UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Diseño e implementación de una plataforma de intercambio de activos: una perspectiva académica

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA TELEMÁTICA

Autor: Pablo Saura Jiménez

Director: María Dolores Cano Baños

Cartagena, a 1 de junio de 2020

INDICE

Tabla de figuras

Capítulo 1. Introducción

En este primer capítulo hablaremos sobre las plataformas web que proporcionan las herramientas y las infraestructuras sobre las que otros pueden construir y crear valor. Asimismo, definen las reglas de gestión de la actividad creando las condiciones necesarias para las interacciones. De esta forma es posible aunar consumidores y proveedores en un mismo punto virtual, y serán ellos los que creen los activos, es decir, el valor añadido. Son muchos los ejemplos de éxito que emplean plataformas siguiendo este concepto (por ejemplo: Airbnb, Uber, etc.).

1.1 Motivación del proyecto

La motivación de este proyecto viene dada por la necesidad de profundizar en los conocimientos aprendidos en la carrera relacionados tanto con el desarrollo web como la seguridad de esta frente a posibles atacantes. Por ello la creación de una plataforma web para una empresa real Startup Europe Week (SEW) es una gran oportunidad para entender cómo funcionan esta web de intercambio de activos y aprovechar ese conocimiento para aplicarlo a un caso práctico real.

1.2 Objetivos

El objetivo de este Trabajo Fin de Grado es diseñar e implementar una plataforma web de intercambio de activos con un enfoque didáctico. Esta tendrá usuarios de todo el mundo capaces de aportar valor a la web con la creación de eventos (conferencias, exposiciones …) con un enfoque empresarial para que otros usuarios (asistentes) puedan saber fácilmente de qué eventos cerca de ellos pueden ser de su interés. Para crear esta web será necesario dividirla en dos partes una pública donde se podrán ver todos los eventos creados y otra privada para la que será necesario estar registrado para poder crear estos eventos.

1.3 Estructura del Proyecto

El proyecto se dividirá en cuatro partes. El capítulo 2 que explicará las tecnologías utilizadas para la creación de la plataforma web haciendo especial hincapié en lo que es una API REST con Node.js. El capítulo 3 hablará de los tipos de usuarios que existen y lo que pueden hacer y se explicarán en detalle todas las partes de la plataforma, así como su funcionamiento. El capítulo 4 servirá para poner aprueba la seguridad del desarrollo. El capítulo 5 explicará la puesta en producción de la plataforma web. Finalmente, el capítulo 6 serán la conclusión del proyecto.

Capítulo 2. API REST con Node.js y tecnologías usadas para la creación de la plataforma SEW

En este capítulo voy a explicar las distintas tecnologías que se han usado para crear la plataforma web tanto del lado del servidor (*back-end*) como del lado del cliente (*front-end*). Para crear el servidor he implementado una API REST con Node.js + express y Mysql para la base de datos. Para la parte del cliente he usado un motor de plantillas llamado *handlebars* para crear las vistas y *jquery* para hacer el *front-end* dinámico.

2.1 API REST

En este apartado explicaré de forma general qué es una API, qué significa REST y sus características principales.

Representational State Transfer, o REST, describe una arquitectura de servicios web. REST se compone de un conjunto de normas o restricciones para intercambio de datos mediante HTTP que permite a diferentes equipos acceder y manipular representaciones textuales de recursos web mediante un set uniforme y predefinido de operaciones sin estado. REST es un concepto abstracto, no un lenguaje, marco o tipo de software. [1]

Un ejemplo de servidor REST podría ser el servidor de Instagram, al que puedes realizarle peticiones para obtener fotos, añadir dichas fotos o borrarlas. La función principal del servidor es proporcionar a la aplicación una serie de herramientas para poder interactuar con la información.

Una API es una interfaz de programación de aplicaciones, una interfaz que permite que los programas de software para comunicarse con los demás. [2]

Características de REST

- **Protocolo cliente/servidor sin estado**: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. [3]

- **Interfaz uniforme: Las operaciones relacionadas con la manipulación de datos en cualquier sistema REST y la especificación HTTP son cuatro:**
    - POST (Crear): Crea un nuevo recurso.
    - GET (Leer): Recupera un recurso.
    - PUT (Actualización): Actualiza un recurso existente.
    - DELETE (eliminar): Borra un recurso.

- **Manipulación de los datos a partir de la URI**: es un identificador único de cada recurso del sistema REST. La URI nos facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.

- **Cacheable**: debe admitir un sistema de almacenamiento en caché. La infraestructura de red debe soportar una caché de varios niveles.

- **Sistema de capas**: el servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

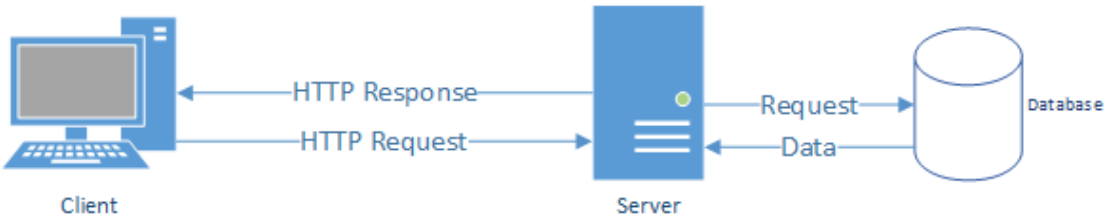


Figura 1 *Api Rest*

2.2 Node.js

Node es una plataforma construida sobre el motor de JavaScript de Google Chrome (V8) que permite un fácil y rápido desarrollo de aplicaciones escalables. Es un entorno que trabaja en tiempo de ejecución, de código abierto, multi-plataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript. La ejecución en tiempo real está pensada para usarse fuera del contexto de un explorador web (es decir, para usare del lado del servidor). Como tal, el entorno omite las APIs de JavaScript específicas del explorador web y añade soporte para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros. [4]

¿Por qué lo he usado en este proyecto?

Las razones por las que he usado Node.js para crear este servidor (back-end) son las siguientes:

- Por su gran rendimiento, Node.js ha sido diseñado para optimizar el rendimiento y la escalabilidad en aplicaciones web y es un muy buen complemento para muchos problemas comunes de desarrollo web, como por ejemplo aplicaciones web en tiempo real.

- El código está escrito en "simple JavaScript", lo que significa que tanto el código del explorador web como del servidor (es siempre JavaScript) como el del lado del servidor son el mismo lo cual facilita el desarrollo de la web.

- Tiene un *footprint* de memoria menor. Es decir, los procesos de Node.js ocupan niveles de memoria sensiblemente menores que los de otros lenguajes, por lo que los requisitos de servidor para atender al mismo número de usuarios son menores. Por aproximar algo, podríamos llegar a tener 1.000 usuarios conectados a la vez y el proceso de NodeJS ocuparía solamente 5 MB de memoria.

- Es portable, con versiones que funcionan en Microsoft Windows, OS X, Linux, Solaris, FreeBSD, OpenBSD, WebOS, y NonStop OS. Además, está bien soportado por muchos de los proveedores de hospedaje web, que proporcionan infraestructura específica y documentación para hospedaje de sitios Node. Entre estos proveedores se encuentra CPanel que es el que usaré para la puesta en producción de la plataforma web.

- Tiene un ecosistema y comunidad de desarrolladores de terceros muy activa por lo que existe muchas y variadas dependencias para Node.js además de mucha documentación.

- Node.js dispone de un ecosistema de librerías llamado NPM, una de las mayores fuentes de librearías *Open Source* del mundo. Esto añade gran valor a la plataforma debido a su modularidad y capacidad de obtener nuevos módulos gracias a la comunidad que lo respalda. (Explicare en detalle NPM en el apartado 2.5)

Aunque a la hora de implementar Node.js en el servidor no lo he usado directamente porque la programación en Node.js para una plataforma web grande, como es este caso, puede ser demasiado complicado. Por ello he usado express.js el cual explicaré en profundidad en el siguiente apartado.

2.3 Express.js

Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. [5]

Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).

- Integración con motores de *renderización* de "vistas" para generar respuestas mediante la introducción de datos en plantillas (html, Handlebars, jade …).

- Establecer ajustes de aplicaciones web como qué puerto usar para conectar y la localización de las plantillas que se utilizan para *renderizar* la respuesta.

- Añade un procesamiento de peticiones *middleware* adicional en cualquier punto dentro de la llamada que maneja la petición.

A pesar de que Express es en sí mismo bastante simple, desarrolladores de todo el mundo han creado paquetes de *middleware* compatibles para abordar casi cualquier problema de desarrollo web. Hay librerías para trabajar con cookies, sesiones, inicios de sesión de usuario, parámetros URL, datos POST, cabeceras de seguridad y muchos más.

Express es **no dogmático**, **transigente**. Esto quiere decir que se puede insertar casi cualquier *middleware* compatible que necesite dentro de una cadena de manejo de la petición, en casi

cualquier orden. Se puede estructurar la aplicación en un fichero o múltiples ficheros y usar cualquier estructura de directorios.

Funcionamiento

En sitios web o aplicaciones web dinámicas, que accedan a bases de datos, el servidor espera a recibir peticiones HTTP del navegador (o cliente). Cuando se recibe una petición, la aplicación determina cuál es la acción adecuada correspondiente, de acuerdo con la estructura de la URL y a la información (opcional) indicada en la petición con los métodos POST o GET. Dependiendo de la acción a realizar, puede que se necesite leer o escribir en la base de datos, o realizar otras acciones necesarias para atender la petición correctamente. La aplicación ha de responder al navegador, normalmente, creando una página HTML dinámicamente para él, en la que se muestre la información pedida, usualmente dentro de un elemento específico para este fin, en una plantilla HTML.

Express posee métodos para especificar qué función ha de ser llamada dependiendo del verbo HTTP usado en la petición (GET, POST, SET, etc.) y la estructura de la URL ("ruta").

También tiene los métodos para especificar qué plantilla ("view") o gestor de visualización utilizar, donde están guardadas las plantillas de HTML que han de usarse y cómo generar la visualización adecuada para cada caso.

Express puede usarse también para añadir funcionalidades para la gestión de cookies, sesiones y usuarios, mediante el uso de parámetros, en los métodos POST/GET. Además, puede utilizarse cualquier sistema de trabajo con bases de datos, que sea soportado por Node.js, en el caso de esta plataforma web se usará una base de datos MySQL.

2.4 NPM

NPM es el sistema de gestión de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript, bajo Artistic License 2.0. [6]

En el desarrollo de este proyecto se usa para:
- Descargar librerías js
- Actualizar en caso de nueva versión las librerías instaladas
- Descargar una versión en específico de la librería
- Gestionar las dependencias entre paquetes

NPM usa un fichero especial llamado *package.json* en el que se declaran las librerías y sus versiones. Esto es necesario para mantener en este archivo un registro con todas las librerías que necesites para que con un solo comando (*npm run install*) se descarguen todas y no sea necesario estar buscándolas en sus respectivos repositorios.

En este archivo, debe estar en la raíz del proyecto, en va a quedar reflejada la configuración del proyecto de Node tales como:

- **Información del proyecto**: donde están los archivos, autores, contacto, licencia, etc.

- **Scripts**: npm nos permite crear nuestros propios comandos que ejecutaran las acciones que establezcamos y para invocarlos escribiremos el `npm [start] ` donde *start* puede ser ejecutar el proyecto node (node.)

- **Dependencies**: son todas las librerías que podemos añadir a nuestro proyecto y cuando ejecutemos *npm install* por defecto se guardarán en la carpeta `node_modules`.

- **DevDependencies**: lo mismo que las dependencias, pero para un entorno de desarrollo

- **Repositorio Git**: Indica el nombre del repositorio git que se está usando

- **Motor de Node**: Indica la versión de Node que se está usando.

Existen dos desventajas a la hora de usar este sistema que debemos tener en cuenta:

- Muchas librerías tienen su propio *package.json* que a su vez llaman a otras dependencias, esto puede producir que la librería pueda tener vulnerabilidades debido a sus dependencias. Estos casos duelen ser avitualles y sus creadores pueden tardar mucho en subir una actualización o a veces incluso abandonar el mantenimiento de la dependencia.

- Relacionado con el punto anterior el hecho de que Node.js esté continuamente en desarrollo implica que es necesario mantener las librerías actualizada constantemente y adaptarla a los nuevos cambios de versión de Node.js.

2.5 Dependencias usadas en la plataforma web

express: Es un *framework* de desarrollo de aplicaciones para Node.js

handlebars: Es una librería de JavaScript que habilita la posibilidad de crear *helpers* personalizados en el servidor.

express-handlebars: Es una librería de JavaScript pensada para para poder *handlebars* con express.

leaflet: Es una librería de JavaScript de código abierto usada para construir aplicaciones de mapeo web. Con un peso de aproximadamente 38 KB de JS, tiene todas las características de mapeo que necesito para esta web. Es compatible con HTML5, CSS3, la mayoría de las plataformas móviles y de escritorio. [7]

Permite visualizar mapas web en mosaico alojados en un servidor público, con superposiciones en mosaico opcionales. Puede cargar datos de características de archivos GeoJSON y crear capas interactivas, como marcadores con ventanas emergentes cuando se hace clic.

leaflet-geosearch: Esta librería de JavaScript agrega soporte para *geocoding* (es el proceso de tomar texto de entrada, como una dirección o el nombre de un lugar, y devolver una ubicación

de latitud / longitud). Viene con controles para incrustar en el mapa, pero para esta plataforma no son necesarios. [8]

La biblioteca utiliza los llamados "*providers*" para encargarse de construir la URL correcta del servicio y analizar los datos recuperados en un formato uniforme. En este trabajo se usará el proveedor OpenStreetMapProvider.

leaflet-gesture-handling: Añade la funcionalidad básica "Gesture Handling" en el un mapa creado con leaflet. [9]

- **Gesture Handling**: Evita que un usuario al usar el *scrol*l del ratón sobre el mapa en vez de desplazarse por la página haga zoom en el mapa. Ahora para hacer zoom en el mapa será necesario mantener pulsada la tecla 'ctrl' a la vez que se usa el *scroll* del ratón.

morgan: Es un *middleware* que nos permite registrar fácilmente solicitudes HTTP, errores y más en la consola de un servidor con node.js. En la plataforma web se ha usado durante el desarrollo para la detección y corrección de errores existentes. [10]

MySQL: Es un módulo para aplicaciones Node.js que habilita la conexión del servidor con una base de datos MySQL. [11]

nodemailer: Es un módulo para aplicaciones Node.js que permite el envío de correos electrónicos de forma sencilla. En la plataforma web se usa para mandar los correos de confirmación al registrarse un usuario, los que informan si un *co-organizer* ha sido aceptado/rechazado, los de recuperación de la contraseña y los que confirman la asistencia a los eventos cuando un usuario decide asistir a uno. [12]
La configuración para usar *nodemailer* en este servidor es la siguiente:

- **host**: startupeuropeweek.eu
- **port**: 465
- **secure**: true
- **auth**: credenciales de autorización.
  - **user**: 'info@startupeuropeweek.eu'
  - **pass**: xxxxxxxxx

express-session: Es un *middleware* de sesión con los siguientes parámetros: [13]

1. **Secret**: Es una clave secreta para firmar las Cookie de sesión.
2. **Resave**: Es una variable booleana que hace que se guarde de nuevo la sesión en la base de datos cada vez que un usuario realiza algún cambio. En este caso tenemos esta opción desactivada.
3. **saveUninitialized**: Obliga a guardar una sesión que no está inicializada en la base de datos. En este caso tenemos esta opción desactivada.
4. **Store**: Es la base de datos donde se guardarán las sesiones.

express-mysql-session: Para que la base de datos guarde las sesiones con MySQL es necesario usar este *middleware*. [14]

Passport: Es *middleware* de autenticación para Node.js. Está diseñado para realizar una tarea concreta: autenticar solicitudes. [15]

passport-local: Es un *middleware* que permite autenticarse utilizando un nombre de usuario y contraseña en sus aplicaciones Node.js. Conectándose con Passport, la autenticación local se integrará fácilmente con Express. [16]

bcryptjs: Es un *middleware* para encriptar las contraseñas de nuestra aplicación. [17]

connect-flash: Es un paquete para Express que nos permite mostrar mensajes en la página bajo ciertas condiciones. Esas condiciones podrían ser cuando un usuario escribe mal su correo o contraseña y le salta un mensaje de error. [18]

crypto-js: Es una librería JavaScript para la implementación de algoritmos criptográficos estándar y seguros. [19]

2.6 Handlebars

Handlebars.js es un motor de plantillas muy popular potente, fácil de utilizar y que cuenta con una gran comunidad. Se basa en el lenguaje de plantillas Mustache, pero lo mejora de distintas maneras. Con Handlebars, podrás separar el diseño HTML del resto de tu Javascript, para así escribir código mucho más limpio. [20]

Un motor de plantillas es un software diseñado para combinar plantillas con un modelo de datos para producir documentos de resultados. El idioma en el que se escriben las plantillas se conoce como lenguaje de plantillas o lenguaje de plantillas. [21]

Helpers

Handlebars no permite escribir Javascript directamente dentro de las plantillas. Para ello, proporciona los *helpers*. Esto son funciones de Javascript a las que se puede llamar desde las plantillas. Los *helpers* se pueden llamar como una expresión {{#nombredelhelper}} y también se les pueden pasar parámetros {{nombredelhelper pram 1 param2 …}}.
Para crear un *helper*, se utiliza la función 'registerHelper'. A continuación, pongo un ejemplo de una función que se utiliza en este proyecto.

```
hbs.registerHelper('ifEquals', function (v1, v2, options) {
    if (v1 === v2) {
      return options.fn(this);
    }
    return options.inverse(this);

  });
```

Figura 2 *Helper de ejemplo*

Esta función crea un *helper* llamado 'ifEquals' que comprueba si los parámetros v1 y v2 son iguales, si lo son *renderiza* el contenido que se encuentre entre la expresión {{#ifEquals}} hasta la expresión {{/ifEquals}}. El siguiente código es un ejemplo para dejarlo más claro.

```
{{#ifEquals pram1 param2}}
     <div> Si pram1 y para2 son iguales renderiza este div </div>
 {{/ifEquals}}
```

Figura 3 *Uso del helper en el código html*

Estos *helpers* serán especialmente útiles en el desarrollo de este proyecto por ejemplo a la hora de ocultar o mostrar contenido html en función de si un usuario está registrado o si es un administrador.

2.7 JQuery

jQuery es un complemento que facilita mucho el desarrollo de aplicaciones enriquecidas del lado del cliente, en Javascript, compatibles con todos los navegadores. Conviene aclarar que jQuery no es un lenguaje, sino una serie de funciones y métodos de Javascript. Por tanto, Javascript es el lenguaje y jQuery es una librería que podemos usar opcionalmente si queremos facilitar la programación en Javascript. [22]

Plugin: jQuery TE

Este plugin sirve para editar un texto. Los cambios que puede realizar son los siguientes: Modificar el tamaño de la letra (3 tipos), escribir en negrita, escribir en cursiva, añadir listas numéricas, justificar el texto a la derecha, justificar el texto al centro, justificar el texto a la izquierda, tachar el texto, añadir enlaces al texto y borrar la modificación echas al texto. [23]
En el desarrollo de este proyecto se utiliza para editar las descripciones de los *co-organizers*, la de los eventos y la de los artículos/noticias.

Figura 4 *Editor de texto personalizado*

2.8 Referencias

[1] REST
https://en.wikipedia.org/wiki/Representational_state_transfer
[2] API REST
https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos
[3] API

https://www.webopedia.com/TERM/A/API.html
[4] Node.js
https://desarrolloweb.com/articulos/intro-nodejs.html
[5] Express.js
https://expressjs.com/
[6] NPM
https://en.wikipedia.org/wiki/Npm_(software)
[7] leafletjs
https://leafletjs.com/
[8] leaflet-geosearch
https://github.com/smeijer/leaflet-geosearch
[9] Leaflet.GestureHandling
https://github.com/elmarquis/Leaflet.GestureHandling
[10] morgan
https://www.npmjs.com/package/morgan
[11] mysql
https://www.npmjs.com/package/mysql
[12] nodemailer
https://nodemailer.com/about/
[13] express-session
https://www.npmjs.com/package/express-session
[14] express-mysql-session
https://www.npmjs.com/package/express-mysql-session
[15] passport
http://www.passportjs.org/docs/
[16] passport-local
http://www.passportjs.org/packages/passport-local/
[17] bcryptjs
https://www.npmjs.com/package/bcryptjs
[18] connect-flash
https://github.com/jaredhanson/connect-flash
[19] cryptojs
https://cryptojs.gitbook.io/docs/
[20] handlebars
https://handlebarsjs.com/
[21] Motor de plantillas
http://www.javahispano.org/antiguo_javahispano_org/2002/6/1/introduccrion-a-los-motores-de-plantillas.html
[22] jquery
https://jquery.com/
[23] jqueryte
http://jqueryte.com/

Capítulo 3. Implementación de la plataforma web SEW

Una vez explicada la estructura usada para la implementación de la plataforma web SEW en el capítulo anterior pasaremos a explicar en detalle los distintos tipos de usuarios que usarán la plataforma, las vistas de la web, todas sus funcionalidades y todas las llamadas al servidor que se usan para obtener, añadir, editar y borrar datos.

3.1 Usuarios

En la plataforma existirán tres tipos de usuarios que tendrán acceso a distintas partes de la web y podrán realizar distintas acciones. Estos son:

- **Usuario no registrado (asistente)**: Los asistentes son los usuarios que visitaran la página con el propósito de buscar eventos a los que les pueda interesar asistir.

- **Usuario registrado (*co-organizers*)**: Los *co-organizers* son usuarios que podrán crear eventos, artículos y editar sus datos personales. Pero para poder crear eventos y artículos necesitaran que su cuenta sea aprobada por algún administrador. Así mismo los eventos y artículos creados también necesitan de la aprobación de algún administrador para que estén disponibles en la web.

- **Administrador**: son los usuarios que se encargan de la gestión de los usuarios, eventos, artículos y también tendrán la posibilidad de crear noticias (obviamente también pueden crear eventos y artículos). En cuanto a los *co-organizers*, los administradores podrán aceptarlos o rechazarlos y también podrán bloquear y desbloquear a un co-organizer previamente aceptado. Todos los eventos tendrán que ser previamente aceptados por algún administrador al igual que los artículos.

3.1.1 Restricciones de seguridad para la plataforma: *middlewares* de autenticación

Aprovechando la capacidad de *express.js* para añadir multitud de *middlewares* en las llamadas al servidor se han creado en este proyecto una serie de *middlewares* de autenticación para que ciertas partes de la plataforma web solo sean accesibles si se cumplen ciertos requisitos. Los *middlewares* de autenticación creados son los siguientes:

- **isLoggedIn**:
  - **Uso**: Protege las rutas para las que un *co-organizer* necesita haber iniciado sesión, pero no necesariamente estar validado por algún administrador (ver su perfil y editar sus datos personales).

  - **Funcionamiento**: Comprueba si el *co-organizer* ha iniciado sesión, verificando en la caché del servidor si el *co-organizer* está autenticado y en caso de que lo esté permite ejecutar la llamada al servidor, en caso de que no lo esté el usuario es re-direccionado a la vista de iniciar sesión.

