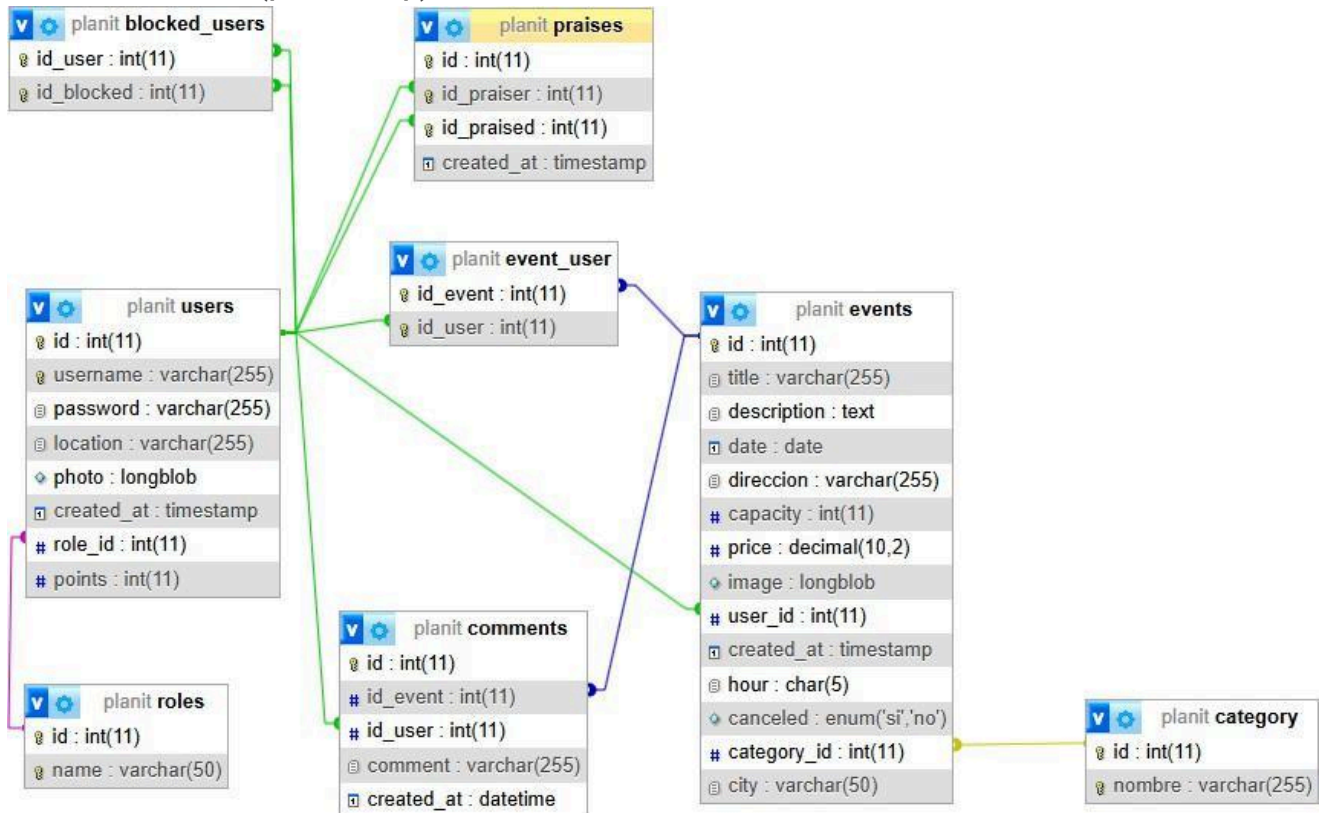
- Node.js y npm: Utilizados para desarrollar en react, crear el proyecto y compilarlo a JavaScript nativo
- Composer: Gestiona las dependencias de PHP. Se utilizó para inicializar e instalar el proyecto de CodeIgniter por primera vez
- XAMPP: Proporciona un entorno de desarrollo web local. Utiliza Apache como servidor web y MySQL como base de datos.
- Spark: Componente integrado en CodeIgniter para inicializar el proyecto y probarlo en un servidor local

Implementación

Estructura del código

El proyecto se estructura en los siguientes archivos utilizando la estructura predeterminada de CodeIgniter:

Base de Datos (planit.sql)



users:

- id (identificador único de cada usuario)
- username (nombre de usuario único de cada usuario)
- password (se guarda la contraseña que es cifrada previamente en el servidor)
- location (ubicación del usuario que elige al crear su cuenta)
- photo (foto de perfil del usuario guardada en formato BLOB)
- role_id (clave foránea de la tabla rol. Identifica si el usuario es usuario normal o administrador)
- points (Planit Points)

events:

- id (identificador único de cada evento)
- title (título del evento mostrado en el catálogo)
- description (descripción detallada mostrada en la vista del evento)
- date (fecha en la que tendrá lugar el evento)
- dirección (ubicación completa donde se hará el evento)
- capacity (campo opcional si tiene un aforo determinado)
- image (carátula del evento)
- hour (hora a la que tendrá lugar el evento)
- canceled (enum que determina si el evento ha sido cancelado o no)
- category_id (clave foránea de la tabla category que identifica la categoría del evento)
- city (ciudad donde tendrá lugar el event)

category:

- id (identificador único de cada categoría)
- nombre (nombre único de cada categoría)

comments:

- id (identificador único de cada comentario)
- id_evento (clave foránea que relaciona el comentario al evento donde debe ir asociado)
- id_user (clave foránea que relaciona el comentario al usuario que lo ha publicado)
- comment(Texto del comentario)

blocked_users:

- id_user (id del usuario que bloquea)
- id_blocked (id del usuario bloqueado)

event_user:

- Tabla que relaciona eventos y usuarios. Cada fila de esta tabla es un usuario apuntado a un evento
- id_event (id del evento al que el usuario se apunta)
- id_user (id del usuario que se apunta al evento)

praises:

- Tabla que gestiona los elogios que los usuarios dan a otros para que un usuario no pueda elogiar más de una vez al mismo usuario
- id (identificador único de cada elogio)
- id_praiser (id del usuario que elogia)
- id_praised (id del usuario elogiado)

roles:

- id (identificador único de cada rol)
- name(nombre de cada rol)

Modelos

- CategoryModel
- CommentModel
- EventModel
- EventUserModel
- PraiseModel
- UserModel
- (Representan a sus respectivas tablas en base de datos)

Vistas

- Admin
 - admin_catalog: Vista de todos los eventos de la aplicación con búsqueda por título y acceso a verlos, editarlos, cancelarlos y rehabilitarlos si están cancelados
 - categories: Vista de todas las categorías de la aplicación con búsqueda por título y acceso a borrarlos
 - createCategory: Vista que permite crear una nueva categoría
 - userAttendedEvents: vista de todos los eventos donde está apuntado un usuario con capacidad de desapuntarlo de eventos
 - userPoints: vista de los puntos que tiene un usuario con capacidad de modificarlos
 - viewUserEvents: Vista de todos los eventos creados por el usuario separados por activos y pasados con capacidad de editarlos, cancelarlos, modificarlos y rehabilitarlos
 - viewUsers: vista del listado de usuarios de la aplicación con buscador de nombre de usuario con capacidad de eliminarlo, y editar sus datos
- auth:
 - login: Formulario de inicio de sesión
 - register: Formulario de registro
- event:
 - catálogo: Index del proyecto. Vista de los eventos con sus respectivos filtros de título, ciudad, categoría, rango de precio, con capacidad de crear un nuevo plan y acceder a los distintos planes del catálogo
 - Comments: vista de los comentarios que tiene un evento que ya ha sucedido
 - create.php: vista del formulario para crear un nuevo evento
 - edit: vista en la que cargan los datos del evento en el formulario desde base de datos desde la que se pueden editar y sobrescribir los valores. Accesible sólo para los creadores del evento o los administradores
 - events: Vista de los planes donde está apuntado o ha asistido el usuario
 - my_events: Vista de todos los planes creados por el usuario separados en activos y pasados. Con capacidad de editar los activos o cancelarlos y ver los comentarios de los pasados
 - show: Vista detallada del evento incluyendo descripción. Si se ha iniciado sesión y el aforo no está lleno, el usuario podrá apuntarse. Si el evento es de pago, se le abrirá un modal donde se simulará una pasarela de pago que deben rellenar correctamente para apuntarse

- user
 - attendees: vista desde la que el creador puede ver los asistentes a su evento y podrá eliminar a quien desee
 - blockedUsers: vista que muestra los usuarios bloqueados por el usuario. Los usuarios bloqueados no podrán acceder a los eventos creados por el usuario que lo bloquea. Incluye un buscador por nombre de usuario para bloquear a usuarios
 - profile: vista con un formulario donde el usuario puede modificar la información de su perfil
 - user: vista pública de un usuario donde se ven todos sus eventos creados y sus puntos. Si no has elogiado al usuario, podrás hacerlo una vez
- Includes
 - confirm: Include que contiene una plantilla de modal que se usa como confirm para distintas operaciones de la aplicación. También incluye su respectivo script para su funcionamiento en cada caso

Controladores (funciones documentadas en el código)

- AdminController: clase que contiene las funciones utilizadas por el administrador, modificando los modelos eventUser, user, event y category
- AuthController: clase que contiene las funciones de autenticación en la aplicación. Como inicio de sesión, registro y cierre de sesión
- EventController: clase que contiene las funciones relacionadas con los eventos (excluyendo las asistencias y altas a otra clase a parte). Hace modificaciones en los modelos event, user, eventUser y category
- EventUserController: clase que contiene las funciones relacionadas con las asistencias de los usuarios a los eventos. Hace modificaciones en los modelos eventUser, event, comment y user
- MapController: API que maneja el uso de OpenStreetMaps en la aplicación. a través de fechas en javascript, modifica la dirección en el input de dirección del formulario, o modifica la ubicación en el mapa según lo escrito en el input
- UserController: clase que contiene todas las funciones relacionadas con la gestión de usuarios, hace cambios en los modelos praise, user y event
- routes: Rutas que asocian cada enlace en la aplicación a una vista o una función