- **isNotLoggedIn:**

    - **Uso**: Impide que un *co-organizer* que ha iniciado sesión pueda acceder de nuevo a la ruta para iniciar sesión (crear/editar eventos y crear/editar artículos).

    - **Funcionamiento**: Hace justo lo contrario a la función 'isLoggedIn', si el *co-organizer* ha iniciado sesión lo re-direcciona a la vista de su perfil.

- **isValidUser:**

    - **Uso:** Protege las rutas para las que un *co-organizer* necesita haber iniciado sesión y estar validado por algún administrador.

    - **Funcionamiento**: Primero hace la misma comprobación que 'isLoggedIn' y una vez verificado que el *co-organizer* está autenticado comprueba que el estado de este sea válido (estados validos: 'valid', 'admin'). Si es válido permite ejecutar la llamada al servidor, en caso de que no lo esté el usuario es re-direccionado a la vista de su perfil o a la de iniciar sesión si no se está autenticado.

- **isAdmin:**

    - **Uso:** Protege las rutas para las que solo un administrador pueda acceder a ellas.

    - **Funcionamiento**: Comprueba que el estado del usuario es igual a 'admin' si lo es permite ejecutar la llamada al servidor, en caso contrario el usuario es re-direccionado a la vista de su perfil o a la de iniciar sesión si no se está autenticado.

- **isOwnerUserEvent**
    - **Uso:** Protege las rutas para editar los eventos para que solo el *co-organizer* creador o un administrador tengan acceso.

    - **Funcionamiento**: (este *middleware* debe ir acompañado de 'isValidUser'). A partir del identificador del evento obtiene de la tabla 'events' el identificador del *co-organizer* creador de dicho evento y si coincide el identificador del *co-organizer* obtenido a partir de la llamada a la base de datos con el identificador del *co-organizer* que ha iniciado sesión o se es administrador, permite ejecutar la llamada al servidor, en caso de que no coincidan el *co-organizer* es re-direccionado a la vista de su perfil.

- **isOwnerUserblog:**
    - **Uso:** Protege las rutas para editar los artículos para que solo el *co-organizer* creador o un administrador tengan acceso.

    - **Funcionamiento:** (este *middleware* debe ir acompañado de 'isValidUser'). A partir del identificador del artículo obtiene de la tabla 'post' el identificador del *co-organizer* creador de dicho artículo y si coincide el identificador del *co-organizer* obtenido a partir de la llamada a la base de datos con el identificador del *co-organizer* que ha iniciado

sesión o se es administrador permite ejecutar la llamada al servidor, en caso de que no coincidan el *co-organizer* es re-direccionado a la vista de su perfil.

3.2 Tipos de llamadas al servidor

Para entender mejor los siguientes apartados explicaré de forma concreta el uso que se le va a dar en este proyecto a los cuatro tipos de llamadas (GET, POST, PUT, DELETE) en la plataforma web.

- Las llamadas cuyo método es GET sirven para:
  - Obtener una vista de la página + datos de la base de datos.
  - Obtener un componente (*) de la página + datos de la base de datos.
- Las llamadas cuyo método es POST sirven para:
  - Añadir nuevos eventos a la base de datos.
  - Añadir nuevos artículos/noticias a la base de datos.
  - Añadir nuevos usuarios a la base de datos.
- Las llamadas cuyo método es POST sirven para:
  - Editar los eventos de base de datos.
  - Editar los artículos/noticias de base de datos.
  - Editar los datos personales de los usuarios en base de datos.
- Las llamadas cuyo método es DELETE sirven para: (**)
  - Borrar artículos/noticias de base de datos.
  - Borrar Co-organizers rechazados de la base de datos.

* Un componente es una parte de la vista que puede estar compuesta por varios componentes. En ejemplo sería la lista de eventos que podemos ver en la vista *events*.
** Los eventos no se pueden borrar de la base de datos solo se puede cambiar su estado ha completado o cancelado.

3.3 Vistas de la página para usuarios no registrados

Es este apartado se explicará todo lo que puede ver y hacer un usuario sin estar registrado (*Asistente* o *Visitante*) en la plataforma web.

3.3.1 Vista principal “Home”

La vista principal se encargará de mostrar una selección destacada de eventos que se realizarán próximamente, un mapa con todos los eventos que se realizan alrededor del mundo, artículos de los *co-organizers*, noticias de los administradores y las redes sociales de SEW.

3.3.1.1 Carrusel de eventos destacados

En la pestaña principal, se mostrará información relevante de SEW y se invitará a los usuarios a participar en los eventos como asistentes o *co-organizers*. En las demás pestañas muestra los cinco eventos más recientes.

- En la parte superior de la imagen se puede ver la barra de navegación que explicaré en el apartado “Componentes auxiliares”

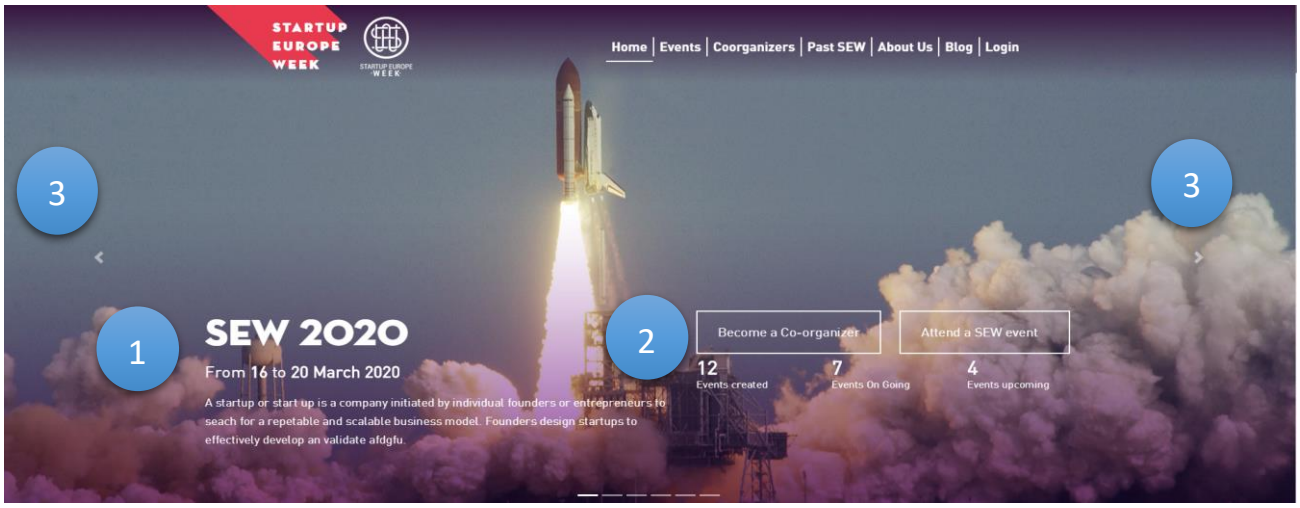


Figura 5 *Carrusel de eventos destacados*

Lista de elementos destacados en el carrusel:

1. Indica la edición SEW de este año, la fecha en la que ocurrirá dicha edición y una breve descripción de la "startup" (en la imagen dicha descripción todavía no es la definitiva)

2. En la parte superior muestra dos botones uno para registrase como *co-organizer* llamado "*Become a Co-organizer*" y otro para poder ver todos los eventos actualmente disponibles llamado "*Attend a SEW event*". En la parte inferior se pueden ver una serie de datos sobre los eventos: La cantidad de eventos creados, la cantidad de eventos activos y la cantidad de eventos que están pendientes de validación. (Los datos en la imagen son solo de ejemplo)

3. En los laterales se pueden ver unos botones con forma de flecha que sirven para moverse por el carrusel para ver los eventos de este. La imagen inferior muestra un ejemplo de cómo se verían estos eventos.

Figura 6 *Eventos del carrusel*

Llamada al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/"
- **Middleware de autenticación**: ninguno.
- **Funcionalidad**: *Renderiza* la página principal "home" de la página y obtiene tres datos de base de datos, el número de eventos total, el número de eventos en estado "On Going" y el número ciudades en las que ocurren los eventos. También obtiene los cinco últimos eventos cuyo estado es "On Going" para mostrarlos en el carrusel de eventos.

3.3.1.2 Mapa de los eventos

Esta sección muestra un mapa del mundo con iconos que señalan la localización de los eventos activos. Cuando pulsamos sobre alguno de estos iconos aparecerá una caja lila con el título del evento en la parte superior y la dirección del mismo en la parte inferior. Si pulsamos en el título nos re-direcciona la vista del evento.

En la parte superior del mapa se puede ver una barra de búsqueda personalizada que posee tres buscadores diferentes. El primero es un buscador genérico que a partir de una cadena de texto dibujará en el mapa todos los eventos cuyo título, país, región, ciudad o nombre del *co-organizer* coincida total o parcialmente con la cadena proporcionada. El segundo es un selector con todos los países en los que se realiza algún evento que devolverá todos los eventos del país seleccionado. El tercer buscador es igual al segundo, pero con las ciudades en las que trascurren los eventos.

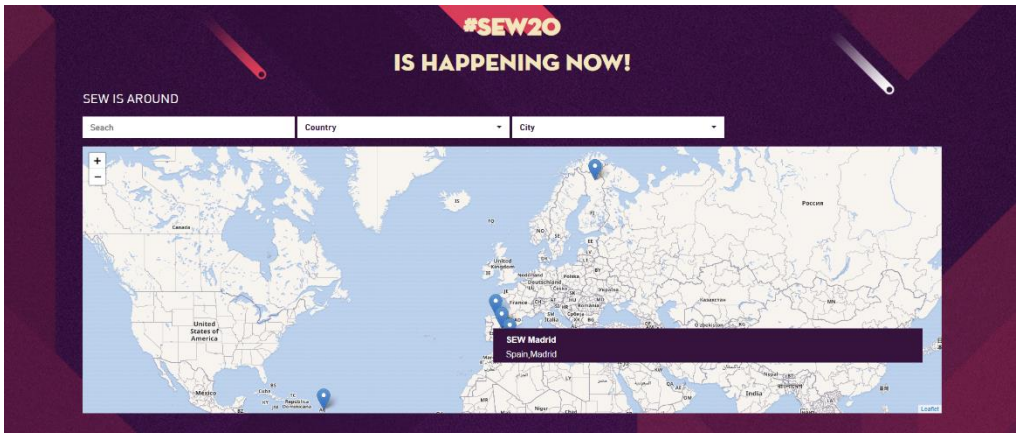


Figura 7 *Mapa de eventos*

Llamada al servidor

- **Método de Solicitud**: GET (X)
- **Ruta**: "/eventsCo/coordinates/xy"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: Devuelve una lista con las coordenadas de todos los eventos cuyo estado sea "On-going" o "Completed".

3.3.1.3 Eventos de “Home”

Esta sección muestra los tres eventos activos más recientes, en la parte inferior derecha la caja roja contiene un botón llamado “See More” que sirve para ver más eventos. En la parte superior está la barra de búsqueda con cuatro buscadores diferentes. Los tres primeros (buscador genérico, selector por país y selector por ciudad) funcionan de la misma manera que los buscadores del mapa, aunque estos solo mostrarán los tres eventos más recientes que coincidan con la búsqueda realizada. El ultimo buscador es un selector por año que devolverá tres eventos del año seleccionado.

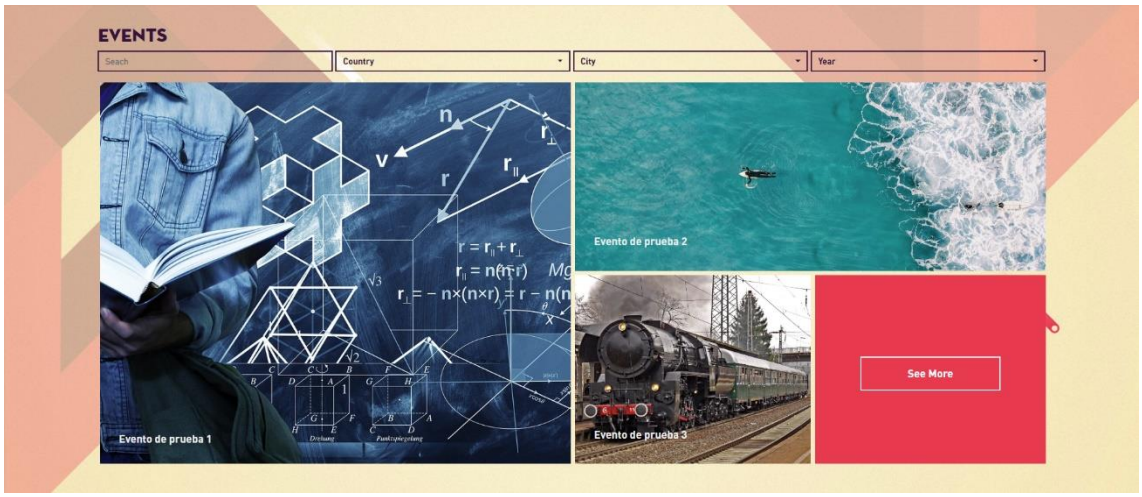


Figura 8 *Eventos de “home”*

Llamada al servidor

- **Método de Solicitud**: GET
- **Ruta**: “eventsCo/public/home?year=Int&country=String&city=String&text=String”
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: En función de los parámetros que contenga la ruta funcionará de forma diferente. Las cinco formas de funcionar son las siguientes:
    - **Sin parámetros**: Obtiene los 3 eventos más ecientes cuyo estado sea igual a ‘On-Going’ o ‘Completed’ y con esta información se *renderiza* el componente ‘partials/events/eventshome”.

    - **País (year):** A partir del nombre de un país comprobará en la tabla “events” de la base de datos los eventos cuyo país coincide con el especificado devolviendo tres de estos eventos y con esta información se *renderiza* el componente “partials/events/eventshome”.

    - **Cuidad (city)**: A partir del nombre de una ciudad comprobará en la tabla “events” de la base de datos los eventos cuya ciudad coincide con la especificada, devolviendo tres de estos eventos y con esta información se *renderiza* el componente “partials/events/eventList”.

- **Año (year)**: A partir de un año comprobará en la tabla "*events*" de la base de datos los eventos que se realizaron el año especificado devolviendo tres de estos eventos y con esta información se *renderiza* el componente "partials/events/eventshome".

- **Texto (text)**: A partir de una cadena de texto comprobara en la tabla "events" de la base de datos los eventos cuyo nombre, país, región, ciudad o nombre del *co-organizer* creador coinciden parcial o totalmente con la cadena de texto introducida devolviendo tres eventos que cumplen esta restricción y con esta información se *renderiza* el componente "partials/events/eventshome".

3.3.1.4 Fotos de "Home"

Esta sección muestra un carrusel personalizado con una serie de fotos asociadas cada una a un evento en concreto. En la parte superior se puede ver un selector que filtra las fotos por año. En el lateral derecho aparece una lista de fotos que el usuario puede clicar si quiere ver a mayor tamaño en la parte central. En la parte inferior aparece un texto con el título del evento asociado a dicha imagen, que también es un enlace al evento.

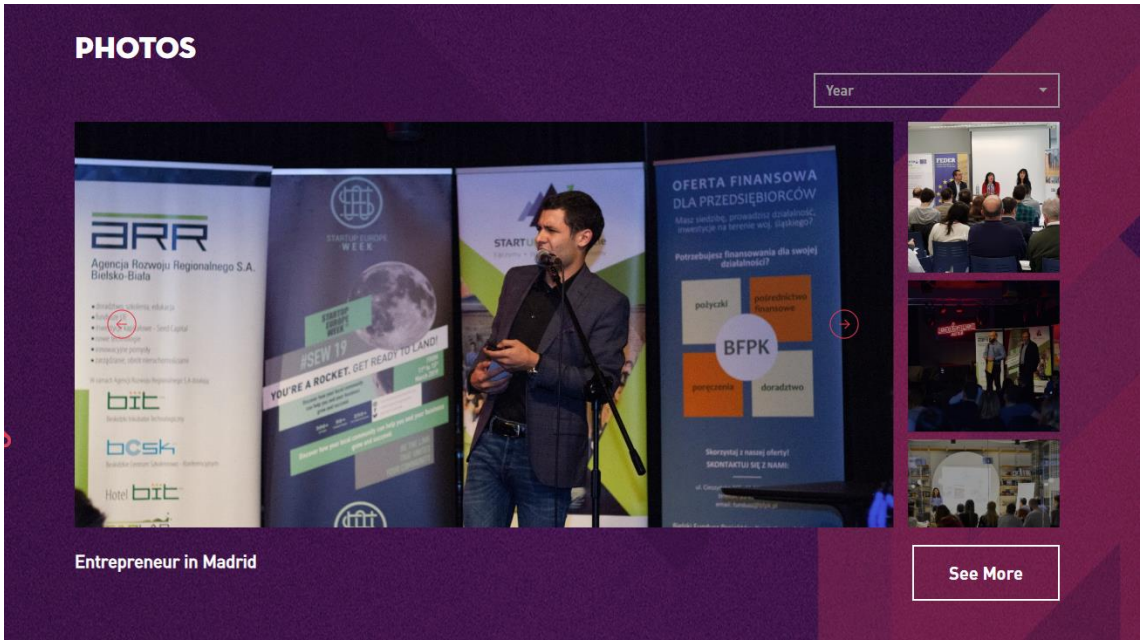


Figura 9 *Fotos de "home"*

Llamada al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/eventsCo/photo/home"
- ***Middleware*** de autenticación: isValidUser.
- **Funcionalidad**: Busca en la tabla "photos" todas las fotos asociadas a un evento cuyo estado sea 'On-Going' o 'Completed' y con esta información *renderiza* el componente "partials/photo/photo_list "

3.3.1.5 JOIN US

Es un pequeño formulario compuesto por el primer nombre, el segundo nombre y el correo, que sirve para inscribirse al boletín de noticias de la plataforma SEW, recibiendo correos con las novedades de este.

Figura 10 *Join us*

3.3.1.6 Artículos de "Home"

Esta sección muestra los tres artículos más recientes creados por un *co-organizer* o administrador que previamente han sido validados por algún administrador. Cada artículo se compone de 4 partes: la imagen principal del artículo, el título del mismo, el autor (*co-organizer*) creador del este, las tres primeras líneas del primer párrafo del artículo y debajo del todo un botón llamado "Visit" que te re-direcciona a la vista en detalle del artículo.

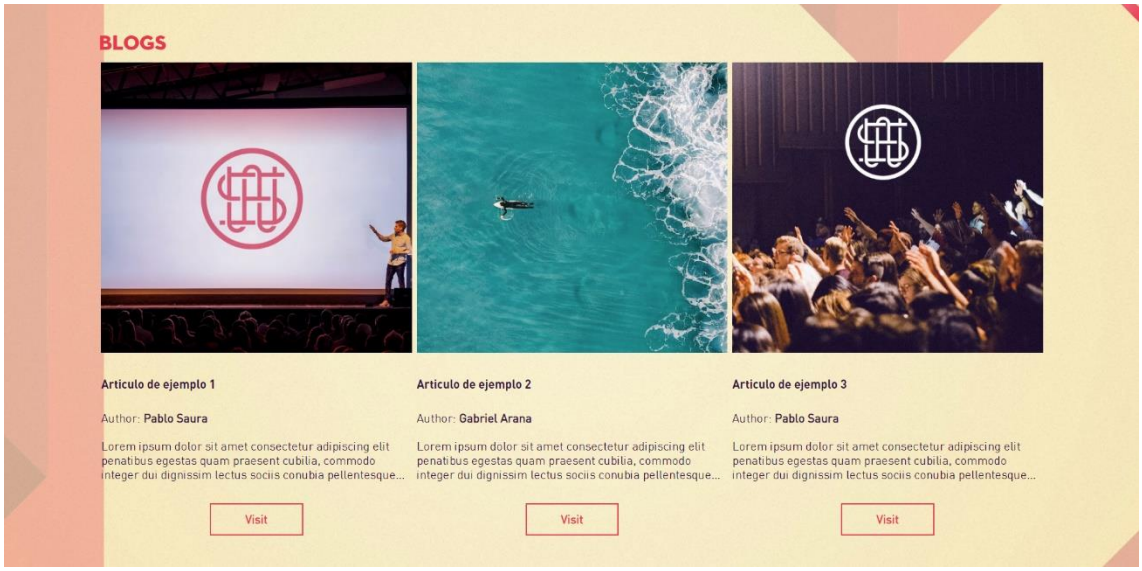


Figura 11 *Artículos de " home"*

<u>Llamada al servidor</u>

- **Método de Solicitud**: GET
- **Ruta**: "/blog/public/home"

- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: Obtiene los 3 artículos más recientes cuyo estado es igual a “valid”. Con esta información *renderiza* el componente “partials/blog/blog_top”.

3.3.1.7 Noticias de “Home”

Esta sección muestra la tres últimas noticias publicadas por algún administrador. Cada noticia se compone de 4 partes: la imagen principal de la noticia alternara la posición de esta de izquierda a derecha al igual que el contenido de la noticia, el título del mismo, los dos primeros párrafos de la noticia (si caben) y debajo de cada noticia un botón llamado “Read more” que te re-direcciona a la vista en detalle de la misma. Debajo del todo un botón llamado “All new” que te re-direcciona a la vista de las noticias.

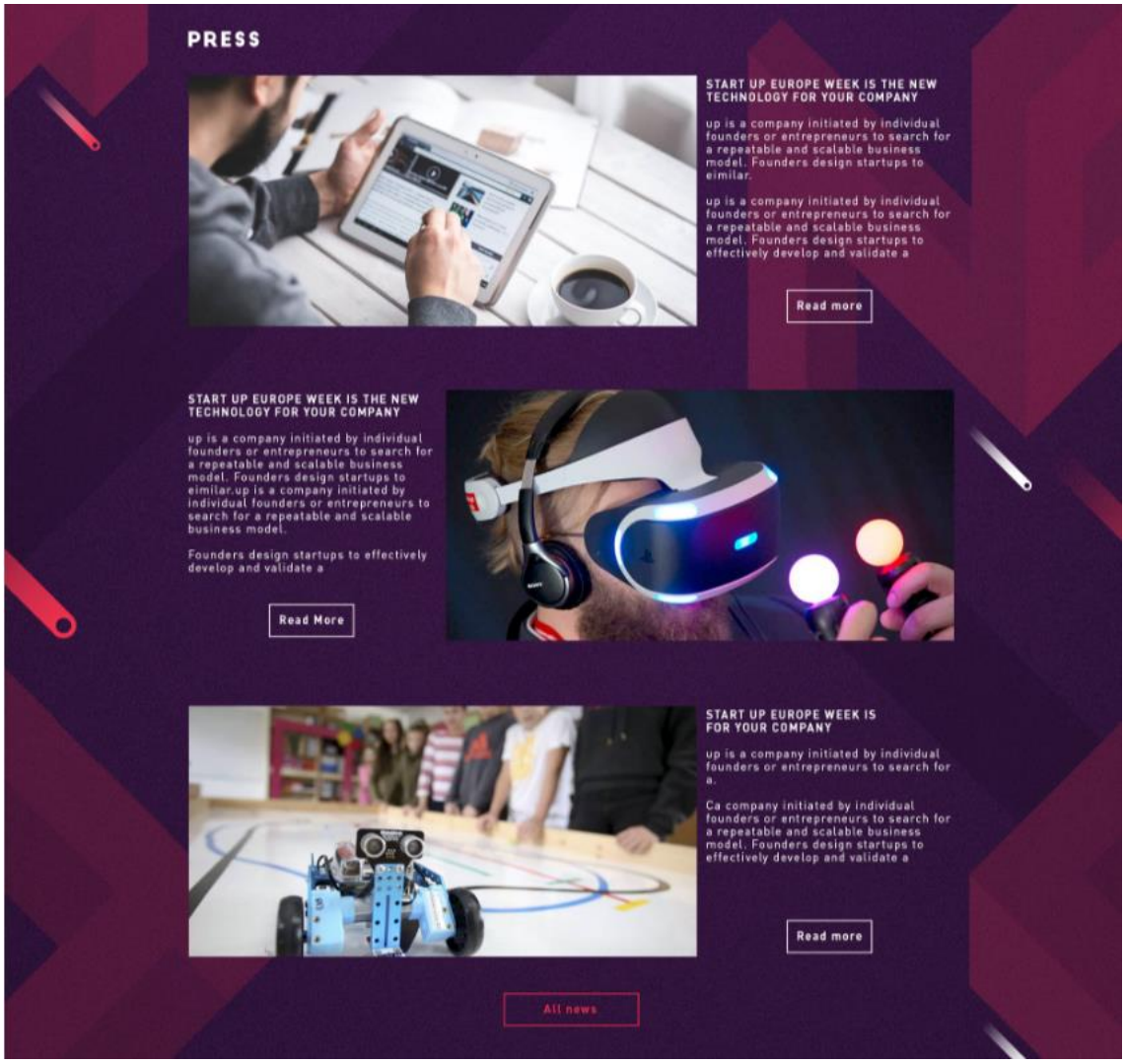


Figura 12 *Noticias de ”home”*

<u>Llamada al servidor</u>

- **Método de Solicitud**: GET (X)

- **Ruta**: "/blog/public/home/new"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: Obtiene las tres noticias más recientes cuyo estado es igual a "admin". Con esta información *renderiza* el componente "partials/blog/new_top".

3.3.1.8 Social media de "Home"

Esta sección muestra las redes sociales de SEW; de izquierda a derecha serían Facebook, Twitter, Instagram y a la derecha del todo hay otra breve descripción de la temática de SEW.

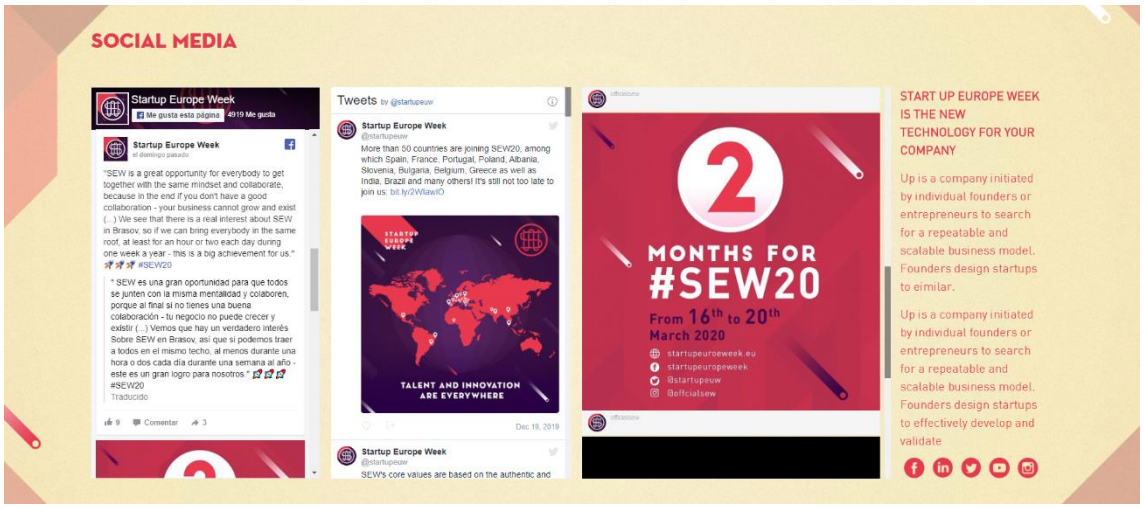


Figura 13 *Social media*

<u>Llamada al servidor</u>

- **Método de Solicitud**: GET
- **Ruta**: "/social/top"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: *Renderiza* el componente que muestra las redes sociales de SEW en la vista "home", estas son Facebook, Twitter, LinkedIn e Instagram.

3.3.2 Vista de los eventos "Events"

La vista "events" mostrará eventos activos para esta edición de SEW y de ediciones anteriores. En principio solo se *renderizan* los tres más recientes, pero se pueden ver más usando los buscadores de la parte superior para que un posible asistente encuentre un evento concreto o pulsando el botón "*load more*" de la parte inferior central para cargar más eventos.
A continuación, explicaré en detalle el funcionamiento de los dos buscadores:

- **Buscador Genérico**: Está compuesto por un único campo al que le pasara una cadena de texto y obtendrá una lista de los eventos cuyo título, país, región, ciudad o nombre del *co-organizer* creador coincidan total o parcialmente con la cadena de texto introducida.
- **Buscador personalizado**: Está compuesto por tres selectores diferentes.
    - **Selector por país**: a partir de una lista con todos los países para los que existe algún evento podemos seleccionar los que ocurran solo en uno de los países.

- **Selector por Ciudad**: a partir de una lista con todas las ciudades para las que existe algún evento podemos seleccionar los que ocurran solo en una de las ciudades.
- **Selector por Año**: a partir de una lista de años podemos seleccionar uno en concreto para ver todos los eventos de ese año.

En cuanto al diseño, cada evento este compuesto por dos partes principales que van alternando de izquierda a derecha, la portada del mismo y su contenido (debajo de este se encuentra un botón "*Read More*" que al pulsarlo te re-direcciona a la vista del evento). Las cuatro partes del contenido del evento son:

- **Título + descripción:** Muestra el título del evento y parte de su descripción (normalmente un párrafo).
- **Localización:** Muestra la dirección del evento, el país y la ciudad en la que ocurrirá. Además, si pulsas en el icono te re-direccionara a "google maps".
- **Fecha:** Muestra el mes y el día del evento en la parte superior, en caso de que dicho evento solo dure un día; si durara más, también aparecería el día que acaba el evento. Justo debajo aparece la hora a la que empieza y a la que acaba el evento.
- **Horario:** Muestra el horario del primer día del evento, en el caso de que durará más de un día, los horarios para los siguientes días se verán en la vista del propio evento.

Figura 14 *Lista de eventos*

Llamadas al servidor de esta vista

- **Método de Solicitud**: GET
- **Ruta**: "/events/public/list"
- ***Middleware* de autenticación**: ninguno.
- Funcionalidad: *Renderiza* la vista que muestra una lista de los eventos que puede ver todo el mundo, esta se llama "events/public_list".

- **Método de Solicitud**: GET
- **Ruta**: "eventsCo/list/selected?year=Int&country=String&city=String&text=String&num=Int"
- ***Middleware* de autenticación**: ninguno
- **Funcionalidad**: En función de los parámetros que se le pase funcionará de diferente manera. Puede ser usada de cinco formas diferentes:
    - **Sin parámetros**: Obtiene los 3 eventos más recientes cuyo estado sea igual a 'On-Going' o 'Completed' y con esta información se *renderiza* el componente 'partials/events/eventList".
    - **País (*year*):** A partir del nombre de un país comprobará en la tabla "events" de la base de datos los eventos cuyo país coincide con el especificado devolviendo tres de estos eventos y con esta información se *renderiza* el componente "partials/events/eventList".
    - **Cuidad (*city*)**: A partir del nombre de una ciudad comprobará en la tabla "events" de la base de datos los eventos cuya ciudad coincide con el especificado devolviendo tres de estos eventos y con esta información se *renderiza* el componente "partial*s/events/eventList".*
    - ***Año (year*)**: A partir de un año comprobará en la tabla "events" de la base de datos los eventos que se realizaron el año especificado devolviendo tres de estos eventos y con esta información se *renderiza* el componente "partials/events/eventList".
    - **Texto (*text*)**: A partir de una cadena de texto comprobará en la tabla "events" de la base de datos los eventos cuyo nombre, país, región, ciudad o nombre del *co-organizer* creador coinciden parcial o totalmente con la cadena de texto introducida devolviendo tres eventos que cumplen esta restricción y con esta información se *renderiza* el componente "partials/events/eventList".
    - **Contador (nun)**: Obtiene tres eventos empezando desde el que se encuentra en la posición "nun" y con esta información se *renderiza* el componente "partials/events/eventList".

3.3.2.1 Vista publica de un evento

Esta vista contiene toda la información de un evento concreto, para explicar mejor la dividiré en tres partes:

Información básica del evento

En la parte superior se mostrará la portada del evento y en el caso de que el *co-organizer* creador añadiera fotos al evento, la portada se convertirá en un carrusel de imágenes que además de contener la propia portada contendrá todas las fotos añadidas al evento. Justo debajo mostrará la información principal del evento compuesta por las mismas cuatro partes que en la vista de eventos anuqué dos de las partes son un poco diferentes:

- **Título + descripción:** La descripción muestra todo el contenido, no solo el primer párrafo.
- **Horario:** Muestra el horario de todos los días que dure el evento.

Debajo de la información del evento disponemos de dos botones uno llamado "*Share*" para compartir el evento en las redes sociales (Twitter, Facebook y LinkedIn) y otro llamado "*Attend*" que re-direccionará al usuario a la vista para asistir al evento. Debajo del todo están los iconos con las redes sociales del *co-organizer* (Twitter, Facebook, Instagram y LinkedIn).

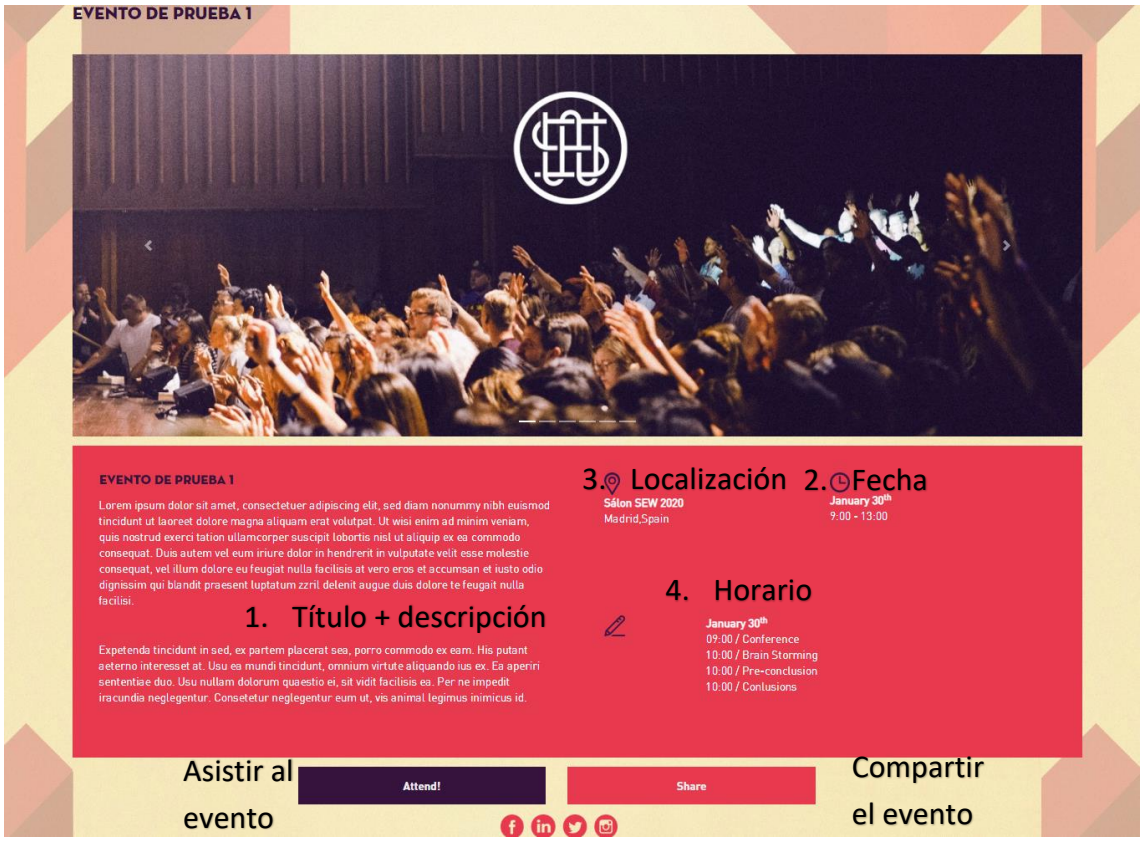


Figura 15 *Información del evento*

Llamada al servidor de esta vista

- **Método de Solicitud**: GET
- **Ruta**: "/events/:id"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: A partir del id de un evento busca en la tabla "events" de la base de datos se obtienen los parámetros del evento y con esta información *renderiza* la vista "events/view".

Vista de asistencia al evento

En esta vista un asistente a un evento tendrá que escribir una serie de datos personales para poder asistir a al evento (los que contienen un asterisco son obligatorios) los datos son los siguientes: correo de contacto “Email”, nombre “First Name”, apellido/s “Last Name”, Edad “Age”, Ocupación “Occupation” y país de procedencia “Country”. Una vez añadidos todos estos datos pulsando el botón “GO!” el usuario será registrado como un nuevo asistente al evento, se le mandará un correo de confirmando que asistirá al evento y será re-direccionado a una vista de confirmación. Si el asistente también es un *co-organizer,* los campos de los datos personales se rellenarán automáticamente con los datos del *co-organizer*.

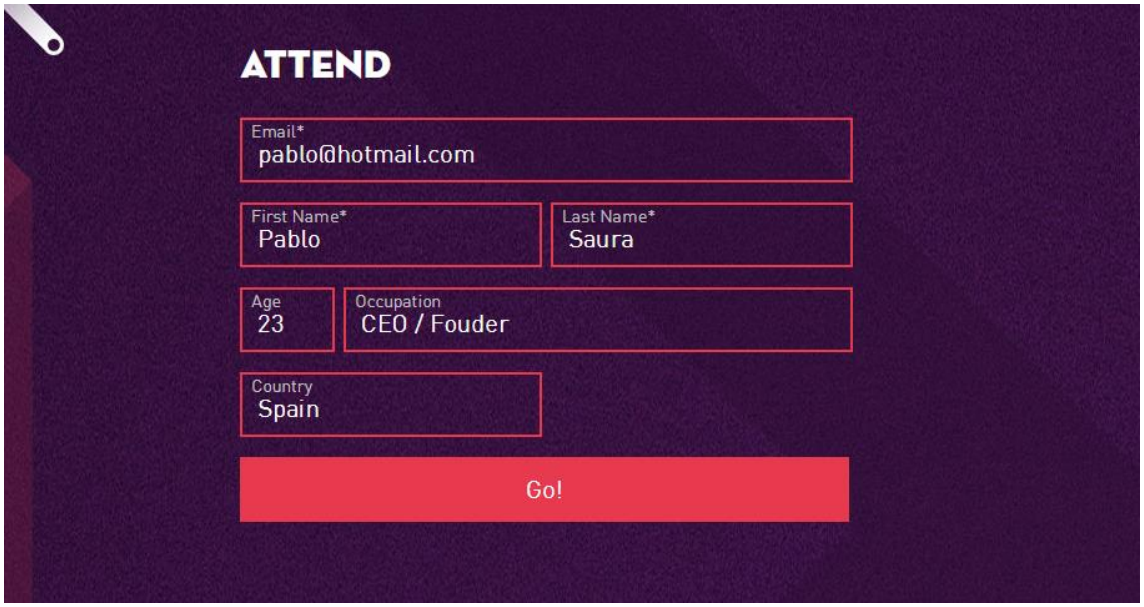


Figura 16 *Formulario de asistencia*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: “/events/assistant/assist/:id
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: A partir del identificador de un evento *renderiza* la vista “events/assist” para que una persona pueda asistir al evento.

- **Método de Solicitud**: POST
- **Ruta**: “/events/assistant/assist”
- ***Middleware* de autenticación**: ninguno.
- **Variables del body**: first_name, last_name, email, age, occupation, id_event,country.
- **Funcionalidad**: A partir de las variables necearías para crear un asistente, añade en la tabla ‘assistants’ de la base de datos una nueva línea con la información del nuevo asistente.

En la vista de confirmación aparecerá el Ticket del asistente al evento con la información principal de este y el nombre del *co-organizer*. En la parte derecha hay dos botones uno llamado "Save" que sirve para descargar un pdf con el Ticket y otro llamado "Send to me by Email" para enviar otro correo de confirmación al asistente.

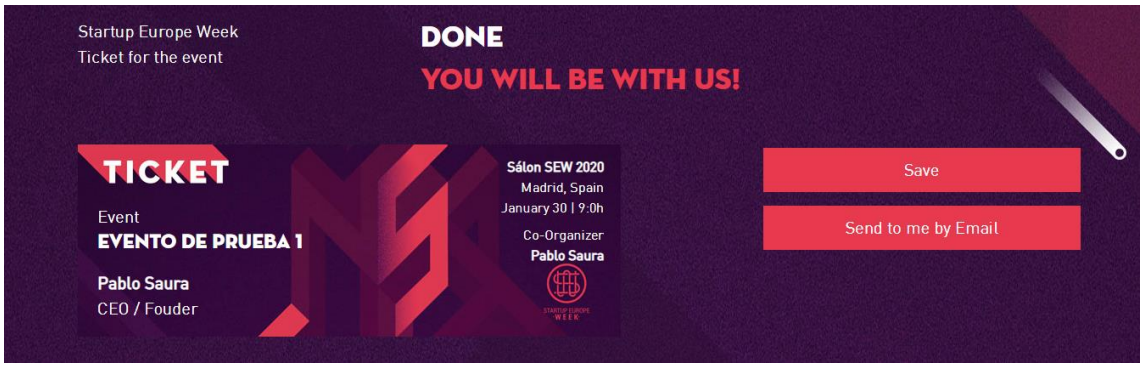


Figura 17 *Ticket de confirmación*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/events/assistant/ confirmation/:id
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: A partir del identificador de un evento *renderiza* el componente "partials/events/assist_confirmation" que sirve que un asistente pueda ver y descargarse el Ticket del evento al que asistirá.

- **Método de Solicitud**: POST
- **Ruta**: "/events/assistant/ email"
- ***Middleware* de autenticación**: ninguno.
- **Variables del body**: email.
- **Funcionalidad**: A partir del correo de un asistente envía un correo de confirmación del evento.

Participantes del evento

En esta sección aparecerán todos los participantes del evento en cuatro secciones:

- **Co-organizer**: Aparece en un perfil con la información del *co-organizer* creador del evento
- **Sponsors**: Aparecen los perfiles con la información de los *sponsors* del evento.
- **Speakers**: Aparecen los perfiles con la información de los participantes que hablarán en el evento.
- **Moderator:** Aparece el perfil con la información del moderador del evento.

Figura 18 *Participantes de un evento*

En la llamada al servidor ““/events/:id” también se obtienen toda la información de los participantes.

<u>Eventos relacionados</u>

En esta sección aparecerán dos eventos relacionados con el evento de la vista.

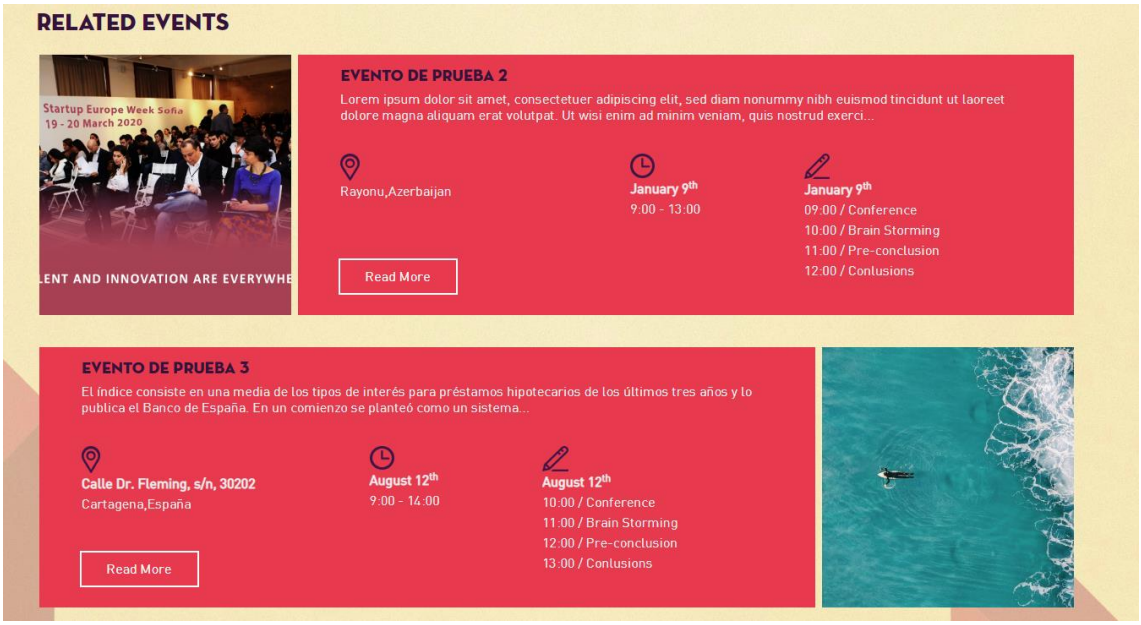


Figura 19 *Eventos relacionados*

Llamada al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/eventsCo/list/related/?id=Int&user_id=Int"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: A partir del identificador del usuario del *co-organizer* busca en la tabla "events" de la base de datos los dos eventos más recientes de este cuyo estado sea "On-Going" o "Completed", en caso de no tener ningún evento creado se obtienen los dos últimos eventos cuyo estado sea "On-Going" o "Completed" y con esta información *renderiza* el componente "partials/events/related_events".

3.3.3 Vista de los *co-organizers*

La vista *co-organizers* mostrará los *co-organizers* validados por el administrador. En principio solo se *renderizan* 8, pero se pueden cargar más pulsando el botón "*load more*" que se encuentra debajo. También se puede buscar a un *co-organizer* en concreto usando el buscador que se encuentra arriba a la izquierda. Cada *co-organizer* muestra cuatro elementos: su imagen de perfil (si no tiene se muestra una imagen por defecto), su nombre, su ocupación y su bio. Clicando en su imagen de perfil seremos re-direccionados a su perfil público de usuario.