CSS

- attendees: estilos adicionales para la página attendees
- auth: encuadrar y añadir márgenes y bordes al formulario de login y registro
- catalog: estilos para encuadrar correctamente la foto del evento y otros elementos y una transición para que su tamaño aumente cuando se pose el ratón sobre ello
- create: estilos centrados en dar apariencia al iframe donde se ubica el mapa y simular una carga cuando el usuario escribe una dirección
- edit: estilos para el mapa y el encuadre del formulario
- events: estilos para encuadrar las imágenes, tarjetas y añadir un overlay con una imagen a los eventos cancelados
- profile: estilos adicionales para el diseño responsive del recuadro donde se ubica la foto de perfil del usuario
- show: Estilos adicionales para la vista detallada del evento
- user: estilos adicionales para el perfil público, incluyendo un modal con la foto de perfil ampliada, encuadres, y ajuste de la foto de perfil

JS

- jquery: conjunto de funciones jquery que manejan los filtros de eventos en el catálogo
- map: Maneja Openstreetmaps y a través de la comunicación con la api, se inicializa el mapa, se cambia la dirección y la ubicación en el mapa, también muestra y oculta errores en la dirección y el overlay que simula la carga
- profile: carga de forma provisional una nueva foto de perfil en el input de foto de perfil antes de cargar en el servidor en forma de preview
- eventPhotoPreview: script que permite la carga dinámica de la carátula del evento en edit.php
- validarEvento: validación del formulario al crear un nuevo evento o al editar un evento existente
- confirmations: script necesario para el funcionamiento del confirm personalizado

React

- ProfileDropdown: Menú desplegable que se activa cuando el usuario hace click en su foto de perfil arriba a la derecha de la pantalla
- ProfilePhotoUploader: componente que gestiona la actualización de foto de perfil y su preview en el formulario de registro y modificación de perfil, asegurando también que los estilos se apliquen correctamente
- main: fichero main que monta ambos componentes en uno para compilarse a javascript nativ

Pruebas

Pruebas Manuales

Se han verificado manualmente todas las funcionalidades de la aplicación en todos los escenarios posibles y se ha cargado a la base de datos con cientos de usuarios y eventos para poner a prueba la robustez de la aplicación. No he encontrado ningún punto donde puedan extraerse datos de los usuarios y realizar inserciones de código sql.

Conclusiones

Dificultades encontradas

- La idea inicial fue implementar Google Maps, pero tras investigar, las apis de google están más restringidas y tienen un número de usos gratuitos limitado. En su lugar, se ha utilizado OpenStreetMap, la cual realiza la misma funcion
- He tenido que implementar React “con calzador”. Apenas se ha utilizado y hubiera preferido utilizar JQuery para hacer los componentes. Pese a ello, veo el potencial que tiene y aunque veo una curva de aprendizaje más dura que con CodeIgniter, me parece positivo haber hecho una primera toma de contacto con React
- No ha sido posible integrar google pay o paypal por las restricciones que tienen para su uso a nivel doméstico. En su lugar he creado yo mismo un modal que simula el pago con tarjeta que valida que los datos introducidos puedan pertenecer a una tarjeta real

Objetivos y Logros

- Gestionar correctamente la creación de usuario, inicios de sesión y contraseñas de forma segura ✓
- Hacer un catálogo funcional que pueda ser filtrado correctamente y haga un buen uso de SQL ✓
- Gestionar correctamente las reservas, teniendo en cuenta aforos ✓
- Realizar una pasarela de pago funcional y creíble ✓
- La página debe tener diseño responsive, soporte para pantallas de móvil y un diseño profesional ✓
- Gestión correcta de la configuración del usuario ✓
- Página segura que cuente con todas las medidas de seguridad aprendidas durante el curso ✓

Posibles Implementaciones Futuras

Se podrían agregar funcionalidades adicionales como notificaciones en tiempo real, un sistema de chat entre usuarios, restablecer contraseña a través de correo electrónico, generación de las entradas a los eventos a través de código QR, etc.

Manual de instalación

Requisitos previos:

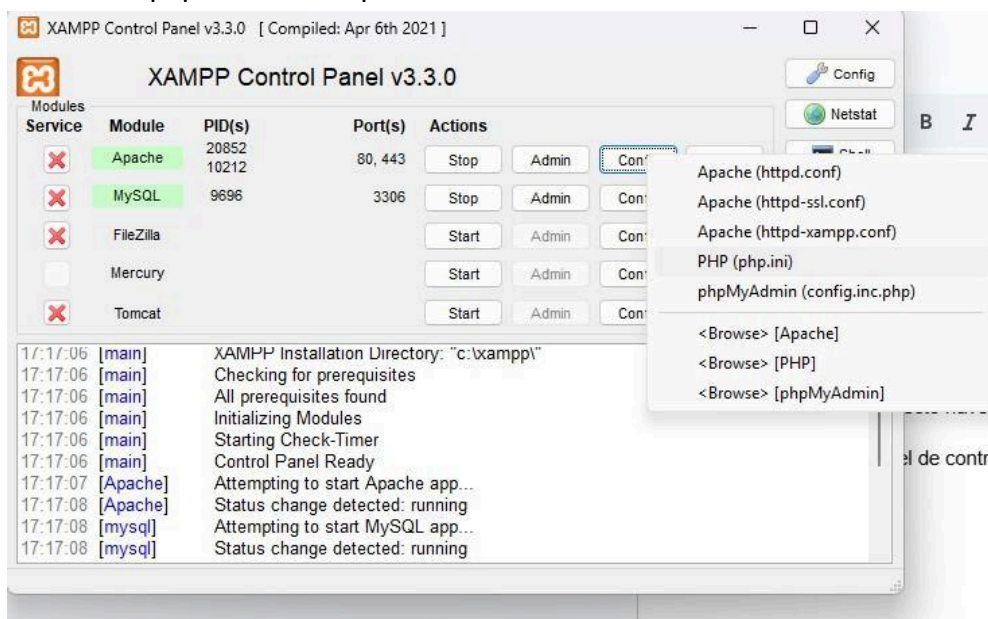
- XAMPP (o un servidor local equivalente como WAMP). Descarga XAMPP [aquí](#).
- PHP (versión 8.1 o superior) Viene incluido con XAMPP pero debemos asegurarnos de esté correctamente configurado
- Composer (gestor de dependencias de php). Puedes descargarlo [aquí](#) (asegúrate de que se instala sobre el PHP de XAMPP)
- Navegador Web
- Extractor de archivos (recomendado: 7zip).Puedes descargarlo [aquí](#)

Descargar e instalar el proyecto.

Extrae con 7zip o winrar y planit.zip y obtendrás la carpeta Planit.

Preparar php y MySQL

Abriremos php.ini desde el panel de control de XAMPP:



Dentro del fichero navegaremos hasta el apartado de extensiones. Asegúrate de que las siguientes extensiones están activadas (sin ; al principio)

```
; otherwise it results in segfault when unloading after using SASL.
; See https://github.com/php/php-src/issues/8620 for more info.
;extension=ldap

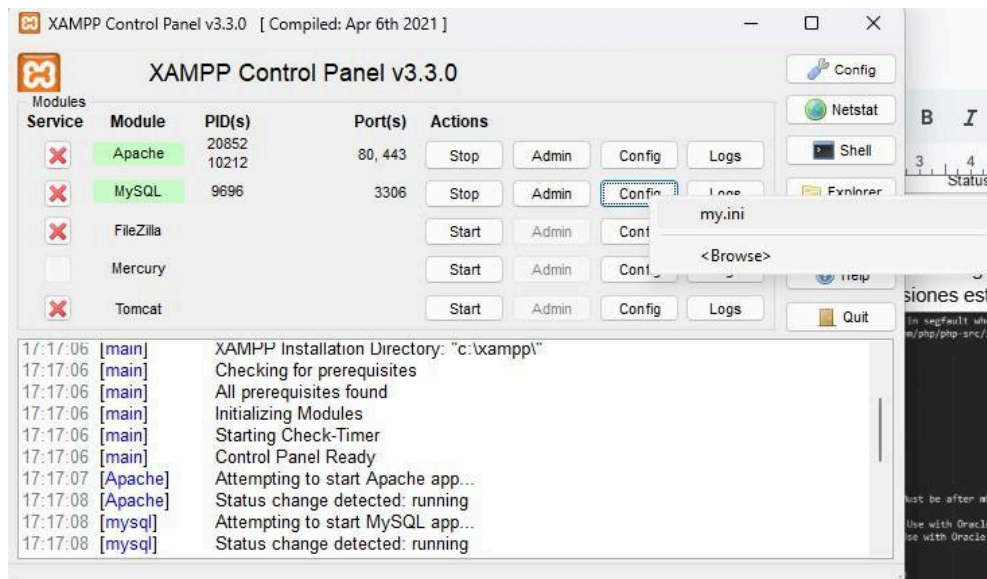
extension=curl
extension=ffi
;extension=ftp
extension=fileinfo
extension=gd
extension=gettext
extension=gmp
extension=intl
extension=imap
extension=mbstring
extension=exif      ; Must be after mbstring as it depends on it
extension=mysqli
;extension=oci8_12c  ; Use with Oracle Database 12c Instant Client
;extension=oci8_19  ; Use with Oracle Database 19 Instant Client
extension=odbc
;extension=openssl
;extension=pdo_firebird
extension=pdo_mysql
;extension=pdo_oci
extension=pdo_odbc
extension=pdo_pgsql
extension=pdo_sqlite
extension=pgsql
extension=shmop

; The MIBS data available in the PHP distribution must be installed.
; See https://www.php.net/manual/en/snmp.installation.php
;extension=snmp

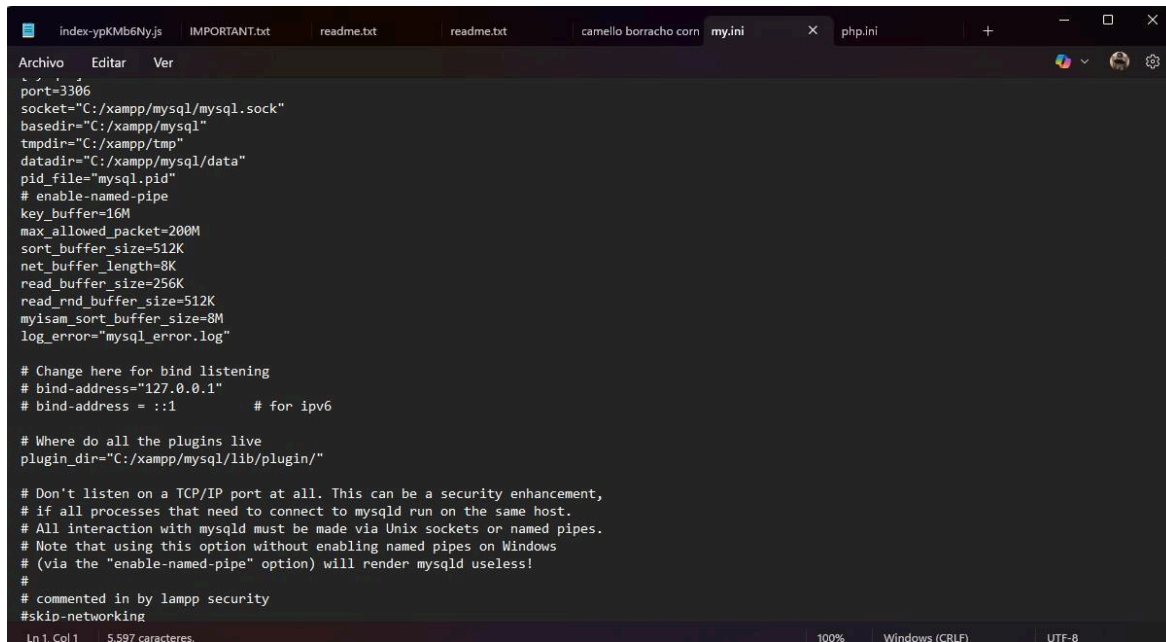
extension=soap
extension=sockets
extension=sodium
extension=sqlite3
extension=tidy
extension=xsl
extension=zip
```