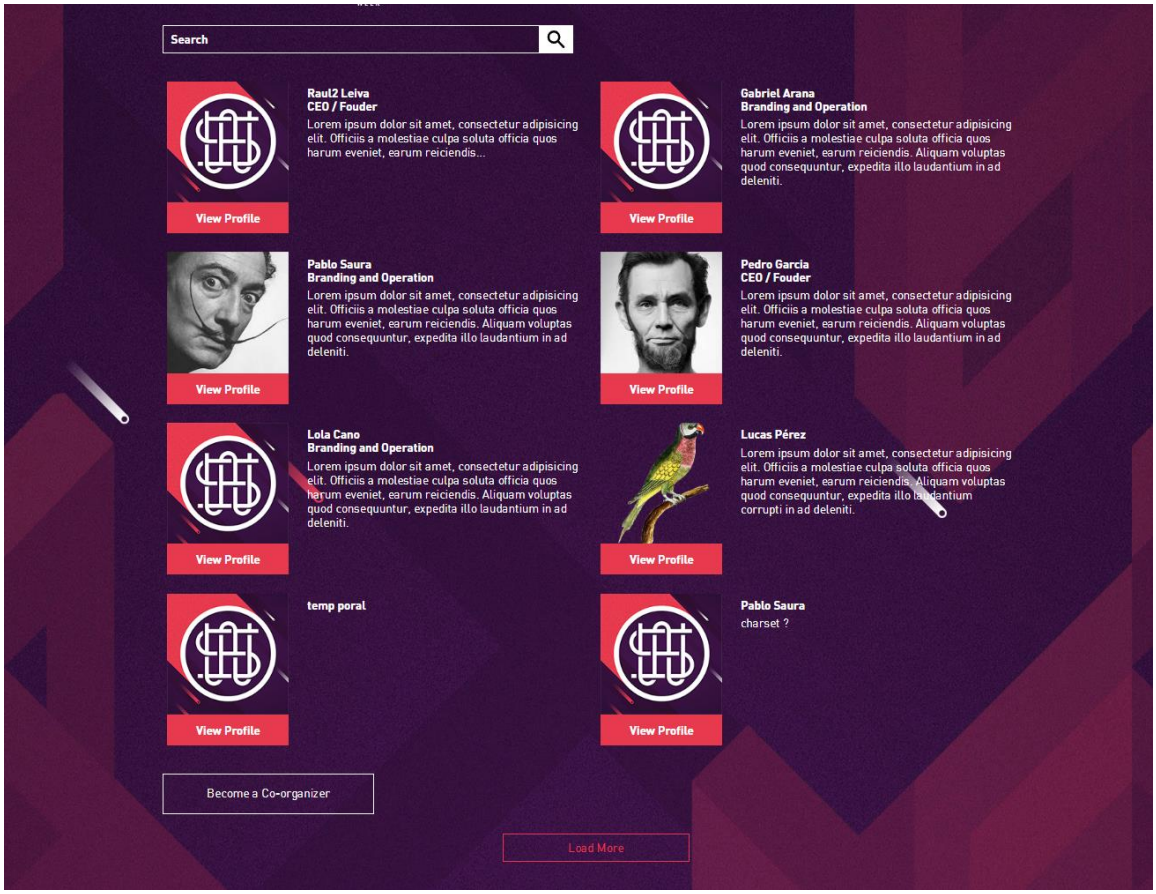


Figura 20 *Lista de Co-organizers*

Llamadas al servidor de esta vista

- **Método de Solicitud**: GET
- **Ruta**: "/profile/Coorganizer/list"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: *Renderiz*a la vista que mostrara una lista publica de *co-organizers*, esta se llama "profile/list".

- **Método de Solicitud**: GET
- **Ruta**: "/profile/list/select?text=String&nun=Int"
- ***Middleware*** de autenticación: ninguno.
- **Funcionalidad**: En función de los parámetros que se le pase funcionará de diferente manera. Puede ser usada de tres formas diferentes:
    - **Sin parámetros**: Obtiene los 8 primeros *co-organizers* cuyo estado sea igual a 'valid' y con esta información se *renderiza* el componente 'partials/profile/profileList"
    - **Texto (text)**: A partir de una cadena de texto comprobara en la tabla "users" de la base de datos si existen *co-organizers* cuyo nombre, país, región o ciudad coinciden parcial o totalmente con la cadena de texto introducida se devolverá una lista con los *co-organizers* y con esta información se *renderiza* el componente ''partials/profile/profileList"
    - **Contador (nun):** Obtiene 8 *co-organizers* empezando desde el que se encuentra en la posición "nun" cuyo estado sea igual a 'valid' y con esta información se *renderiza* el componente 'partials/profile/profileList"

3.3.3.1 Vista publica de un Co-organizer

Esta vista contiene toda la información de un Co-organizer concreto la imagen de perfil de usuario, su nombre, su ocupación, su email de contacto y su bio. Debajo de su imagen de perfil hay unos iconos con las redes sociales del Co-organizer (Twitter, Facebook, Instagram y LinkedIn).

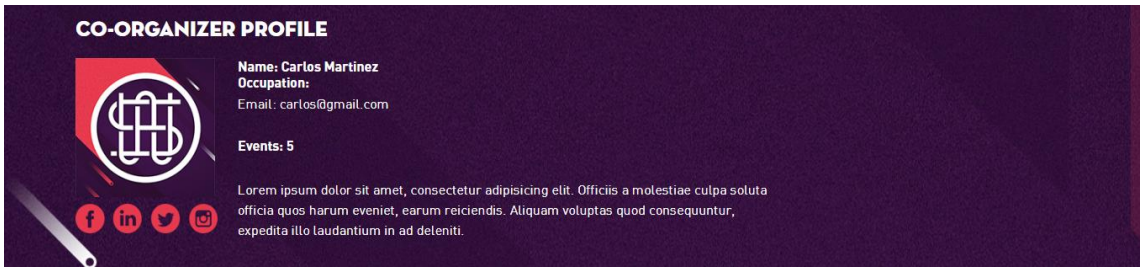


Figura 21 *Perfil del Co-organizer*

Llamada al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/profile/: id"
- ***Middleware* de autenticación**: ninguno.

- **Funcionalidad**: A partir del id del *co-organizer* busca en la tabla “users” de la base de datos tanto los datos personales del *co-organizer* como el número de eventos que ha creado este y con esta información *renderiza* la vista “profile/index”.

En esta sección aparecerán dos eventos relacionados con el *co-organizer* de la vista.

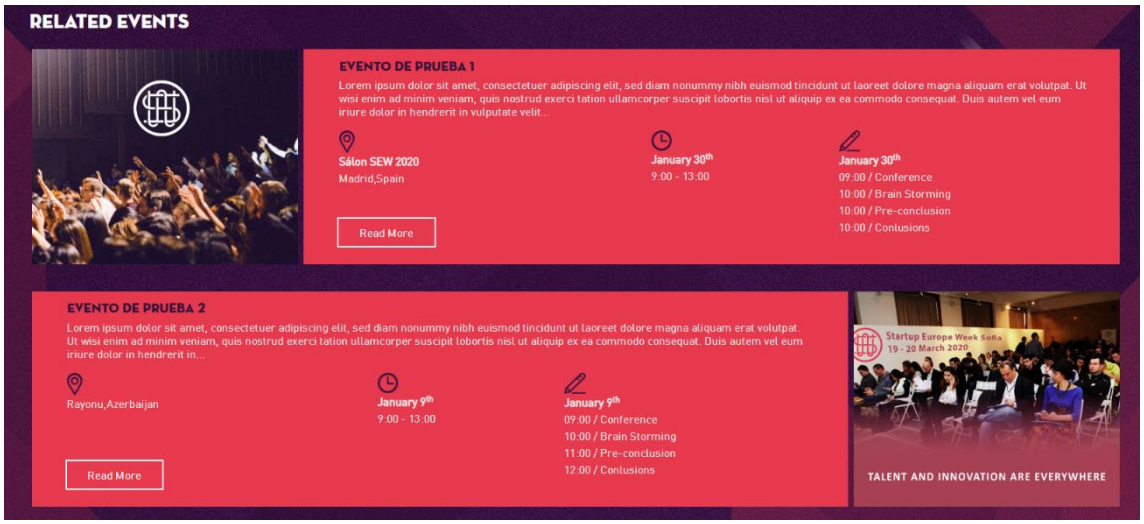


Figura 22 *Eventos relacionados*

Llamada al servidor

- **Método de Solicitud**: GET
- **Ruta**: “/profile/list/related/: id”
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: A partir del id del *co-organizer* busca en la tabla “events” de la base de datos los dos eventos más recientes de este cuyo estado sea “On-Going” o “Completed”, en caso de no tener ningún evento creado se obtienen los dos últimos eventos cuyo estado sea “On-Going” o “Completed” y con esta información *renderiza* el componente “partials/events/related_events”.

3.3.4 Vista de los SEW pasados “Past SEW”

Esta vista contiene un glosario de noticias y artículos de todas las ediciones anteriores y actuales. En principio muestra las tres noticias/artículos más recientes con el mismo diseño que el de las noticias en la vista home, pero se pueden ver más pulsando el botón “*Know More*” que cargará tres noticas más. Además, la vista tiene un buscador por justo debajo del título para filtrar por las noticias/artículos por año.

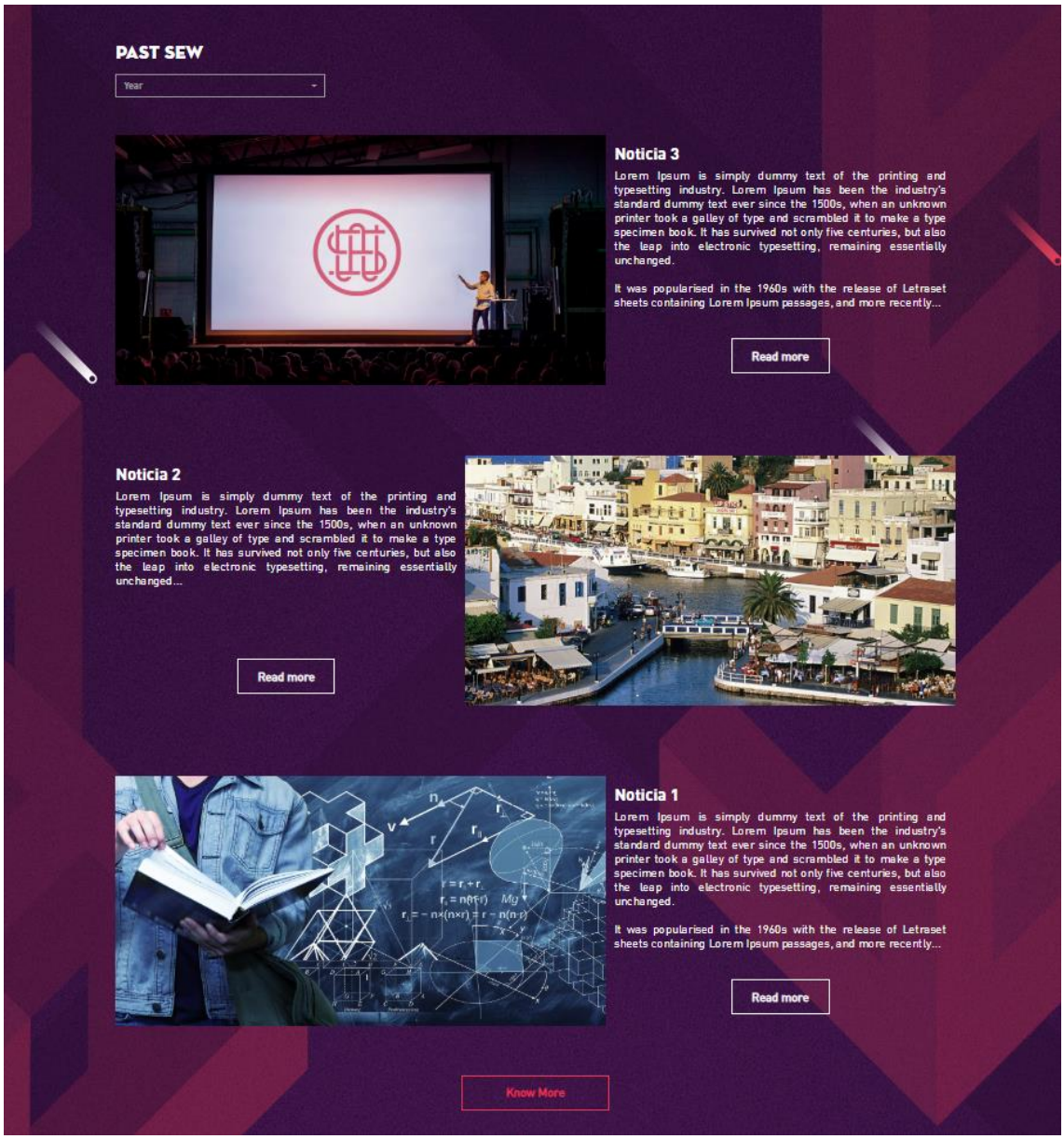


Figura 23 *SEW pasados*

<u>Llamadas al servidor de esta vista</u>

- **Método de Solicitud**: GET
- **Ruta**: "/past_sew"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: *Renderiza* la vista que mostrara las noticias de los SEW pasados.

- **Método de Solicitud**: GET (Revisar)
- **Ruta**: "/past_sew/posts?year=int&nun= int"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: En función de los parámetros que se le pase funcionará de diferente manera. Puede ser usada de tres formas diferentes:
    - **Sin parámetros**: Obtiene los 3 artículos/noticias más recientes cuyo estado es igual a "admin". Con esta información *renderiza* el componente "partials/pastsew".

- **Año (year)**: A partir de un año comprobará en la tabla “posts” de la base de datos las noticias/artículos que se realizaron el año especificado devolviendo tres y con esta información se *renderiza* el componente “partials/pastsew”.

- **Contador (nun):** Obtiene 3 noticias/artículos empezando desde el que se encuentra en la posición “nun” cuyo estado sea igual a ‘admin’ y con esta información se *renderiza* el componente ‘partials/profile/profileList”

3.3.5 Vista de información sobre SEW “About Us”

Esta vista explica qué es Startup Europe Week (SEW), la magnitud de la iniciativa y también explica por qué SEW es útil y como puede ayudar a la gente.

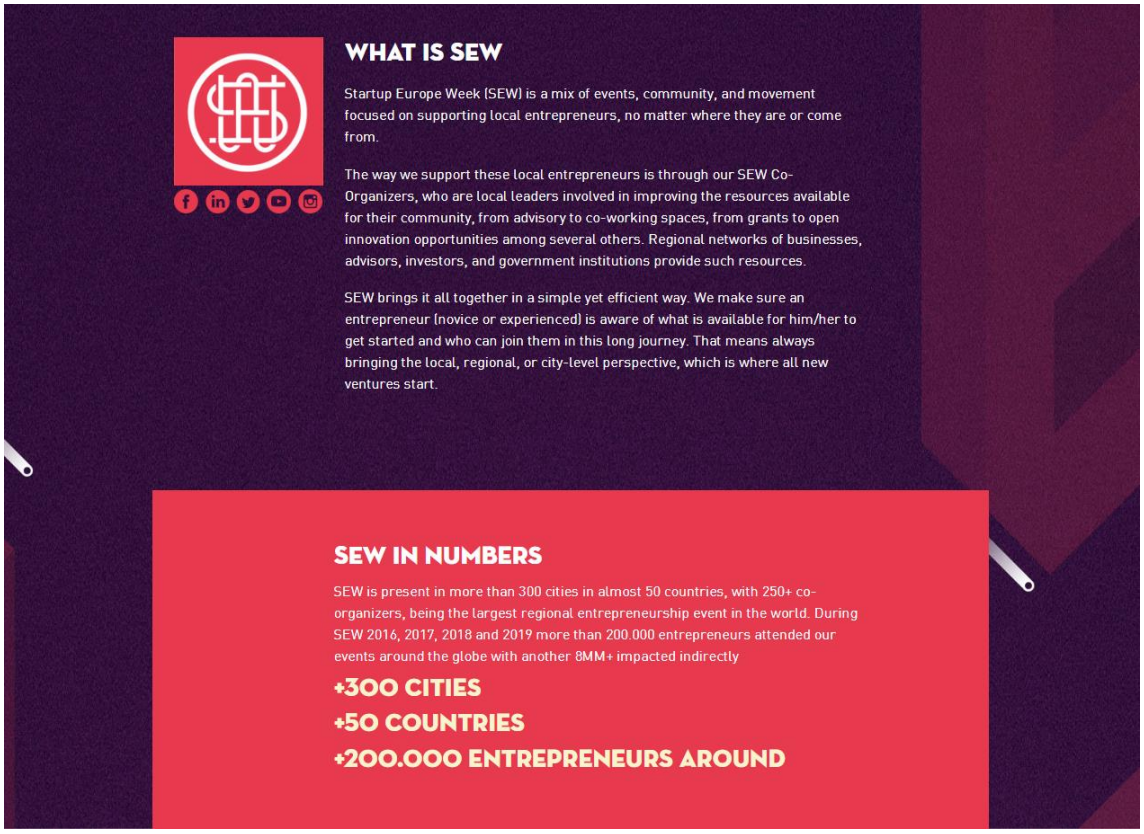


Figura 24 *About Us*

Llamada al servidor de esta vista

- **Método de Solicitud**: GET
- **Ruta**: “/about”
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: *Renderiza* la vista “About Us” que explica qué es y para qué sirve SEW

3.3.6 Vista de los artículos de los Co-organizers “Blog”

La vista blog se encargará de mostrar los tres artículos más recientes. Tanto los *co-organizers* como los administradores pueden crear artículos, pero los *co-organizers* deberán esperar a la validación del administrador por cada artículo que creen. En cuanto al diseño cada artículo se compondrá de 2 partes principales, la imagen de portada del artículo en la parte superior que se podrá convertir en un carrusel de imágenes si el creador del artículo añade a estas fotos, y la parte inferior, que está dividida en dos columnas: el título del artículo se encuentra en la parte superior de la columna izquierda y justo debajo el nombre del autor, el resto de las dos columnas lo ocupa el propio contenido del artículo.

En la parte superior izquierda se encuentra un selector para filtrar los artículos por año y en la parte inferior central se encuentra el botón “*load more*” que sirve para cargar tres artículos más.

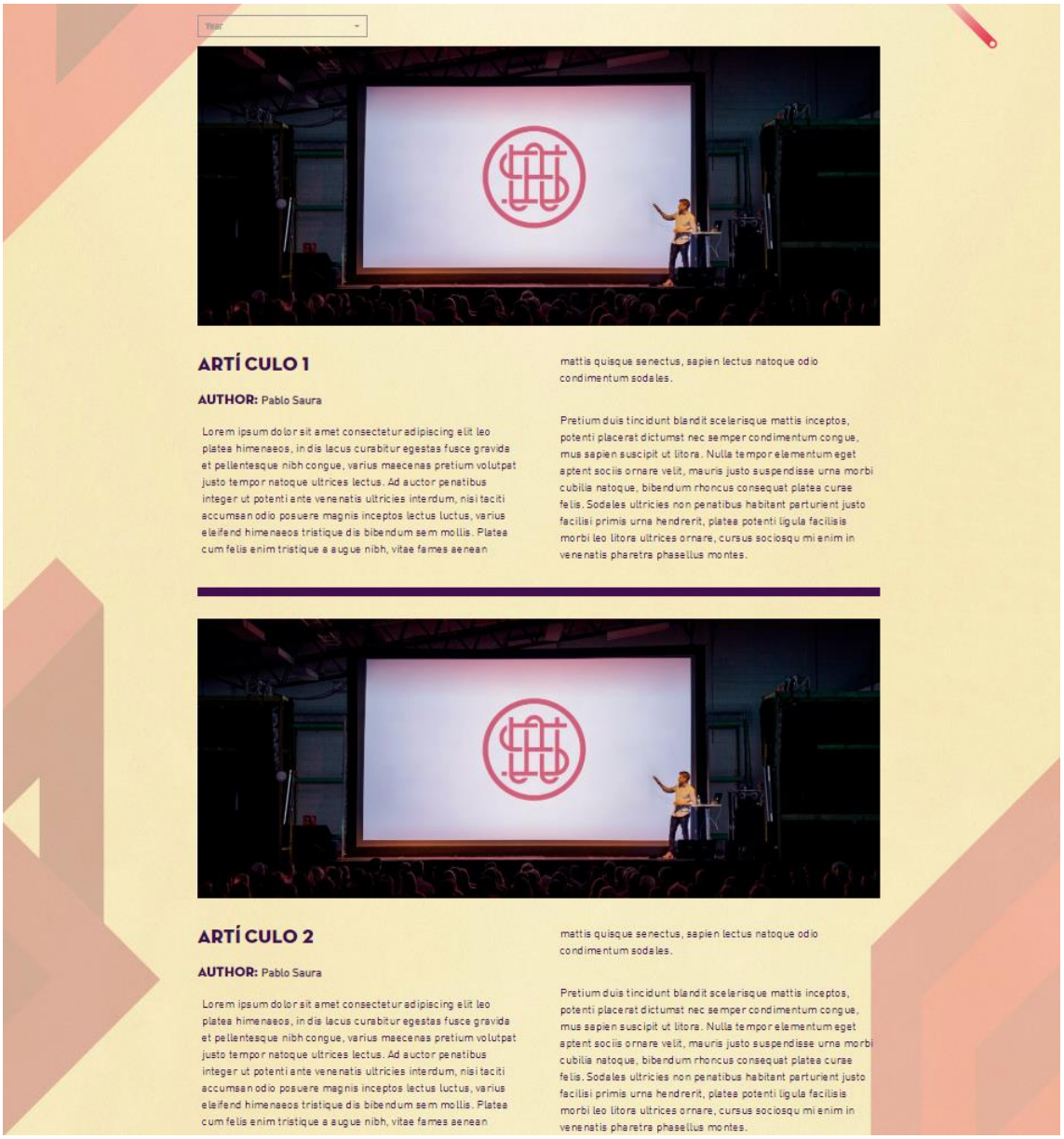


Figura 25 *Blog*

Llamadas al servidor de esta vista

- **Método de Solicitud**: GET
- **Ruta**: "/blog/public/list"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad:** *Renderiza* la vista que muestra una lista de los artículos que puede ver todo el mundo, esta se llama "blog/public_list".

- **Método de Solicitud**: GET
- **Ruta**: "/blog/public/valid_list?nun=int&year=int"
- ***Middleware* de autenticación**: ninguno.
- **Funcionalidad**: Busca en la tabla "post" de la base de datos los 3 artículos más recientes cuyo estado es "valid" en caso de que no se especifiquen las variables "nun" (valor numérico) y "year" (valor numérico), en caso de que la variable "year" se especifique la función buscará los artículos de un año concreto. Si la variable "nun" se especifica, se devolverán 3 artículos de la lista empezando por el artículo que coincida con el "nun" especificado. Con esta información se *renderiza* el componente "partials/blog/public_list".

3.4 Creación y edición de los eventos.

3.4.1 Creación de un evento

Los campos que un Co-organizer debe rellenar para crear un evento son los siguientes (los campos con un asterisco son obligatorios):

1. **Title**: Input de texto para el título del evento. (*)
2. **Event pic**: Input de archivo para la Imagen principal.
3. **Nº attendeers**: Input numérico para el número máximo de asistentes (se puede indicar que el número de asistentes no está limitado clicando en "No limit"). (*)
4. **Stard Date**: Input de texto para la fecha de inicio del evento. Se indicará tanto el día como la hora de inicio. (*)
5. **End Date**: Input de texto para la fecha de fin del evento. Se indicará tanto el día como la hora de inicio. (*)
    - Aclaración de los puntos 4 y 5: si el *co-organizer* indica unas fechas no válidas le saltará un mensaje de error y bloqueará el botón de creación del evento hasta que indique unas fechas válidas.
6. **Country**: Selector del país en el que transcurre el evento. (*)
7. **Region**: Selector de la región en la que transcurre el evento. En función del país seleccionando se rellenará con las regiones de este. (*)
8. **City**: Input de texto para la ciudad en la que ocurre el evento.
9. **Address**: Input de texto para la dirección en la que ocurre el evento.
10. **Desc**: Caja de texto (textbox) para la descripción del evento.

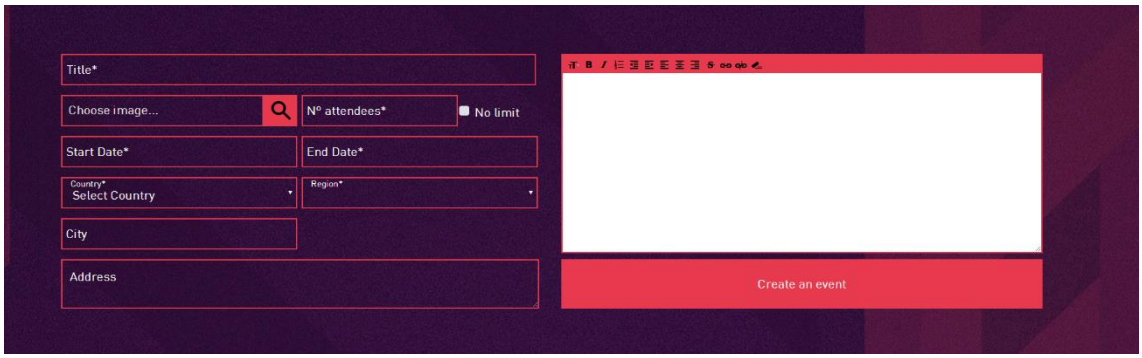


Figura 26 *Creación de un nuevo evento*

Hay una serie de campos que requieren una explicación adicional sobre su funcionamiento, estos son:

- Los campos de inicio y fin de fecha: He usado el componente de JavaScript "bootstrap-datetimepicker" para crear el selector de día y de hora. Una vez obtenidas las fechas genero una lista con los días que dura el evento. Ej: Si un evento empieza el 3 de marzo y acaba el 5 de marzo obtendré los días 3, 4 y 5 de marzo. Esta información será necesaria para la creación del horario que veremos en el aparatado de editar eventos.

- Los campos que indican la localización del evento: Una vez rellenados estos campos haciendo uso de la librería de geolocalización "leaflet-geosearch" obtengo tanto la latitud como la longitud del lugar en el que se realizara el evento. Estos datos son necesarios para poder añadir el evento al mapa de la vista home.

Llamadas al servidor:

- **Método de Solicitud**: GET
- **Ruta**: "/events/add"
- ***Middleware* de autenticación**: isValidUser.
- **Funcionalidad**: *Renderiza* la vista "events/add" cuyo propósito es permitir a un *co-organizer* crear un nuevo evento.

- **Método de Solicitud**: POST
- **Ruta**: "/events/add"
- ***Middleware* de autenticación**: isValidUser.
- **Variables del body**: title, start_date, end_date, country, region,desc, city, address, n_attendees, first_name, last_name,price,days,x,y.
- **Funcionalidad**: A partir de los parámetros necesarios para crear un evento, añade en la tabla 'events' de la base de datos una nueva línea con la información del evento. También añade en la tabla 'date' los días que durara el evento (normalmente es solo uno, pero pueden ser varios)

3.4.2 Edición de un evento

Un *co-organizer* podrá acceder a una vista llamada '*Your events*' donde aparecerán todos los eventos creados por el *co-organizer*. Cada evento dispondrá de un selector para que el *co-*

organizer le cambie el estado, aunque este no es capaz de validarlos (ponerlos en estado 'On-going'), eso lo tendrá que hacer un administrador; el *co-organizer* podrá cancelar el evento y darlo por completado. También cada evento puede ser editado pulsando en el botón 'edit' situado en la esquina inferior derecha de cada evento. Este re-direccionara al *co-organizer* a la vista de edición.

La vista con los eventos de un *co-organizer* es la siguiente.

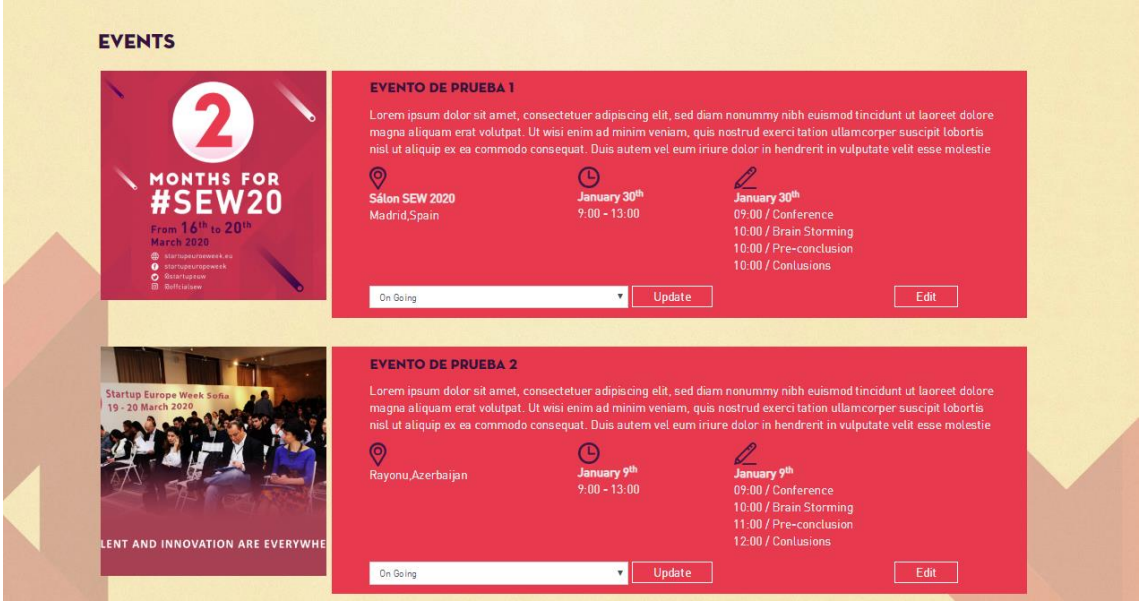


Figura 27 *Lista de eventos de un Co-organizer*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/events/"
- ***Middleware* de autenticación**: isValidUser.
- **Funcionalidad**: Primero busca en la tabla 'events' de la base de datos todos lo eventos cuya columna 'id_user' sea igual al identificador del *co-organizer* que ha iniciado sesión y con esta información *renderiza* la vista "events/list" en la cual un *co-organizer* puede ver una lista con todos los eventos creados por él.

En la vista de edición de un evento, además de poder modificar los campos que añadimos cuando lo creamos también podemos crear el horario para cada uno de los días que durará el evento, añadir fotos al evento, añadir personas que participaran el evento como '*sponsors*', '*speakers*' o moderadores, y también se puede ver una lista con los asistentes del evento. A continuación, explicaré en detalle el funcionamiento de cada una de estas funcionalidades.

Llamada al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/events/edit/:id"

- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: A partir del identificador del evento busca en la tabla y obtiene sus datos, los participantes de este (‘*sponsors*’, ‘*speakers*’ o moderadores) y los asistentes al evento, con esta información *renderizará* la vista “events/edit”.

1. Edición del título, descripción y la imagen de portada

En esta sección el Co-organizer creador del evento podrá actualizar el título del evento, la descripción del evento y su imagen de portada.

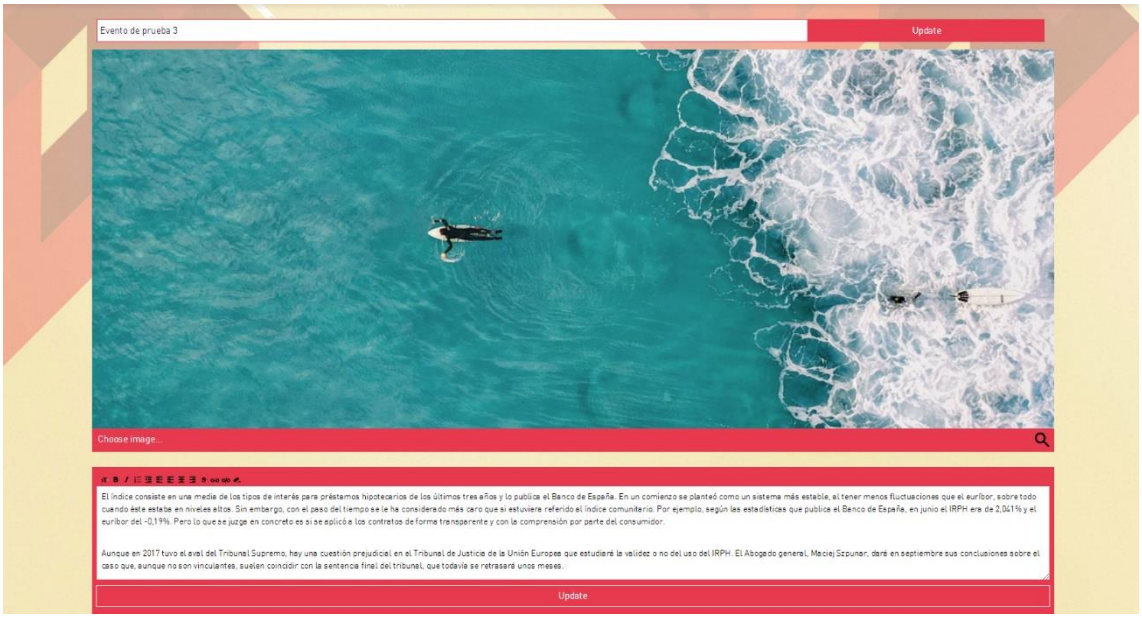


Figura 28 *Edición del evento*

Llamadas al servidor

- **Método de Solicitud**: PUT
- Ruta: “/eventsCo/edit/image/:id”
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: Recibe una nueva imagen de un evento, después comprueba si el evento cuyo identificador coincide con el que se le pasado por la ruta pose ya una imagen en cuyo caso la elimina, después guarda la imagen recibida en la carpeta “events_pic” del servidor con el nombre event_pic + identificador del evento y por último modifica en la tabla “events” de la base de datos con el parámetro de la ruta a la imagen.

- **Método de Solicitud**: PUT
- **Ruta**: “/eventsCo/edit/data/:id”
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Variables del body**: title, desc
- **Funcionalidad**: A partir del identificador de un evento y los parámetros ‘title’ y ‘desc’ actualiza en la tabla ‘events’ el título y la descripción del evento cuyo identificador coincida con el que se el pasado.

2. Edición de la localización del evento

En esta sección el *co-organizer* creador del evento podrá actualizar los datos de la dirección del evento (país, región, ciudad y dirección). Además, cuando actualice estos datos también se actualizarán automáticamente las coordenadas del evento para que coincidan con la nueva localización.

Figura 29 *Edición de la localización*

Llamadas al servidor

- **Método de Solicitud**: PUT
- **Ruta**: "/eventsCo/edit/location/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Variables del body**: country, region, city, address
- **Funcionalidad**: A partir del identificador de un evento y los parámetros 'country ', 'region', 'city' y 'address' actualiza en la tabla 'events' el país, región, ciudad y dirección del evento cuyo identificador coincida con el que se el pasado.

- **Método de Solicitud**: PUT
- **Ruta**: "/eventsCo/coordinates/edit/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Variables del body**: x, y
- **Funcionalidad**: A partir del identificador de un evento y los parámetros 'x' e 'y' actualiza en la tabla 'events' las coordenadas del evento cuyo identificador coincida con el que se el pasado.

3. Actualización de la fecha del evento y creación de horario

En esta sección el *co-organizer* creador del evento podrá actualizar el día o días en cuales transcurrirá el evento modificando los campos de fecha de inicio y fecha situados en la parte superior de la sección, así como la hora a la que empieza y la hora a la que acabará el evento. Además, en la parte inferior se encuentra un listo con una hora, nombre y botón de acción por cada día que dure el evento esto sirve para crear un horario personalizado con el nombre y la hora a la que ocurrirán cada una de las actividades de dicho evento en el día seleccionado. Una vez creada una actividad se podrá eliminar pulsando el botón situado a su derecha.

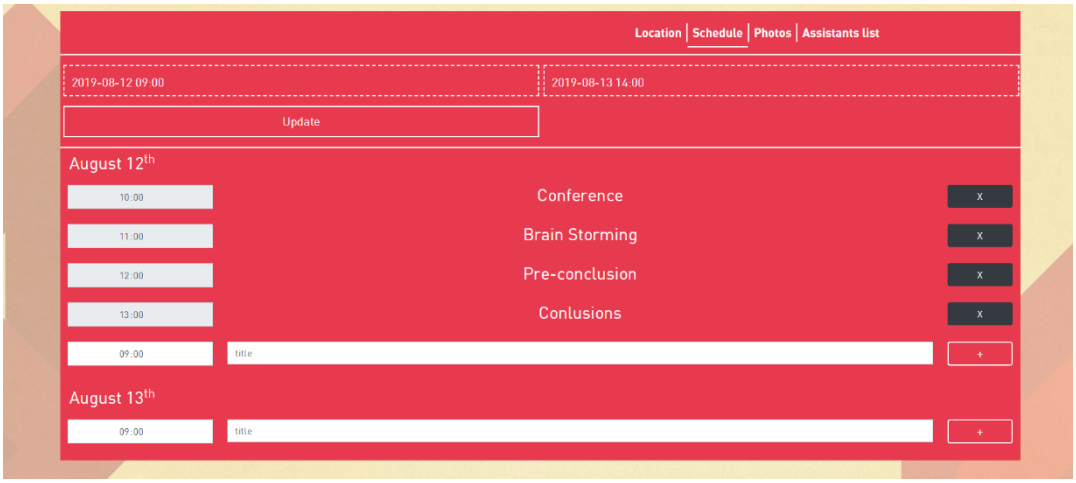


Figura 30 *Edición del horario*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "eventsCo/editSchedule?event_id=Int"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: A partir del identificador del evento "event_id" busca en la tabla "date" de la base de datos los días que dura el evento y también busca en la tabla "ítem" el horario de cada día. Con esta información *renderiza* el componente "partials/events/edit_schedule".

- **Método de Solicitud**: POST
- **Ruta**: "eventsCo/add-item"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Variables del body**: time, name, date_id
- **Funcionalidad**: A partir del nombre de una actividad, la hora que se llevara cabo y el identificador de un día asociado a un evento en concreto, añade en la tabla 'items' una nueva línea con la actividad.

- **Método de Solicitud**: DELETE
- **Ruta**: "eventsCo/delete-item/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: A partir del identificador de una actividad concreta borra esta de la tabla 'item' de la base de datos.

4. Añadir fotos

En esta sección el *co-organizer* creador del evento podrá añadir fotos al evento, que podrá ver justo debajo del input "*Add Photo*". Una vez añadida una foto también podrá ser eliminada pulsando en el icono de la papelera situado en la esquina inferior derecha de cada foto.

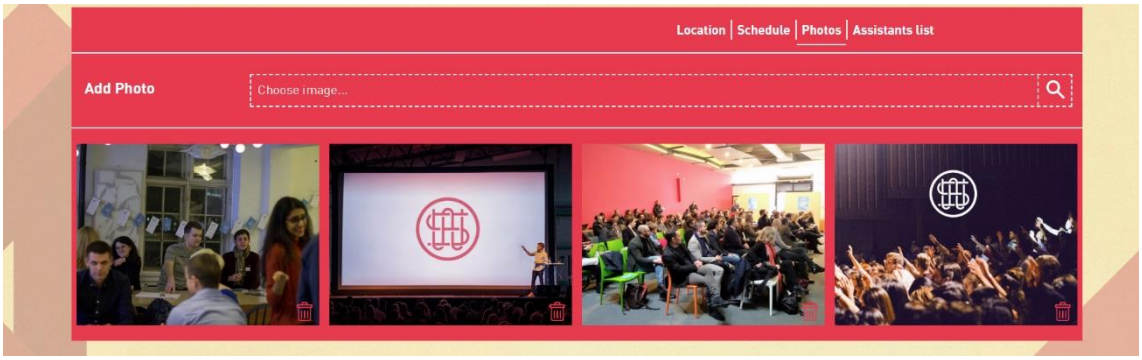


Figura 31 *Añadir fotos al evento*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/eventsCo/photo/edit/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: A partir del identificador de un evento busca en la tabla 'photos' todas las fotos asociadas a dicho evento y con esta información *renderiza* el componente "partials/photo/photo_edit "

- **Método de Solicitud**: POST
- **Ruta**: "/eventsCo/photo/add/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: Añade una foto a un evento concreto. Primero guarda la foto en la carpeta 'photos' y después añade a la tabla "photos" de la base de datos una nueva línea con la ruta de la foto, el identificador del evento al que pertenece y el país en el que se realiza el evento.

- **Método de Solicitud**: DELETE
- **Ruta**: "/eventsCo/ photo/delete/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: Elimina una foto a partir de su identificador.

5. Lista de asistentes

Esta sección permite al *co-organizer* creador ver una tabla con los asistentes al evento. Además, se puede descargar la lista de asistentes en varios formatos (JSON, CSV, XML, TXT, SQL, MS-Excel) a partir del botón situado arriba a la derecha.

Figura 32 *Lista de asistentes*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/events/assistants/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: A partir del identificador de un evento busca en la tabla 'assistants' todos los asistentes cuya columna 'id_events' coincida con el identificador pasado y devuelve una lista de todos los asistentes al evento.

6. Añadir participantes

Un *co-organizer* puede añadir tanto a personas como a entidades que participarán en su evento. Estas pueden ser de tres tipos de roles diferentes:

- **Sponsors**: Serán los que patrocinen el evento.
- **Speakers**: Serán las personas que participarán en el evento haciendo exposiciones, dando charlas o ponencias.
- **Moderador**: Será el encargado de moderar el evento.

A la hora de añadir un *co-organizer* se puede hacer de dos formas. Si también es un *co-organizer* se puede buscar entre una lista de *co-organizers* validados y los datos personales del participante (nombre, ocupación, bio e imagen de perfil) se obtienen automáticamente del perfil del *co-organizer*. Si el participante no es un *co-organizer,* se pueden añadir sus datos manualmente.

Las tres secciones para añadir participantes son parecidas, la tres tienen un botón para añadir un nuevo participante y una vez añadido debajo de la imagen de perfil existe un botón 'Remove' para eliminarlo.

Sección para añadir Sponsor (a diferencia de los *speakers* y moderadores, a estos no se le añade la ocupación ni la bio, solo la imagen y el nombre.

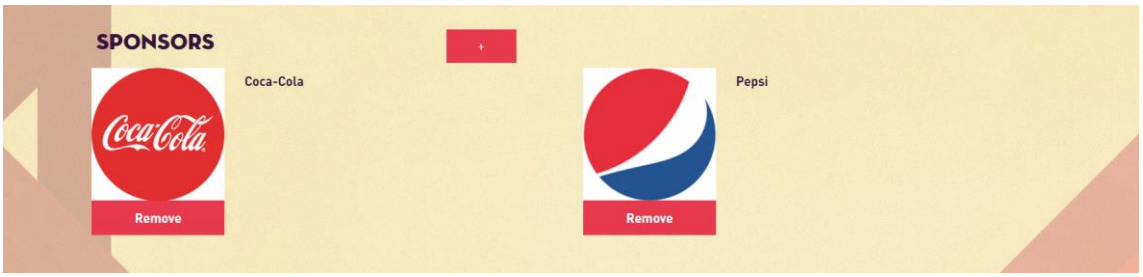


Figura 33 *Lista de sponsor de un evento*

Sección para añadir Speaker

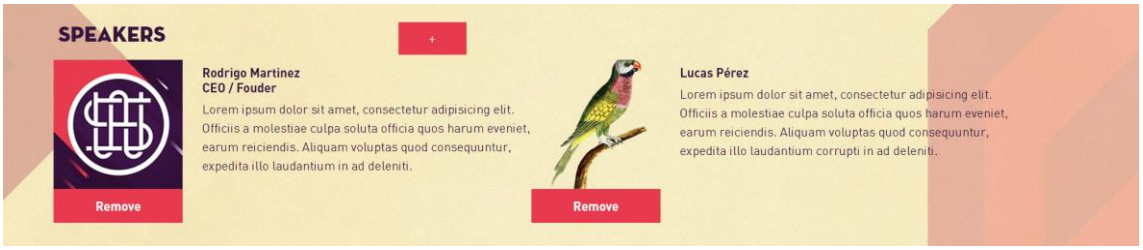


Figura 34 *Lista de speakers de un evento*

Sección para añadir Moderador

Figura 35 *Lista de moderadores de un evento*

Llamadas al servidor

- **Método de Solicitud**: POST
- **Ruta**: "/eventsCo/add-new-participant"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Variables del body**: first_name,last_name, event_id,rol,occupation,bio
- **Funcionalidad**: A partir de los parámetros necesarios para crear un nuevo participante (Primer nombre, apellido, ocupación, bio y rol) añadimos en la tabla 'participants' una nueva línea asociando el participante con su evento añadiendo el identificador del evento a la de la columna 'event_id'.

- **Método de Solicitud**: POST
- **Ruta**: "/eventsCo/add-participant"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Variables del body**: user_id, rol, event_id
- **Funcionalidad**: Es parecida a la anterior pero el nuevo participante será un *co-organizer* que a sí que lo único que necesitamos añadir a la tabla 'participants' es el identificador del *co-organizer*, su rol y el identificador del evento.

- **Método de Solicitud**: DELETE
- **Ruta**: "/eventsCo/delete-participant /:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.
- **Funcionalidad**: Elimina un participante a partir de su identificador.

- **Método de Solicitud**: POST
- **Ruta**: "/eventsCo/add/participantImage"
- ***Middleware* de autenticación**: isValidUser, isOwnerUser.

- **Variables del body**: id, name
- **Funcionalidad**: Recibe una imagen de un participante, después guarda la imagen recibida en la carpeta "participan_pic" del servidor con el nombre "participan_pic + identificador del evento + nombre de la imagen" y por último modifica en la tabla 'participants' de la base de datos con el parámetro de la ruta a la imagen.