Guardamos los cambios y cerramos.

Turno de mysql. Abriremos my.ini desde el panel de control de xampp



Asegúrate de que el valor de Max Allowed Packet sea de mínimo 200M



```
Archivo  Editar  Ver
index-ypkMb6Nyjs  IMPORTANT.txt  readme.txt  readme.txt  camello borracho corn  my.ini  x  php.ini  +  -  □  x
port=3306
socket="C:/xampp/mysql/mysql.sock"
basedir="C:/xampp/mysql"
tmpdir="C:/xampp/tmp"
datadir="C:/xampp/mysql/data"
pid_file="mysql.pid"
# enable-named-pipe
key_buffer=16M
max_allowed_packet=200M
sort_buffer_size=512K
net_buffer_length=8K
read_buffer_size=256K
read_rnd_buffer_size=512K
myisam_sort_buffer_size=8M
log_error="mysql_error.log"

# Change here for bind listening
# bind-address="127.0.0.1"
# bind-address = ::1          # for ipv6

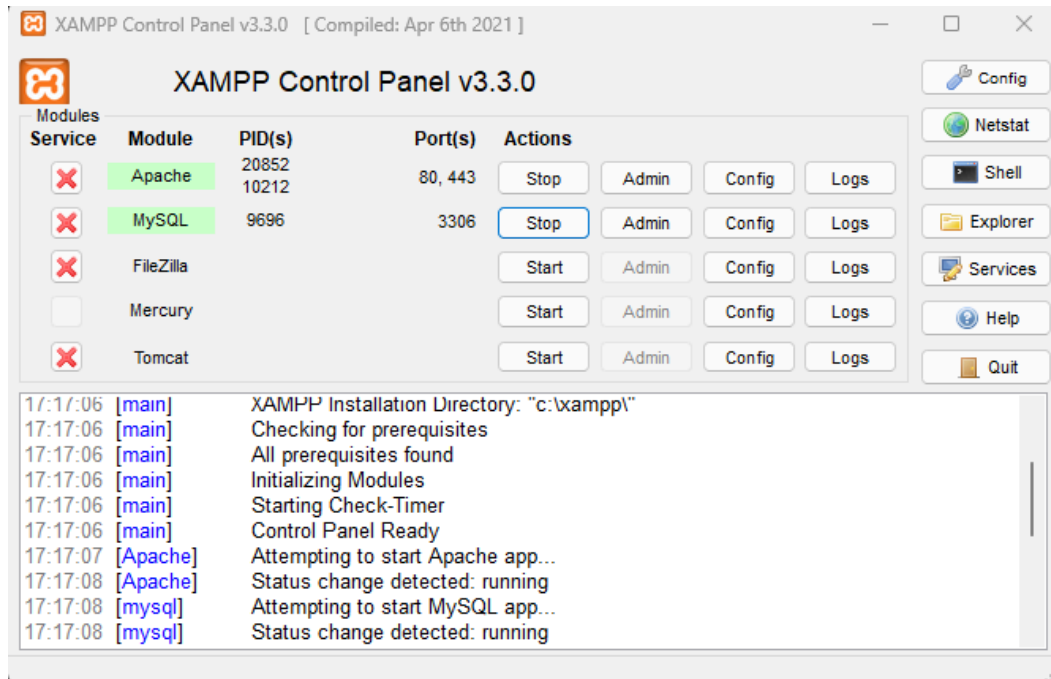
# Where do all the plugins live
plugin_dir="C:/xampp/mysql/lib/plugin/"

# Don't listen on a TCP/IP port at all. This can be a security enhancement,
# if all processes that need to connect to mysqld run on the same host.
# All interaction with mysqld must be made via Unix sockets or named pipes.
# Note that using this option without enabling named pipes on Windows
# (via the "enable-named-pipe" option) will render mysqld useless!
#
# commented in by lampp security
#skip-networking

Ln1, Col1  5.597 caracteres  100%  Windows (CRLF)  UTF-8
```