3.4.3 Estructura de la base de datos de los eventos

Tabla de eventos 'events'

Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador del evento
user_id	INT	11	Identificador del usuario creador del evento
title	TEXT		Título del evento
image	TEXT		Ruta a la Imagen principal
video	TEXT		URL del video (actualmente no se usa)
desc	TEXT		Descripción del evento
type	ENUM	'Free','Paid'	Tipo del evento (actualmente no se usa)
price	FLOAT		Precio del evento (actualmente no se usa)
status	ENUM	'Completed','On-Going','Pending','Candelled by me','Cancelled by admin'	Estado del evento
start_date	DATETIME		Fecha de inicio del evento
end_date	DATETIME		Fecha de fin del evento
country	VARCHAR	50	País en el que ocurre el evento
region	VARCHAR	50	Región en la que ocurre el evento
city	VARCHAR	50	Ciudad en el que ocurre el evento
address	TEXT		Dirección en la que ocurre el evento
n_attendees	INT	4	Número de personas que asistirán al evento
max_n_attendees	INT	4	Máximo nuero de personas que pueden asistir al evento

full_name	TEXT		Nombre completo del creador del evento
x	VARCHAR	10	Latitud
y	VARCHAR	10	Longitud
create_at	DATETIME		Fecha de creación del evento

Cada evento aparte de tener su propia tabla en la base de datos está relacionado mediante claves foráneas a 5 tablas más; estas son:

1. Asistentes a los eventos '*assistants*': contiene los datos de todos los asistentes de los eventos, esta tabla estará relacionada con la de los eventos gracias la clave foránea (Id 'events' -> id_events 'assistants')

2. Participantes a los eventos '*participants*': contiene tanto el rol de que estos ejercerán en el evento como una serie de datos personales, esta tabla estará relacionada con la de los eventos gracias la clave foránea (Id 'events' -> events_id 'participants')

3. Fotos de los eventos '*photos*': contiene la URL de todas las fotos que han sido añadidas a un evento, esta tabla estará relacionada con la de los eventos gracias la clave foránea (Id 'events' ->id_events 'photos')

4. Días de los eventos '*date*': contiene el día o días en los que transcurre un evento, esta tabla estará relacionada con la de los eventos gracias la clave foránea (Id 'events' ->id_events 'date')

   - A su vez la tabla 'date' está relacionada con la tabla 'time'

5. *Co-organizer* creador del evento 'user': Contiene toda la información personal del creador del evento, esta tabla estará relacionada con la de los eventos gracias la clave foránea (user_id 'events' ->id 'user')

Tabla de participantes asociados a un evento 'participants'

Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador del participante
event_id	INT	11	Identificador del evento asociado al participante
user_id	INT	11	Si el participante es un Co-organizer se usará este identificador para obtener los datos personales del participante
rol	ENUM	'sponsor','speaker','moderator'	Rol del participante

first_name	VARCHAR	30	Primer nombre del participante
last_name	VARCHAR	30	Primer apellido del participante
image	TEXT		Ruta a la imagen que aparecerá en el perfil del participante
bio	TEXT		Bio del participante
occupation	VARCHAR	50	Ocupación del participante

Tabla de participantes asociados a un evento 'assistants'

Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador asistente
event_id	INT	11	Identificador del evento asociado al asistente
first_name	VARCHAR	30	Primer nombre del asistente
last_name	VARCHAR	30	Primer apellido del asistente
email	TEXT		Correo del asistente
occupation	TEXT		Ocupación del asistente
age	INT	11	Edad del asistente
country	VARCHAR	50	País del asistente

Tabla de días en los que transcurre un evento 'date'

Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador del día
event_id	INT	11	Identificador del evento asociado al día
day	DATE	30	Fecha de uno de los días en los que transcurre el evento

Tabla con el horario de cada día de un evento 'time'

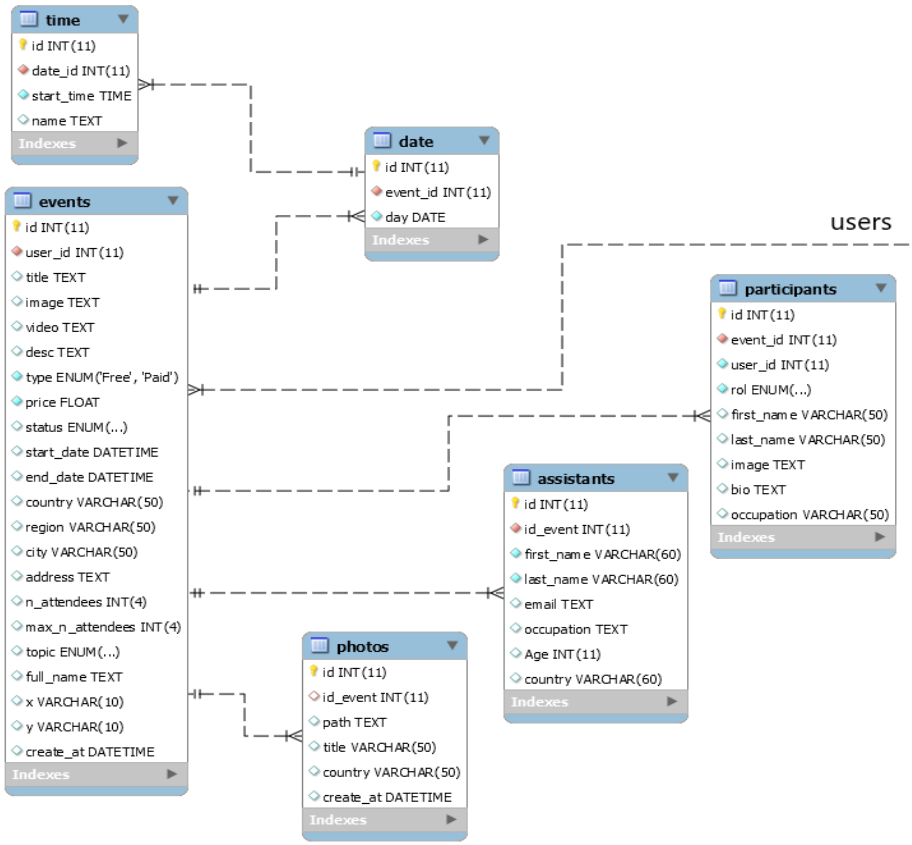
Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador del
date_id	INT	11	Identificador del día asociado al
start_time	DATE		Hora a la que ocurre
name	TEXT		Descripción de

Tabla de las fotos asociadas a los eventos 'photos'

Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador de la foto

event_id	INT	11	Identificador del evento asociado a la foto
image	TEXT		Ruta a la foto
title	VARCHAR	50	Título del evento asociado a la foto
country	VARCHAR	50	País del evento asociado a la foto
create_at	DATETIME		Fecha en la que se añadió la foto al evento

A continuación, podemos ver un diagrama con las tablas que se usan en este apartado.



3.5 Registro e inicio de sesión de un co-organizer

3.5.1 Registro de un co-organizer

Los campos que un *co-organizer* debe rellenar para registrarse son los siguientes (los campos con un asterisco son obligatorios):

1. **Email**: Input de texto para el correo del *co-organizer*. (*)
2. **Password**: Input de tipo 'password' para la Contraseña del *co-organizer* (como mínimo tiene que poseer 8 caracteres). (*)

3. **Confirm password**: Input de tipo 'password' para la repetición de la contraseña para asegurarse de que no se ha equivocado al escribirla. (*)
4. **Profile pic**: foto del *co-organizer* que usara como imagen de perfil.
5. **First name**: Input de texto para el nombre del usuario. (*)
6. **Last name**: Input de texto para el apellido o apellidos del *co-organizer*. (*)
7. **Age**: Input numérico para la edad del *co-organizer*.
8. **Country**: Selector del país del *co-organizer*. (*)
9. **Region**: Selector de la región del *co-organizer*. En función del país seleccionando se rellenará con las regiones de este. (*)
10. **City**: Input de texto para la ciudad del *co-organizer*.
11. **Address**: Input de texto para la dirección del *co-organizer*.
12. **Bio**: Caja de texto (textbox) para añadir una breve descripción del *co-organizer*.
13. **LinkedIn:** Input de texto para la URL del LinkedIn del *co-organizer*.
14. **Facebook:** Input de texto para la URL del Facebook del *co-organizer*.
15. **Twitter:** Input de texto para la URL del Twitter del *co-organizer*.
16. **Occupation:** Input de texto para la ocupación del *co-organizer*.
17. **Instagram:** Input de texto para la URL del Instagram del *co-organizer*.

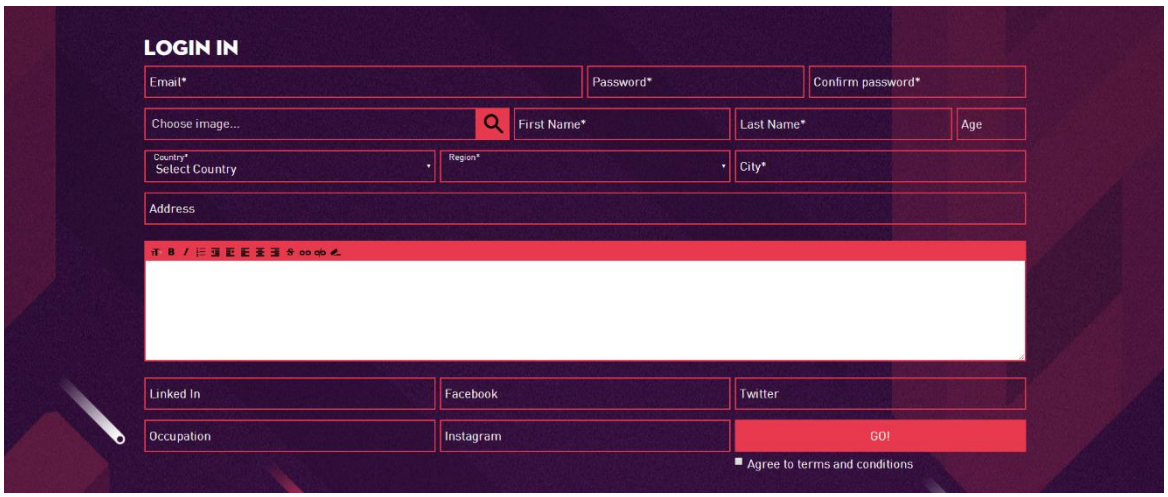


Figura 36 *Formulario de registro*

A parte de rellenar todos estos campos también es necesario que acepte los términos y condiciones para poder registrase. Una vez registrado recibirá un correo de SEW con instrucciones, pero todavía no podrá crear o editar eventos. Tendrá que esperar a que un administrador compruebe sus datos y confirme que es un *co-organizer* válido para aceptarlo.

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/signup"
- ***Middleware* de autenticación**: isNotLoggedIn.
- **Funcionalidad**: Renderiza la página "auth/signup" en la cual un *co-organizer* se registrará introduciendo sus datos personales.

- **Método de Solicitud**: POST
- **Ruta**: "/signup"
- **Variables del body**: password2, first_name, last_name, age, country, region, city, address, facebook, linked, twitter, instagram, occupation, bio

- ***Middleware* de autenticación**: isNotLoggedIn.
- **Funcionalidad**: Se usa para registrar a un *co-organizer* y automáticamente después iniciar sesión. A continuación, explico por pasos su funcionamiento:
    1. Usa la función de autenticación del middleware 'passport' el cual redireccionará al Co-organizer a su perfil en caso de tener éxito o lo redireccionará a la vista de registro de usuario si falla. También mandara un mensaje de error o confirmación.
    2. Comprueba que el correo del nuevo Co-organizer no se encuentra en la tabla 'users', si ya existe cancela el registro de usuario y devuelve un mensaje de error diciendo que el correo introducido ya está siendo usado por un Co-organizer existente en nuestra base de datos.
    3. Comprueba que la contraseña y la contraseña de confirmación son la misma, si no coinciden se cancela el registro de usuario y devuelve otro mensaje de error diciendo que las contraseñas no coinciden.
    4. Si el paso 2 y 3 se realizan correctamente introduciremos una nueva linea en tabla 'users' con los datos del Co-organizer.
        - La contraseña será guardada en texto cifrado.
        - El estado del nuevo Co-organizer será el de 'pending_approval' que siginifica que necesita la aprobación de un administrador.
    5. Se manda un mensaje de agradecimiento por registrarse al Co-organizer y explicándole en que consiste el SEW.
    6. Si todo ha salido correctamente se crea una sesión con los datos del Co-organizer y también se guardará en la tabla 'sessions', por último, el Co-organizer será redireccionado a su perfil.

- **Método de Solicitud**: GET:
- **Ruta**: "/protection_law"
- ***Middleware* de autenticación**: ninguno
- **Funcionalidad**: *Renderiza* una vista con la información de protección de datos personales.

3.5.2 Inicio de sesión de un *co-organizer*

Una vez se ha registrado un *co-organizer*, para volver a iniciar sesión usará la vista 'Log in' en la cual tendrá que introducir su correo y contraseña. Si escribió bien su contraseña y el *co-organizer* existe en la base de datos, iniciará sesión, y en el caso contrario, recibirá un mensaje de error diciendo que su contraseña o correo son erróneos.

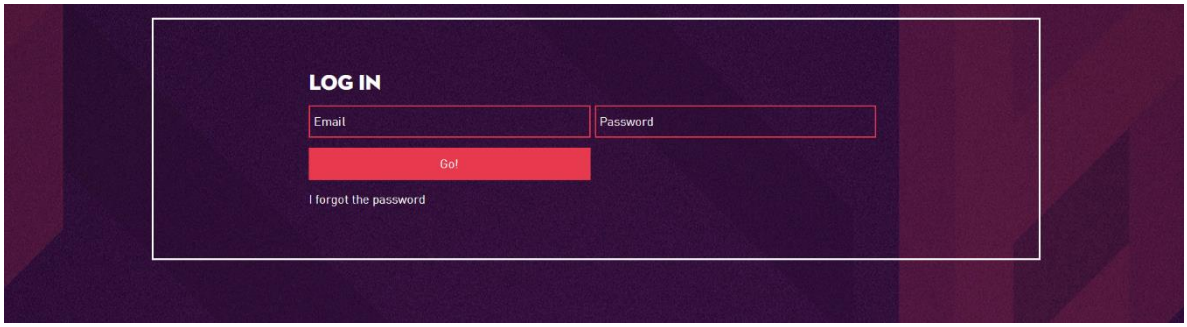


Figura 37 *Formulario de registro*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/signin"
- ***Middleware* de autenticación**: isNotLoggedIn.
- **Funcionalidad**: *Renderiza* la página "auth/signin" en la cual un *co-organizer* iniciará sesión con su correo y contraseña.

- **Método de Solicitud**: POST
- **Ruta**: "/signin"
- ***Middleware* de autenticación**: isNotLoggedIn.
- **Funcionalidad**: A partir de un correo y una contraseña comprueba en la tabla "users" de la base de datos si existe el correo en caso afirmativo comparara la contraseña pasada por el usuario con la que se encuentra en la base de datos (la comparación se realiza en texto cifrado) en caso de coincidir se creara una sesión para el *co-organizer*, se guardara está en la tabla 'sessions' de la base de datos y se redirigirá a la página de perfil del *co-organizer*. En caso de fallar recargara la página de inicio de sesión mostrando un mensaje de error.

- **Método de Solicitud**: GET
- **Ruta**: "/logout"
- ***Middleware* de autenticación**: isLoggedIn.
- **Funcionalidad**: Sirve para cerrar sesión y redirige al *co-organizer* a la página de iniciar sesión.

3.5.3 Edición de datos personales

En el perfil del *co-organizer* este puede modificar todos sus datos personales, menos la contraseña, en la sección llamada 'Personal Information'.

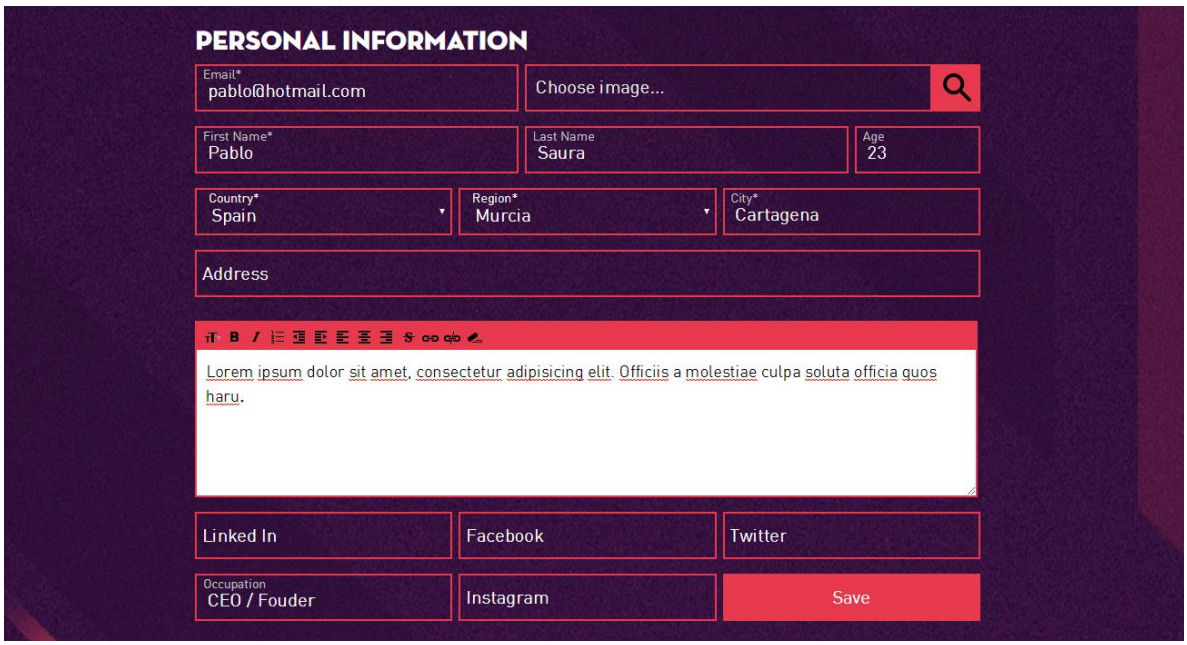


Figura 38 *Edición de datos personales*

En la sección llamada ‘Change your Password’, un *co-organizer* puede actualizar su contraseña. Para hacerlo primero tendrá que introducir su antigua contraseña y después la nueva dos veces si se realiza todo correcta mente la contraseña será actualizada.

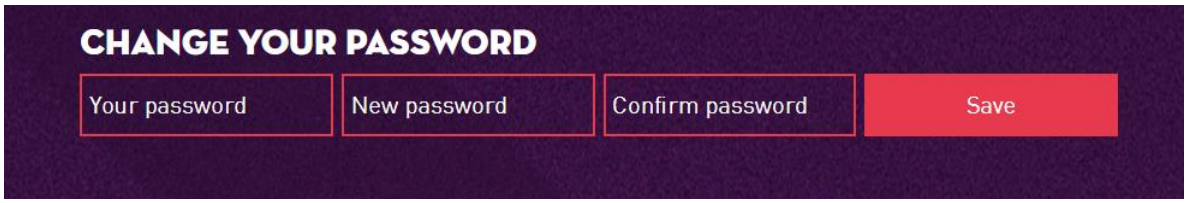


Figura 39 *Cambio de contraseña*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: “/profile /data/edit”
- ***Middleware* de autenticación**: isLoggedIn.
- **Funcionalidad**: Obtiene la información personal del *co-organizer* (menos la contraseña y la imagen del usuario) a partir del cache del servidor (previamente a tenido que iniciar sesión) y con esta información *renderiza* el componente “partials/profile/edit”.

- **Método de Solicitud**: PUT
- **Ruta**: “/profile /data/edit”
- ***Middleware* de autenticación**: isLoggedIn.
- **Variables del body**: email, first_name, last_name, age, country, region, city, address,bio,linked,facebook,twitter,occupation,instagram.
- **Funcionalidad**: A partir de los nuevos datos del *co-organizer* y su identificador obtenido a partir del cache del servidor (Cuando inicias sesión los datos de un *co-organizer* se guardan temporalmente en el servidor) actualizar en la tabla ‘user’ los datos del *co-organizer* (menos la contraseña y la imagen del usuario).

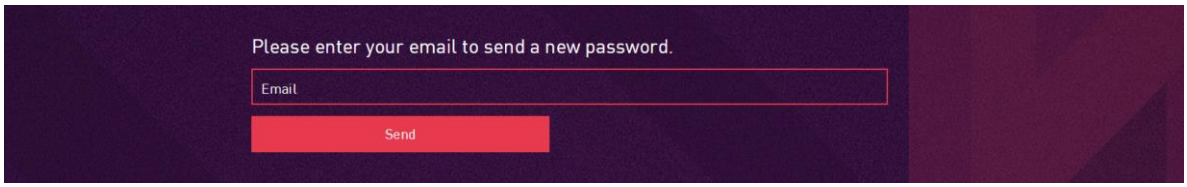
- **Método de Solicitud**: POST
- **Ruta**: “/profile /edit/image/”
- ***Middleware* de autenticación**: isLoggedIn.
- **Funcionalidad**: Recibe una nueva imagen del usuario, después comprueba si el usuario pose imagen de perfil en cuyo caso la elimina, guarda la imagen en la carpeta “user_pic” del servidor con el nombre 'userpic' + id de usuario y por último modifica en la tabla “users” de la base de datos el parámetro de la ruta a la imagen.

- **Método de Solicitud**: POST
- **Ruta**: “/profile /edit/pass/”
- ***Middleware* de autenticación**: isLoggedIn.
- **Funcionalidad**: Recibe la antigua contraseña, la nueva contraseña y la repetición de la nueva contraseña. Primero comprueba que la antigua contraseña coincide con la que se encuentra con la base de datos, en caso afirmativo comprueba que la nueva contraseña y su repetición son iguales en caso afirmativo se modifica la contraseña del usuario.

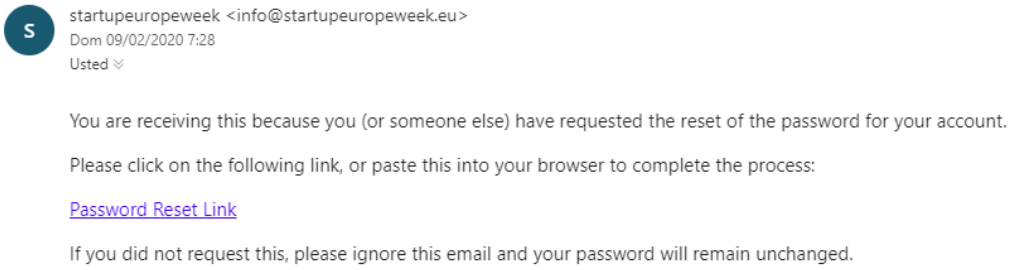
3.5.4 Recuperación de contraseña

En el caso de que un *co-organizer* olvidará su contraseña, podrá recuperarla usando el enlace llamado 'I forgot the password' que se encuentra justo debajo de los campos para iniciar sesión. Una vez pulsado será re-direccionado una vista para que introduzca su correo de *co-organizer*.

La vista para mandar el correo para cambiar la contraseña es la siguiente.

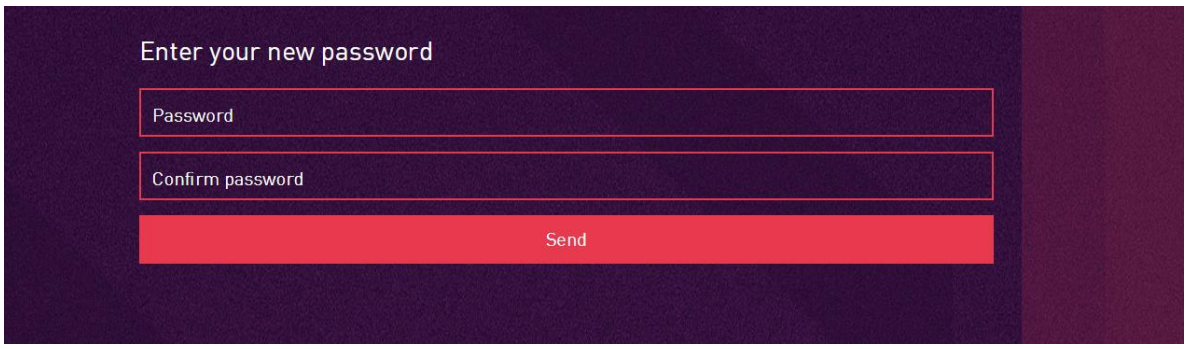


Después comprueba que el correo introducido en la función anterior existe en la base de datos, en caso afirmativo manda un correo al *co-organizer* con un enlace para cambiar su contraseña. El correo con el enlace para cambiar la contraseña es el siguiente.