Inicializar base de datos

Dentro del panel de control de XAMPP, debemos iniciar Apache y MySQL

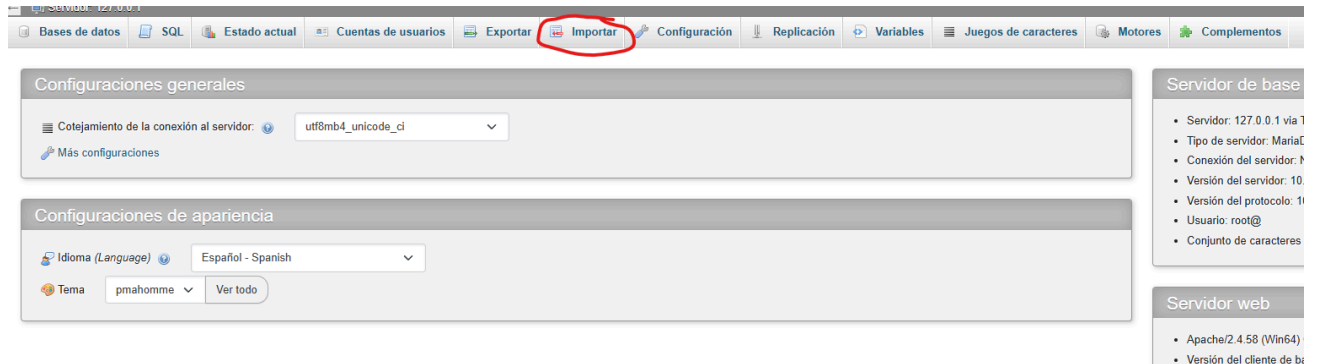


Después debemos ingresar "localhost" en el navegador



Después ingresar en phpMyAdmin,

Dentro debemos hacer clic en importar:



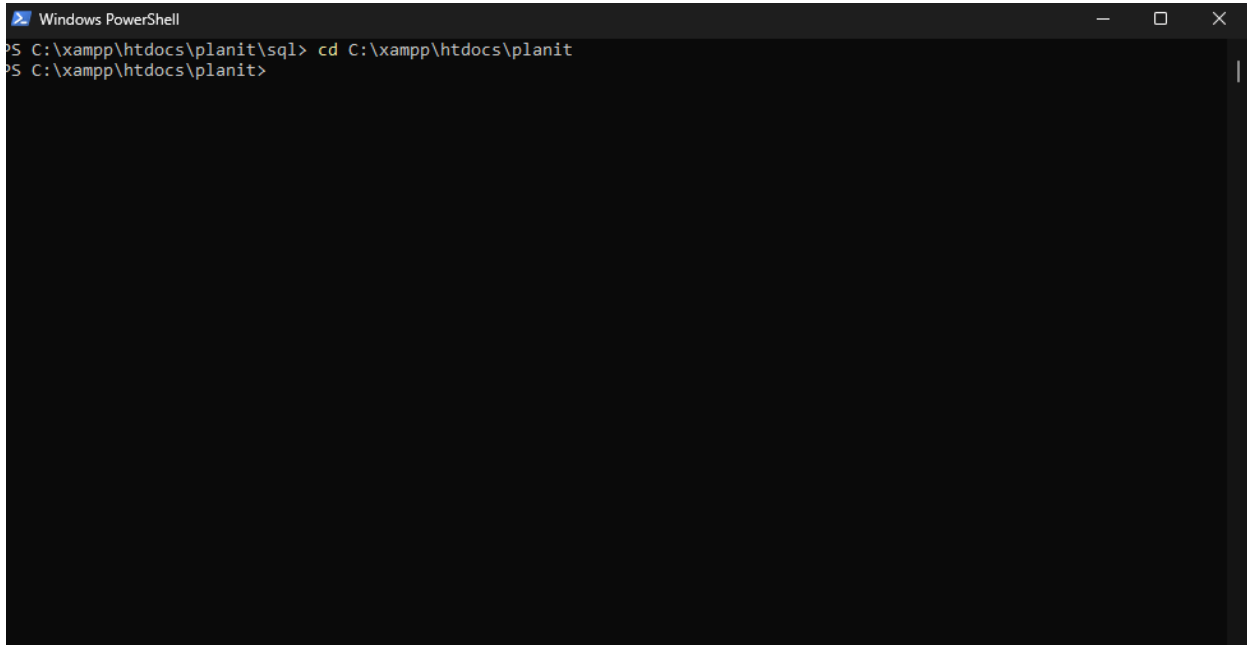
Después, seleccionar el archivo ubicado en /planit/sql/planitdb.zip y hacemos clic en "Importar"

Importando al servidor actual

The screenshot shows the 'Importar' (Import) form in phpMyAdmin. The 'Archivo a importar' (File to import) section is active, showing a text input field with the filename 'planitdb.zip' and a 'Seleccionar archivo' (Select file) button. Below this, there is a message indicating that the file is being loaded and that details are not available. The 'Conjunto de caracteres del archivo' (File character set) is set to 'utf-8'. The 'Importación parcial' (Partial import) section is also visible, with a checkbox for 'Permitir la interrupción de una importación' (Allow interruption of an import) checked. The 'Otras opciones' (Other options) section shows a checkbox for 'Habilite la revisión de las claves foráneas' (Enable foreign key checking) checked. The 'Formato' (Format) section shows 'SQL' selected. The 'Opciones específicas al formato' (Format-specific options) section shows 'Modalidad SQL compatible' (SQL compatibility mode) set to 'NONE' and a checkbox for 'No utilizar AUTO_INCREMENT con el valor 0' (Do not use AUTO_INCREMENT with the value 0) checked.

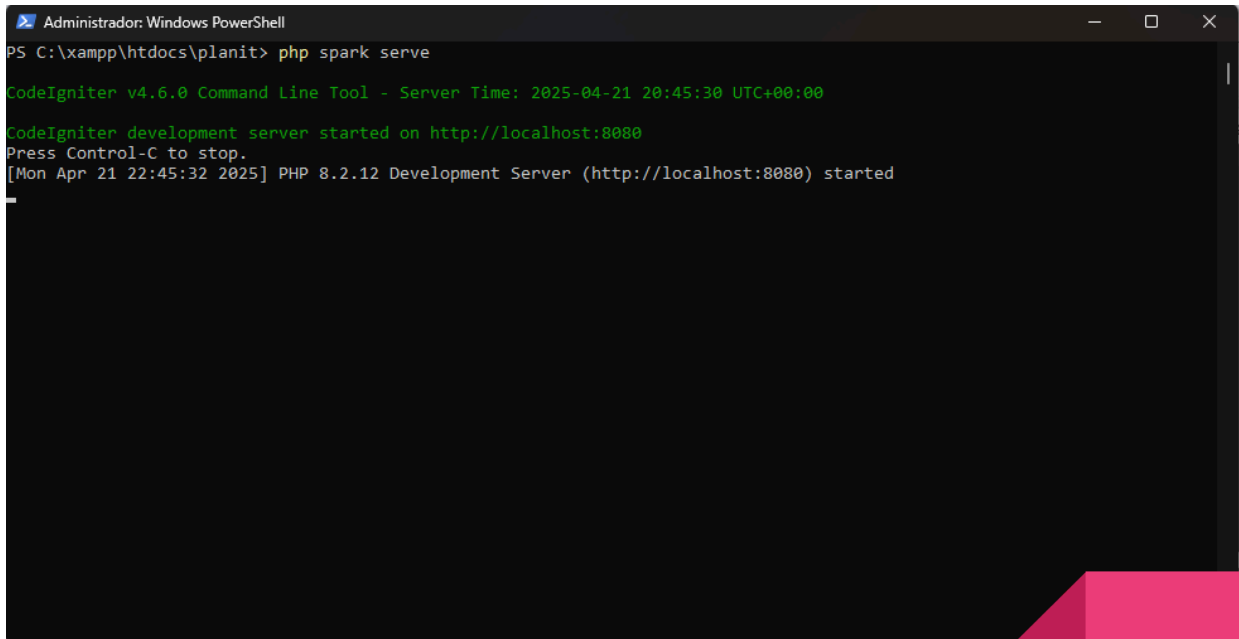
Iniciar Planit

Por último, queda arrancar el servidor de CodeIgniter para iniciar Planit
Abre PoweShell como administrador y navegar hasta la carpeta raíz de nuestro proyecto:



```
Windows PowerShell
PS C:\xampp\htdocs\planit> cd C:\xampp\htdocs\planit
PS C:\xampp\htdocs\planit>
```

Aquí debemos ejecutar “php spark serve” y nuestro servidor arrancará.
Podemos acceder a Planit a través de la url que se nos ha proporcionado. En este caso:
“localhost:8080”



```
Administrador: Windows PowerShell
PS C:\xampp\htdocs\planit> php spark serve

CodeIgniter v4.6.0 Command Line Tool - Server Time: 2025-04-21 20:45:30 UTC+00:00

CodeIgniter development server started on http://localhost:8080
Press Control-C to stop.
[Mon Apr 21 22:45:32 2025] PHP 8.2.12 Development Server (http://localhost:8080) started
```