Usando el enlace 'Password Reset link' el *co-organizer* será re-direccionado a una vista para cambiar la contraseña, en que le pedirán escribir la nueva contraseña dos veces, si coinciden la contraseña y no ha pasado más de una hora desde que se envió el correo de reseteo, entonces la contraseña será actualizada.

La vista para actualizar la contraseña es la siguiente.



Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/forgot"
- ***Middleware* de autenticación**: isNotLoggedIn.
- **Funcionalidad**: *Renderiza* la página "auth/forgotPass" en la cual un *co-organizer* puede introducir su correo en caso de no recordar su contraseña.

- **Método de Solicitud**: GET
- **Ruta**: "/email"
- ***Middleware* de autenticación**: isNotLoggedIn.
- **Variables del body**: email.
- **Funcionalidad**: A partir de la variable email comprueba en la tabla 'users' si exite algún *co-organizer* con dicho email, si existe devuelve una variable booleana con valor 'True' y si no existe el valor es 'False'

- **Método de Solicitud**: POST
- **Ruta**: "/forgot"
- ***Middleware* de autenticación**: isNotLoggedIn.
- **Variables del body**: email.
- **Funcionalidad**: Enviara un correo al *co-organizer* para que recupere su contraseña, este correo contendrá una URL con un token (código aleatorio único) que llevara al *co-organizer* a la página para cambiar su contraseña y además guardara el token generado y el correo en el la tabla "tokens" de la base de datos con un tiempo de vida de una hora.

- **Método de Solicitud**: GET
- **Ruta**: "/reset/: token"
- ***Middleware* de autenticación**: isNotLoggedIn.
- **Funcionalidad**: Renderiza la página "auth/resetPass" en que un *co-organizer* puede introducir su nueva contraseña.

- **Método de Solicitud**: POST
- **Ruta**: "/reset/"
- ***Middleware* de autenticación**: isNotLoggedIn.
- **Variables del body**: token, Pass.
- **Funcionalidad**: Primero comprueba que el token que le hemos pasado coincide con el que existe en la base de datos en caso afirmativo actualizaremos la contraseña (en texto cifrado) de la base de datos correspondiente al correo que tiene asociado el token con la que le hemos pasado (variable Pass).

3.5.5 Estructura de la base de datos de los *co-organizers* (users)

Tabla de *co-organizers* 'users'

Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador del Co-organizer
email	VARCHAR	60	Correo del Co-organizer
password	VARCHAR	60	Contraseña del Co-organizer guardada en texto cifrado
status	ENUM	'pending_approval', 'valid','rejected', 'block','admin'	Estado del Co-organizer
profile_pic	TEXT		Ruta a la imagen de perfil del Co-organizer
first_name	VARCHAR	30	Primer nombre del Co-organizer
last_name	VARCHAR	30	Primer apellido del Co-organizer
age	INT		Edad del Co-organizer
country	VARCHAR		País del Co-organizer
region	VARCHAR	50	Región del Co-organizer
city	VARCHAR	50	Ciudad del Co-organizer
address	VARCHAR	60	Dirección del Co-organizer
skills	TEXT		Habilidades del del Co-organizer (actualmente no se usa)
facebook	TEXT		URL de Facebook del Co-organizer
linked	TEXT		URL de LinkedIn del Co-organizer
twitter	TEXT		URL de Twitter del Co-organizer
instagram	TEXT		URL de Instagram del Co-organizer
occupation	TEXT		Ocupación del Co-organizer
bio	TEXT		Bio del Co-organizer

Tablas relacionadas con la de *co-organizers*

Cada evento aparte de tener su propia tabla en la base de datos está relacionado mediante claves foráneas a 2 tablas más estas son:

1. Eventos creados 'events': contiene los datos básicos de un evento, esta tabla estará relacionada con la de los usuarios gracias la clave foránea.
   (Id 'user' -> user_id 'events')

2. Artículos/noticias creadas ‘post’: contiene los datos básicos de los artículos/noticias creados por los *co-organizer* /Administradores, esta tabla estará relacionada con la de los usuarios gracias la clave foránea.
   (Id ‘user’ -> user_id ‘posts’)

A parte de las tablas de eventos y Artículos/Noticias existen dos tablas relacionadas con la de los *co-organizer*:

- **Tokens**: Se encara de guardar un token con un email y se usara cuando un Co-organizer necesite recuperar su contraseña.
- **Sessions**: Se encarga de guardar las sesiones cunado un Co-organizer inicia sesión.

Tabla de sesiones ‘sessions’

Nombre	Tipo de datos	Longitud/conjunto	Descripción
session_id	VARCHAR	128	Identificador de la sesión
expires	INT	11	Tiempo de expiración
data	TEXT		Información de la sesión

Tabla de tokens ‘tokens’

Nombre	Tipo de datos	Longitud/conjunto	Descripción
token	VARCHAR	256	Token necesario para la autenticación del enlace de reseteo de contraseña
email	TEXT		Correo del Co-organizer que quiere resetear su contraseña
timestamp	DATETIME		Fecha que en la que fue creado el token

3.6 Creación y edición de los Artículos/noticias

Un *co-organizer* dispone de la vista ‘Post’ que tiene dos secciones una llamada ‘Add Post’ que sirve para crear un nuevo Articulo y otra llamada ‘Your Post’ en la cual aparecerán todos los Artículos creados por el *co-organizer*.

Llamada al servidor

- **Método de Solicitud**: GET
- **Ruta**: “/blog”
- ***Middleware* de autenticación**: isValidUser.
- **Funcionalidad**: *Renderiza* la vista que muestra los artículos del *co-organizer* y permite crear uno nuevo, esta se llama “blog/add_and_list”.

3.6.1 Crear un nuevo artículo

En la sección 'Add post' el *co-organizer* puede crear un nuevo Artículo simplemente añadiendo un título en el input 'title', añadiendo su descripción en el 'textbox' que está justo debajo y pulsando el botón 'Save'. También se puede borrar el contenido del artículo pulsando el botón 'Delete'. Una vez creado el artículo el *co-organizer* tendrá que esperar a la validación algún administrador para que este se muestre en la vista 'Blog' de la web.

Figura 40 *Crear un nuevo artículo*

Llamada a al servidor

- **Método de Solicitud**: POST
- **Ruta**: "/blog/add"
- ***Middleware* de autenticación**: isValidUser.
- **Variables del body**: title, desc
- **Funcionalidad**: A partir de los parámetros necesarios para crear un artículo (Titulo y descripción) y el identificador del *co-organizer* crea en la tabla 'post' de la base de datos un nuevo artículo asociado al *co-organizer* creador que estará pendiente de la validación de algún administrador. Por último, dirige al *co-organizer* a la página de edición del artículo.

3.6.2 Lista de artículos creados

En la sección 'Your post' aparecerán los artículos creados por el *co-organizer*, mostrando el título y la descripción de estos. Cada artículo tiene dos botones en la esquina inferior derecha uno llamado 'edit' que sirve para re-direccionar al *co-organizer* a la vista de edición del artículo y otro botón llamado 'Delete' para eliminar el artículo.

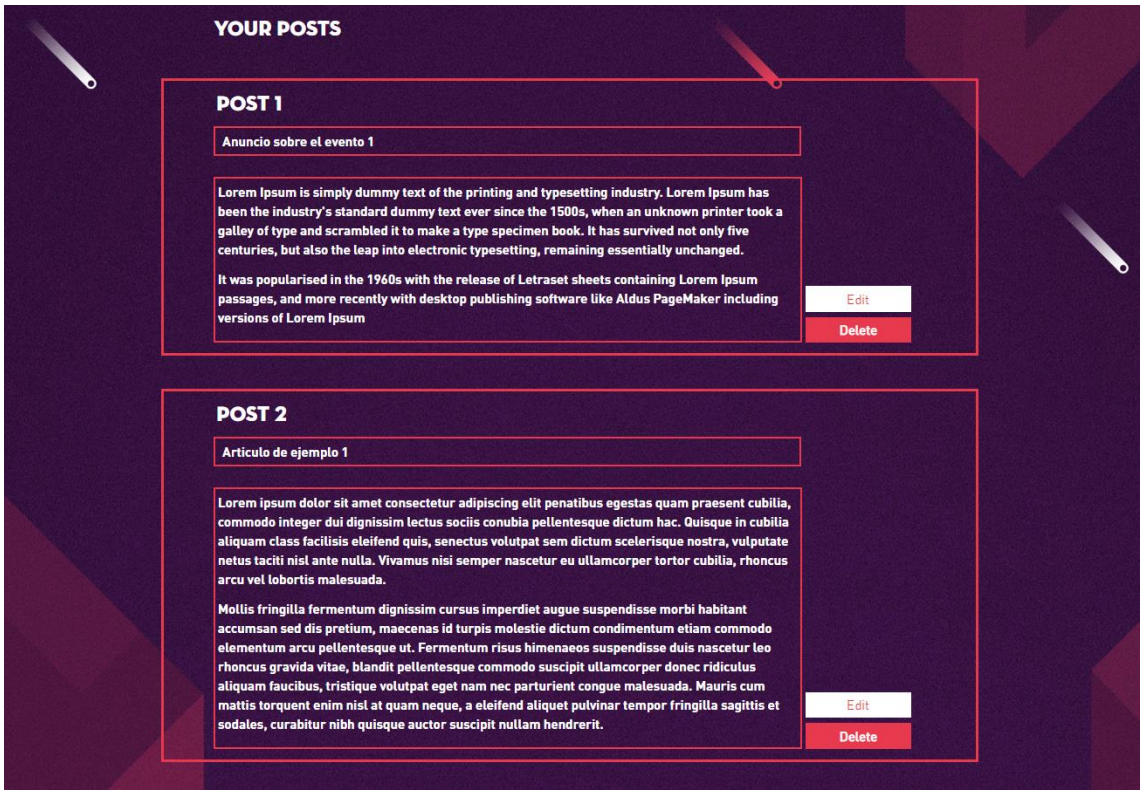


Figura 41 *Lista de artículos de un Co-organizer*

Llamada a al servidor

- **Método de Solicitud**: GET
- **Ruta**: "blog/add/list"
- ***Middleware* de autenticación**: isValidUser.
- **Funcionalidad**: A partir del cache del servidor se obtiene el id del *co-organizer* que previamente ha tenido que iniciar sesión. Usa el id para buscar en la tabla "posts" de la base de datos todos los artículos que ha creado el *co-organizer* y con esta información *renderizamos* el componente "partials/blog/list".

- **Método de Solicitud**: DELETE
- **Ruta**: "/blog/delete/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUserblog
- **Funcionalidad**: A partir del id de un artículo lo busca en la tabla 'post' de la base de datos borra el articulo obtenido de la base de datos.

3.6.3 Edición de un artículo

En la vista de edición de un artículo además de poder modificar los campos que añadimos cuando lo creamos, también podemos añadirle una imagen de portada. A igual que en la sección de 'Yours post' aquí también disponemos del botón 'Delete' para eliminar un artículo.

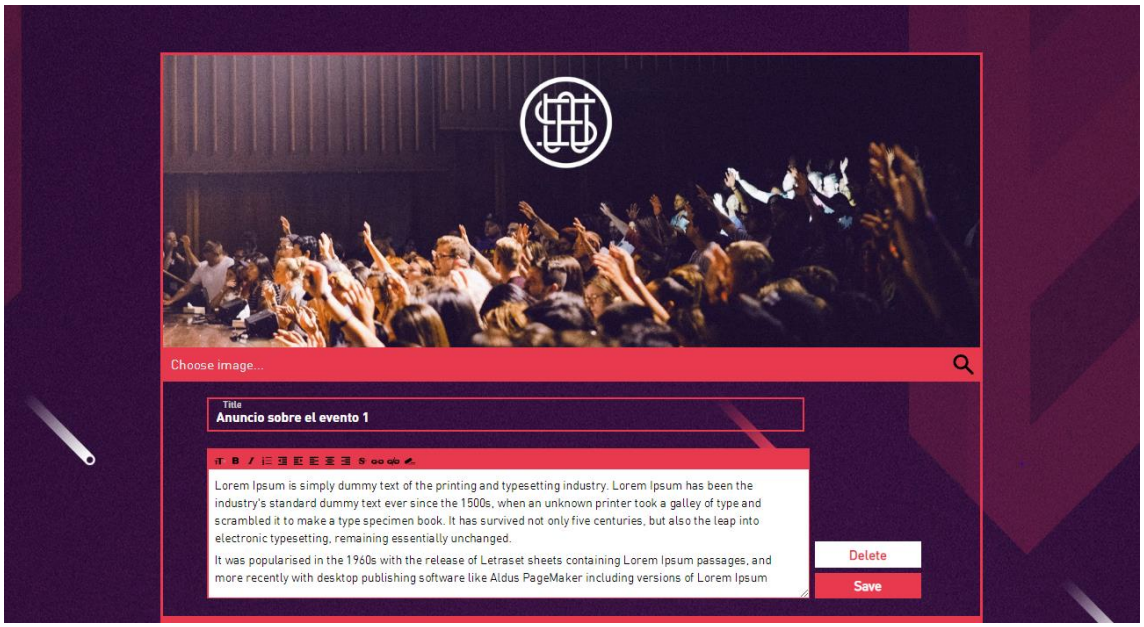


Figura 42 *Edición de un artículo*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/blog/edit/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUserblog.
- **Funcionalidad**: A partir del id de un artículo busca en la tabla "post" de la base de datos se obtienen los parámetros del articulo y con esta información *renderiza* la vista "blog/edit".

- **Método de Solicitud**: POST
- **Ruta**: "/blog/edit/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUserblog.
- **Variables del body**: title, desc
- **Funcionalidad**: Sirve para actualizar tanto el título como la descripción del blog, a parir del id del articulo lo busca en la tabla "posts" de la base de datos y actualiza los parámetros.

- **Método de Solicitud**: POST
- **Ruta**: "/blog/edit/image/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUserblog.
- **Funcionalidad**: Recibe una nueva imagen de un artículo, después comprueba si el articulo pose imagen ya una imagen en cuyo caso la elimina, guarda la imagen en la carpeta "post_pic" del servidor con el nombre post_pic + id del articulo y por último modifica en la tabla "post" de la base de datos el parámetro de la ruta a la imagen.

- **Método de Solicitud**: POST
- **Ruta**: "/blog/edit/image/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUserblog.
- **Funcionalidad**: Recibe una nueva imagen de un artículo, después comprueba si el articulo pose imagen ya una imagen en cuyo caso la elimina, guarda la imagen en la carpeta "post_pic" del servidor con el nombre post_pic + id del articulo y por último modifica en la tabla "post" de la base de datos el parámetro de la ruta a la imagen.

Figura 43 *Añadir fotos a un artículo*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/blog/photo/edit /:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUserblog.
- **Funcionalidad**: A partir del identificador de un artículo busca en la tabla 'photos_post' todas las fotos asociadas a dicho artículo y con esta información *renderiza* el componente "partials/photo/ photo_post_edit "

- **Método de Solicitud**: POST
- **Ruta**: "/blog/photo /add/:id"
- ***Middleware* de autenticación**: isValidUser, isOwnerUserblog.
- **Funcionalidad**: Añade una foto a un artículo concreto. Primero guarda la foto en la carpeta 'photos_post' y después añade a la tabla "photos_post" de la base de datos una nueva línea con la ruta de la foto y el identificador del articulo al que pertenece.

- **Método de Solicitud**: DELETE
- **Ruta:** "/blog/ photo/delete/:id"
- ***Middleware* de autenticación:** isValidUser, isOwnerUserblog.
- **Funcionalidad:** Elimina una foto a partir de su identificador.

3.6.4 Estructura de la base de datos de los artículos/noticias

Tabla de los artículos 'post'

Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador del articulo / noticia
user_id	INT	11	Identificador del usuario creador del articulo / noticia
title	VARCHAR	60	Título del articulo / noticia
desc	TEXT		Descripción del evento
image	TEXT		Ruta a la Imagen principal
full_name	TEXT		Nombre completo del Co-organizer
video	TEXT		URL del video asociado (actualmente no se usa)

status	ENUM	'pending','valid','admin'	Estado que determina si es un artículo o una noticia
create_at	DATETIME		Fecha de creación del articulo / noticia
update_at	DATETIME		Fecha de la última modificación del artículo / noticia

Cada artículo aparte de tener su propia tabla en la base de datos está relacionado mediante claves foráneas a 2 tablas más estas son:

1. Fotos de los artículos 'photos_posts': contiene la URL de todas las fotos que han sido añadidas a un artículo, esta tabla estará relacionada con la de los eventos gracias la clave foránea (Id 'post' ->id_post 'photos_post')

2. *Co-organizer* creador del artículo 'user': Contiene toda la información personal del creador del artículo, esta tabla estará relacionada con la de los eventos gracias la clave foránea (user_id 'posts' ->id 'user')

Tabla de las fotos asociadas a los artículos 'photos_post'

Nombre	Tipo de datos	Longitud/conjunto	Descripción
id	INT	11	Identificador de la foto
post_id	INT	11	Identificador del artículo asociado a la foto
image	TEXT		Ruta a la foto
title	VARCHAR	50	Título del artículo asociado a la foto
create_at	DATETIME		Fecha en la que se añadió la foto al articulo

3.7 Administración de *co-organizers*, Eventos y Artículos/post

3.7.1 Administración de *co-organizers*

Los administradores disponen de dos vistas de administración para *co-organizer*, una que contiene los que están pendientes de validación (*Pending Co-organizer*) y otra con los *co-organizers* que han sido validados (*Register Co-organizer*). Ambas vistan muestran a los *co-organizers* en una tabla con las columnas de email, nombre "Frist name", apellidos "Last name", edad "Age" y estado "Status" el cual aparte de mostrar el estado también puede ser modificado. También tienen disponible un botón "Edit" el cual abrirá modal con todos los datos del *co-organizer* (menos la contraseña y la imagen de perfil) siendo todos estos datos modificables por el administrador.

- Pending Co-organizer

Los administradores revisaran los perfiles de los futuros *co-organizers* en esta vista. Para validar un *co-organizer* tendrán que actualizar el estado de "*Pending_approval*" a "*valid*" y pulsan el botón "Update" aparte de validar al *co-organizer* también se mandará un correo avisándolo de que ha sido aceptado y que ya puede crear eventos. En el caso de rechazar al *co-organizer* tendrá que cambiar el estado de "Pending_approval" a "Rejected" y pulsan el botón "Update" aparte de rechazar al *co-organizer* también se mandará un correo avisándolo de que no cumple los requisitos.

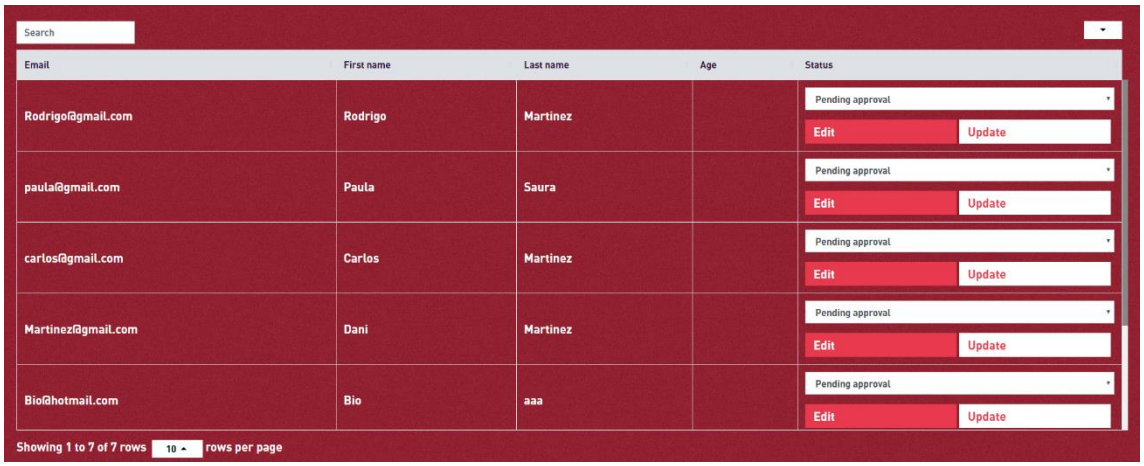


Figura 44 *Tabla de Co-organizers pendientes*

- Register *co-organizer*

Los administradores podrán revisar los perfiles de los *co-organizers* que ya han sido validados. Podrán bloquear a un *co-organizer* si lo ven necesario, para hacer esto solo tienen que cambiar su estado de "Unblock" a "block" y mientras este bloqueado el *co-organizer* no podrá crear nuevos eventos, ni nuevos artículos y no aparecerá en la vista de *co-organizers*.

Figura 45 *Tabla de co-organizers válidos*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/admin/users?sta=String"
- ***Middleware* de autenticación**: isAdmin.
- **Funcionalidad**: *Renderiza* la vista "admin/userList" en la que un administrador podrá ver una lista con los *co-organizers* pendientes de validación si la variable "sta" es igual a 'pending' o vera una lista con los *co-organizers* registrados si la variable "sta" es igual a 'registered'.

- **Método de Solicitud**: POST
- **Ruta**: "/admin /users/status"
- ***Middleware* de autenticación**: isAdmin.
- **Funcionalidad**: Sirve para que un administrador actualice el estado de un *co-organizer*.

3.7.2 Administración de eventos

Los administradores podrán revisar todos los eventos creados por los *co-organizers* en esta vista. Los eventos se muestran en una tabla con las columnas de título "Title", fecha de inicio "Start date", fecha de fin "End date", ciudad "city" y estado "Status" que además de mostrar el estado también puede ser modificado. También tienen disponible un botón "Edit" que re-direccionara al Administrador a la vista de edición del evento seleccionado.

Solo un administrador podrá poner en marcha un evento modificando el estado de "Pending" a "On Going".

Figura 46 *Tabla de eventos*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: "/admin /events"
- ***Middleware* de autenticación**: isAdmin.

- **Funcionalidad**: *Renderiza* la vista “admin/ eventsList” en la cual un administrador puede ver una lista con todos los eventos creados por los *co-organizers*.

- **Método de Solicitud**: POST
- **Ruta**: “/admin /events/status”
- ***Middleware* de autenticación**: isAdmin.
- **Funcionalidad**: Sirve para que un administrador actualice el estado de un evento de algún *co-organizer*.

3.7.3 Administración de artículos/noticas

Los administradores podrán revisar todos los artículos creados por los Co-organizers y los suyos propios en esta vista. Los artículos/noticias se muestran en una tabla con las columnas de título “Title”, nombre del crador “User name” y estado “Status” el cual aparte de mostrar el estado también puede ser modificado. También tienen disponible un botón “Edit” que redireccionara al Administrador a la vista de edición del artículo seleccionado.
Un administrador podrá validar un artículo cambiando su estado a “valid” y si quiere publicar una noticia entonces tendrá que cambiar el estado de uno de sus artículos a “Admin”.