Guia de usuario

1 Registro

- Ve a la página de inicio de Planit
- Haz click en "Registrarse"
- Completa el formulario con tu nombre de usuario, contraseña, tu ubicación (como puede ser tu ciudad, pueblo, provincia...)
- Opcional: Añade una foto de perfil personalizada

2 Inicio de sesión

- En la página de inicio, haz clic en "Iniciar Sesión"
- Ingresa tu usuario y tu contraseña
- Haz clic en "Iniciar Sesión"

3 Crear un nuevo Plan

- Con la sesión iniciada con tu cuenta, haz clic en "Crear nuevo Plan"
- Completa el formulario con la información del evento:
 - Imagen: La imagen se mostrará a los usuarios como carátula de presentación del evento
 - Título
 - Descripción: Información detallada del evento que verán los usuarios cuando clickean en el evento
 - Fecha
 - Hora
 - Categoría: Selecciona o Crea una nueva categoría donde quieres encasillar tu evento
 - Aforo (opcional)
 - Precio (opcional)
 - Dirección:
 - Puedes Introducir manualmente la dirección
 - O arrastrar la chincheta del mapa en la localización del evento
 - Para publicar el evento, haz clic en "Publicar Plan"

4 Buscar y Filtrar Eventos

- En la página principal, verás los planes activos disponibles para tí
- Puedes usar los filtros para buscar los eventos según:
 - Categoría: Elige entre todas las categorías disponibles
 - Rango de precio: Desde 0 a 100 euros
 - Ciudad: Introduce el nombre de la ciudad
 - Título: Busca un evento en concreto por su título
- Para restablecer los filtros, haz clic en “Reiniciar Filtros”

5 Participar en un Evento

- Dentro del catálogo, haz click en un evento para ver sus detalles
- Si hay plazas disponibles, haz click en “Apuntarse”
- Si el evento es de pago, completa la pasarela de pago con la información requerida
- Una vez apuntado, el evento aparecerá en “Mis Planes”.

6 Gestionar tus Eventos

6.1 Ver tus Eventos Creados

- Haz click en tu foto de perfil y en el menú desplegable, haz clic en “Eventos Creados”
- Verás una lista de los eventos creados separados en activos e inactivos
- Puedes editar o cancelar los eventos desde aquí

6.2 Ver los eventos a los que estás apuntado

- Haz click en tu foto de perfil y en el menú desplegable, haz click en “Mis Planes”
- Verás los eventos en los que te has inscrito separados en pasados y activos
- Puedes desapuntar si lo deseas

7 Interactuar con Comentarios

- En la página de un evento pasado, puedes ver los comentarios de los usuarios que asistieron
- Si asististe o creaste el evento, puedes dejar comentarios
- Puedes eliminar tus comentarios. Si eres el creador del evento, puedes borrar los comentarios de cualquier usuario

8 Perfil de Usuario

- Haz click en tu foto de perfil en la parte superior derecha y haz click en “Mi Perfil”
- Aquí podrás editar:
 - Tu nombre de usuario
 - Tu contraseña
 - Tu foto de perfil
 - Tu ubicación
- Puedes acceder a la vista de tu perfil público haciendo click en “Ver Perfil Público”

9 Bloquear usuarios y administrar usuarios bloqueados

- Dentro del menú desplegable haz clic en “Usuarios Bloqueados”
- Aquí verás la lista de usuarios bloqueados si hay alguno
- Puedes buscar usuarios por su nombre y bloquearlos
- Los usuarios bloqueados no podrán apuntarse a tus eventos ni interactuar contigo

10 Interacción y elogio entre usuarios

- Puedes acceder al perfil de otros usuarios desde sus eventos creados
- Aquí verás la información pública del usuario y sus eventos creados.
- Aquí podrás Elogiar a un usuario, lo que hará que gane Planit Points.

11 Cerrar Sesión

- Haz clic en tu nombre de usuarios y selecciona “Cerrar Sesión”

12 Funciones de administrador

- Si eres administrador:
- Panel de administrador: Gestiona eventos y usuarios
- Categorías: Crea, edita o elimina categorías
- Moderación: Rehabilita, cancela y elimina eventos, categorías o comentarios inapropiados

13 Planit Points

Planit points es un sistema de puntos de experiencia para incentivar la actividad de los usuarios en la aplicación. Los usuarios ganarán puntos por crear eventos, uniéndose a ellos o siendo elogiados.

Según los puntos del usuario, se dividen en:

- De 0 a 100 puntos: Usuario Nuevo
- De 100 a 500 puntos: Usuario
- De 500 a 1000: Usuario Experimentado
- >1000: Usuario Veterano

Bibliografía

- [Documentación de CodeIgniter 4.6.0](#)
- [Documentación de React](#)
- Documentación de [OpenStreetMap](#) y [Leaflet](#) (necesario para su funcionamiento)
- [Documentación de JQuery](#)
- [Documentación de JavaScript](#)
- [Documentación de SQL](#)
- [Documentación de Bootstrap](#)
- [Documentación de CSS](#)
- [Documentación de PHP](#)
- [Documentación de Node.js](#)
- [Documentación de npm](#)
- [Documentación de Composer](#)
- [Documentación de XAMPP](#)