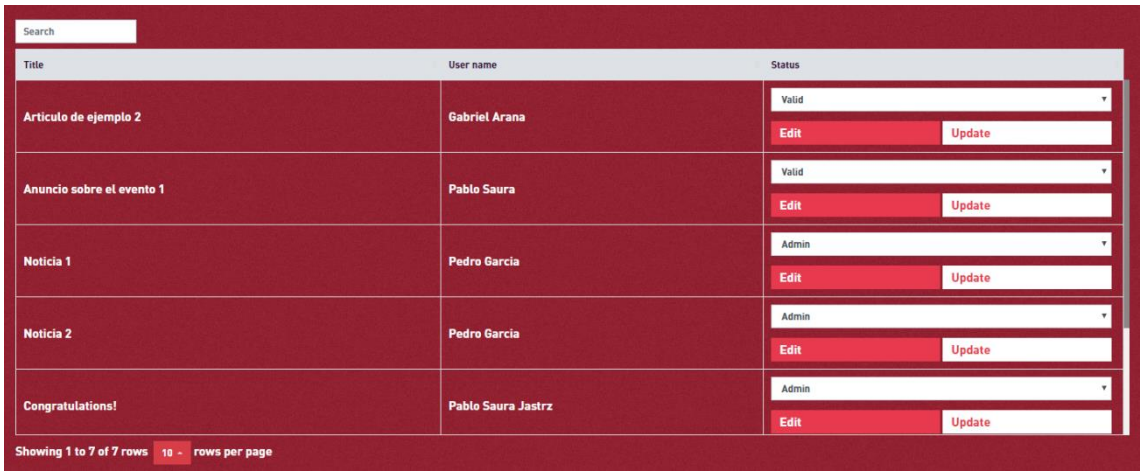


Figura 47 *Tabla de Artículos*

Llamadas al servidor

- **Método de Solicitud**: GET
- **Ruta**: “/admin /posts”
- ***Middleware* de autenticación**: isAdmin.
- **Funcionalidad**: *Renderiza* la vista “admin/postList” en la cual un administrador puede ver una lista con todos los artículos creados por los *co-organizers*.

- **Método de Solicitud**: POST
- **Ruta**: “/admin /posts/status”
- ***Middleware* de autenticación**: isAdmin.
- **Funcionalidad**: Sirve para que un administrador actualice el estado de un artículo de algún *co-organizer*.

Capítulo 4. Análisis de vulnerabilidades de la plataforma web

En este capítulo se va analizar la seguridad frente a los ataques más comunes a los que una plataforma web está expuesta. Para realizar esta prueba se usará software diseñado específicamente para encontrar vulnerabilidades en las webs mediante la simulación de ataques.

4.1 Pruebas de vulnerabilidad con OWASP ZAP

OWASP ZAP (abr. para Zed Attack Proxy) es un escáner de seguridad web de código abierto. Pretende ser utilizado como una aplicación de seguridad y como una herramienta profesional para pruebas de penetración que permite encontrar vulnerabilidades en las aplicaciones y plataformas web. [26]

Empezaremos realizando un análisis automatizado contra la plataforma web, el cual no encuentra ninguna vulnerabilidad grave. Sólo se obtienen 2 alertas con un riesgo bajo y una alersta con un riego medio. Ninguna de ellas supone ningún riesgo grave para la plataforma web.

Las tres alertas son las siguientes:

1. CSP Scanner:Wildcard Directive

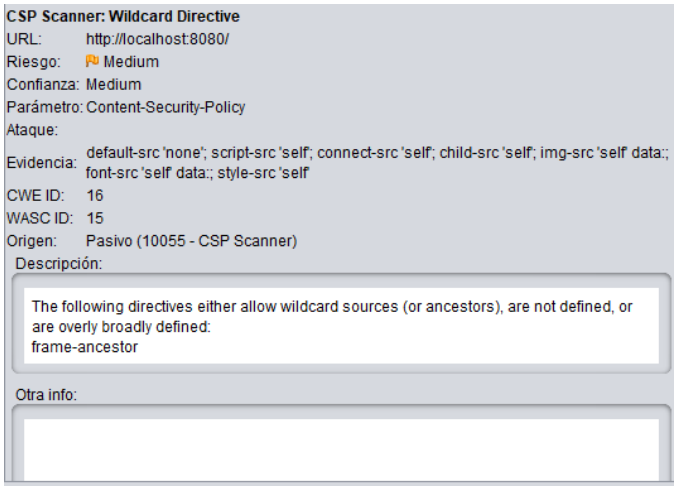


Figura 48 *CSP Scanner:Wildcard Directive*

Descripción: Las directivas 'frame-ancestor' no están definidas o están definidas de forma demasiado amplia.

2. Absence of Anti-CSRF Tokens

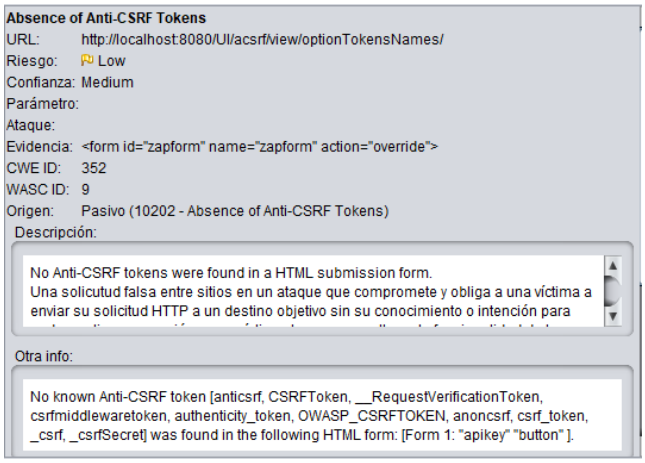


Figura 49 *Absence of Anti-CSRF Tokens*

Descripción: No se encontraron tokens Anti-CSRF en en los formulario de envío HTML.

Una solicitud falsa entre sitios es un ataque que compromete y obliga a una víctima a enviar su solicitud HTTP a un destino objetivo sin su conocimiento o intención para poder realizar una acción como víctima. La causa oculta es la funcionalidad de la aplicación utilizando acciones de URL/formulario que pueden ser adivinados de forma repetible. La naturaleza del ataque es que CSRG explota la confianza que un sitio web proporciona a un usuario. Por el contrario, las cadenas de comandos de los sitios cruzados (XSS) explotan la confianza que un usuario proporciona en un sitio web. Al igual que XSS, los ataques CSRG no son de forma necesaria de sitios cruzados, pero existe la posibilidad de que sí pueden serlo. La falsificación de las solicitudes ente los sitios también se conoce como CSRF, XSRG, ataques con un solo clic, montaje de sesión, diputado confundido y navegación en alta mar.

Los ataques de CSRG son muy efectivos en varias situaciones, que incluyen:

- La victima tiene una sesión activa en el sitio de destino.
- La víctima se autoriza por medio de la autenticación HTTP en el sitio de destino.
- La víctima se encuentra en la misma red local que el sitio de destino.

CSRF se ha utilizado especialmente para poder realizar una acción contra un sitio objetivo utilizando los privilegios de la víctima, pero se han revelado técnicas recientes para difundir información al obtener el acceso a la respuesta. El riesgo de divulgación de información aumenta de forma drástica cuando el sitio de destino se encuentra vulnerable a XSS, porque XSS se puede utilizar como una plataforma para CSRF, lo que le permite al atacante que opere desde adentro de los límites de la misma política de origen.

3. CSP Scanner: Notices

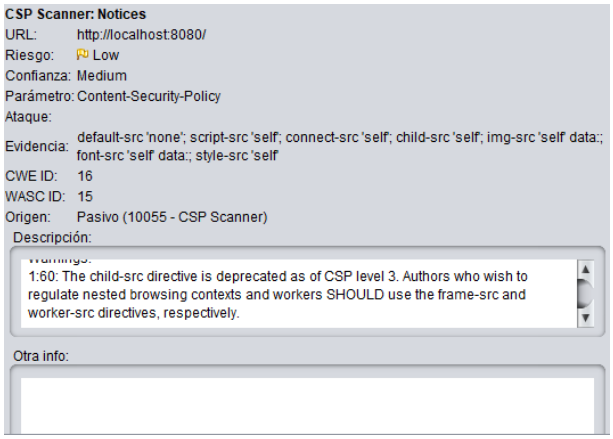


Figura 50 *CSP Scanner: Notices*

Descripción: La directiva child-src está en desuso a partir del nivel 3 de CSP. Los autores que deseen regular los contextos de navegación anidados y los trabajadores deberían utilizar las directivas frame-src y worker-src, respectivamente.

4.2 Pruebas de vulnerabilidad con ScanMyServer

ScanMyServer es un web que sirve para conocer el estatus actual de los certificados de seguridad de una plataforma web, así como del servidor donde está hospedado. También analiza el sitio web de malware, inyecciones de SQL y algunas otras vulnerabilidades que serán de mucha utilidad para mantener un sitio seguro o hacer los ajustes necesarios para lograrlo. [27]

Para poder realizar este test primero hemos de permitir recibir llamadas de tres direcciones IP de ScanMyServer. Para ello añadiré a mi lista blanca de la web las direcciones IP (162.213.1.246, 54.235.163.229, 52.25.189.91). Cuando finalice el test retiraré las direcciones IP de la lista blanca.

Una vez pasado el test, ScanMyServer no encuentra ninguna vulnerabilidad potencialmente peligrosa en la web, solo encuentra 8 alertas de nivel bajo que no son potenciales amenazas.

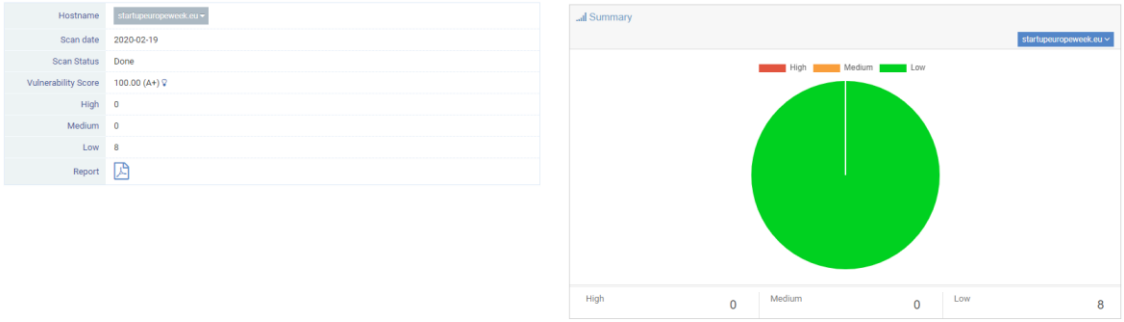
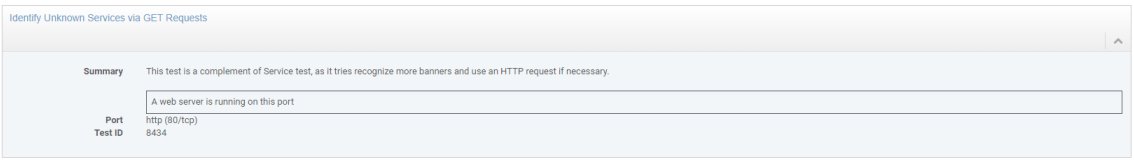
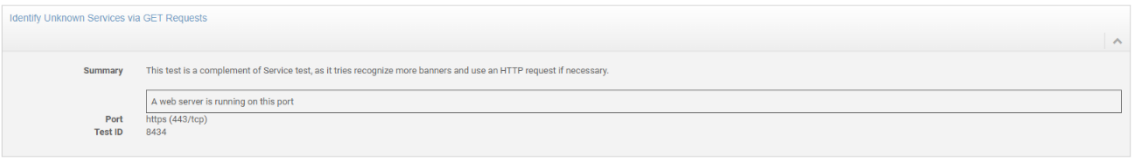
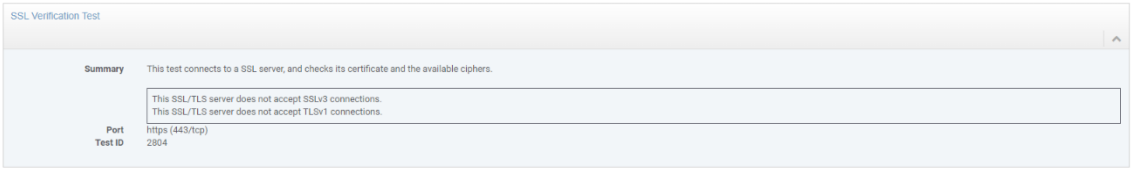


Figura 51 *Grafico con las alertas*

Las ocho alertas obtenidas son las siguientes:







Tras realizar estas pruebas podemos afirmar que la plataforma web es segura y no es vulnerable a amenazas serias como inyecciones SQL o falsifica

4.3 Referencias

[26] OWASP ZAP
https://www.zaproxy.org/docs/api/
[27] ScanMyServer
https://scanmyserver.com/ción de solicitudes entre sitios, por ejemplo.

Capítulo 5. Puesta en producción de la plataforma web

En este capítulo se explicará paso por paso, todas las acciones necesarias para poner en funcionamiento la plataforma web implementada en el domino startupeuropeweek.eu haciendo uso de cPanel.

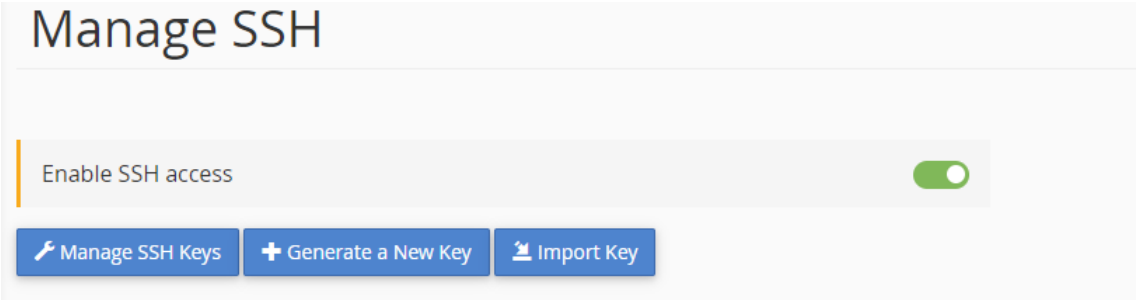
5.1 Subida del directorio a cPanel

En primer lugar, será necesario traer el repositorio de la plataforma web que se encuentra alojado en github como repositorio privado al alojamiento web Namecheap haciendo uso de cPanel.

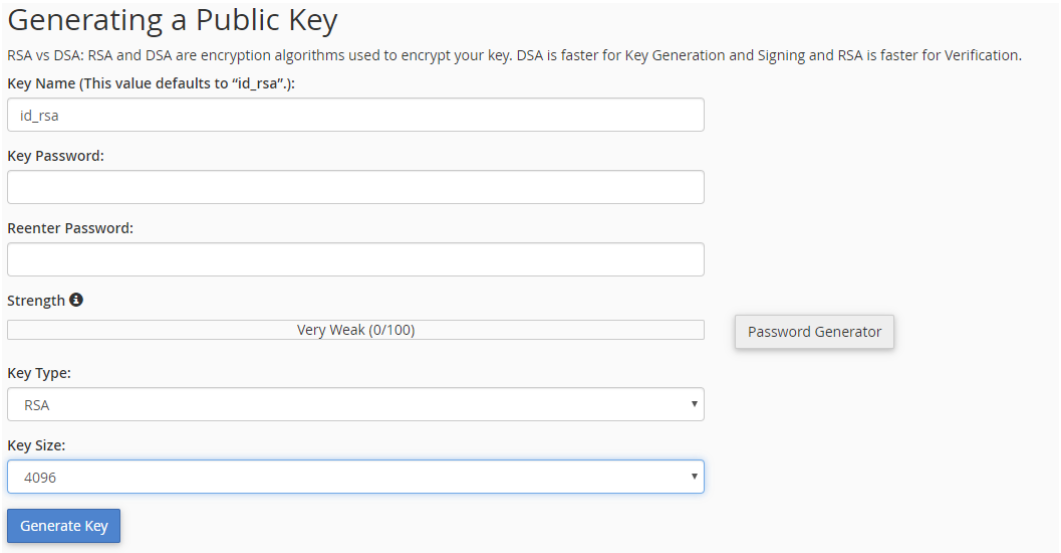
cPanel es un panel de control para administrar servidores de alojamiento web que proveen herramientas de automatización y una interfaz gráfica basada en páginas web. [28]

Pasos a seguir:

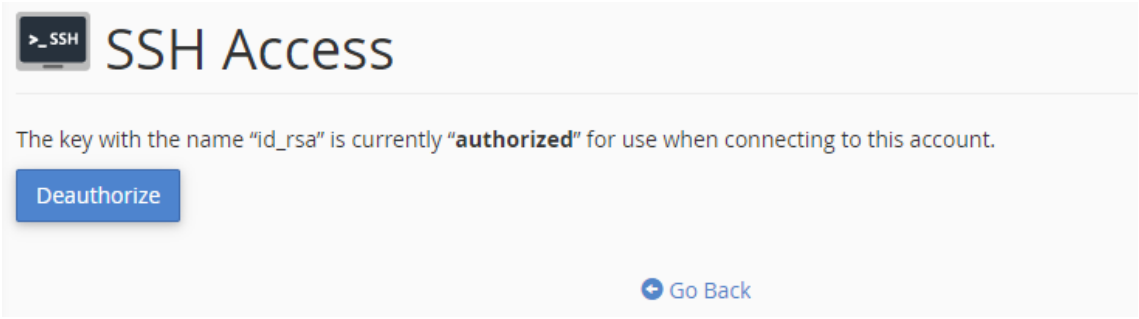
1. Habilitar el acceso mediante SSH en cPanel que está cerrado por seguridad, por lo tanto, cuando terminemos este proceso lo volveremos a cerrar.



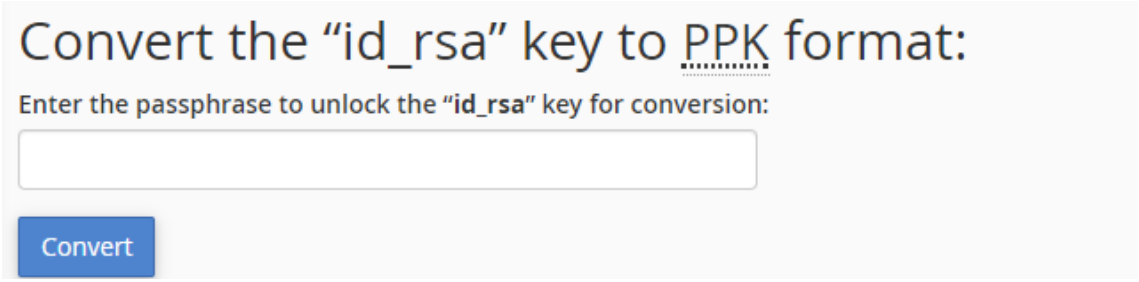
Para establecer la conexión mediante SSH primero generaremos una clave pública y una privada mediante el sistema criptográfico RSA. El propio cPanel nos la posibilidad de generar una clave pública a la que necesitamos pasarle un nombre, contraseña, tipo de clave y tamaño. Una vez generada, su clave privada será creada automáticamente.



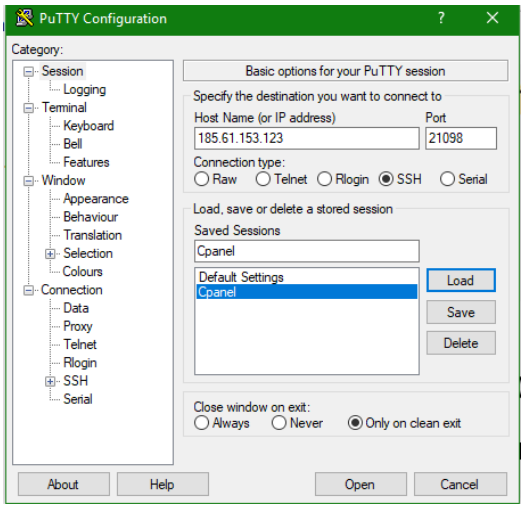
Ahora será necesario autorizar la clave pública creada para poder utilizarla.



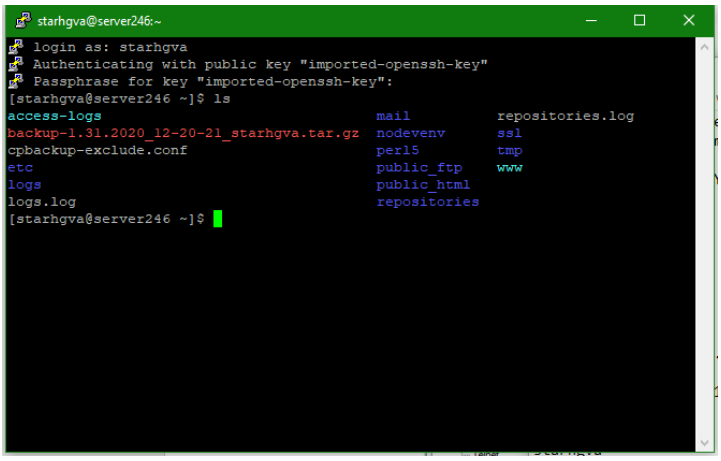
Por último, tendremos que descargarnos la clave privada en formato PPK para poder usarla en el ordenador.



2. Para poder conectarnos remotamente usaremos el programa puTTY, que además de introducir la dirección IP y el puerto del servidor, también tendrá que configurarse para que use la clave privada generada en el punto anterior.



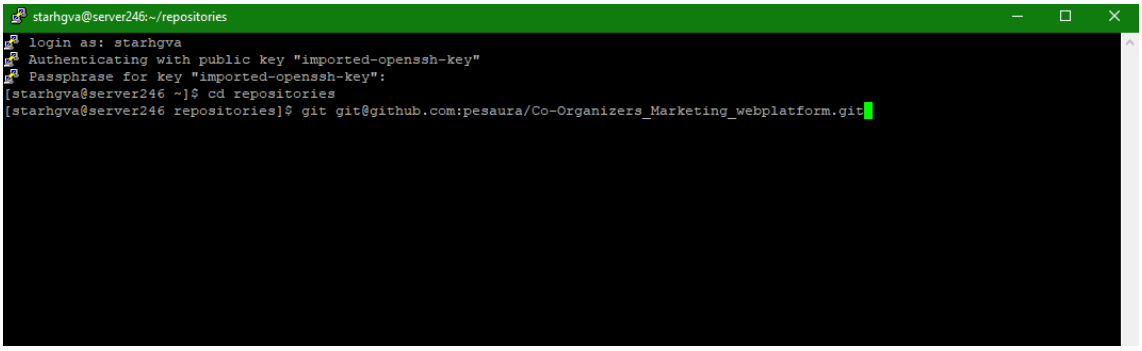
Una vez establecida la conexión nos pedirá que iniciemos sesión, pero cuando escribamos nuestro usuario en vez de pedirnos su contraseña nos avisará que la autenticación se está realizado mediante clave pública y la contraseña que debemos utilizar es la que usamos en el punto anterior al generar la clave privada.



3. En el servidor crearemos una carpeta llamada 'repositories' y será ahí donde clonaremos el repositorio de la web haciendo uso de git. Solo será necesario entrar en el directorio y usar el siguiente comando:

   git git@github.com:pesaura/Co-Organizers_Marketing_webplatform.git

   - Al ser un repositorio privado tendremos que añadir la clave pública generada en el punto uno a mi cuenta de github.



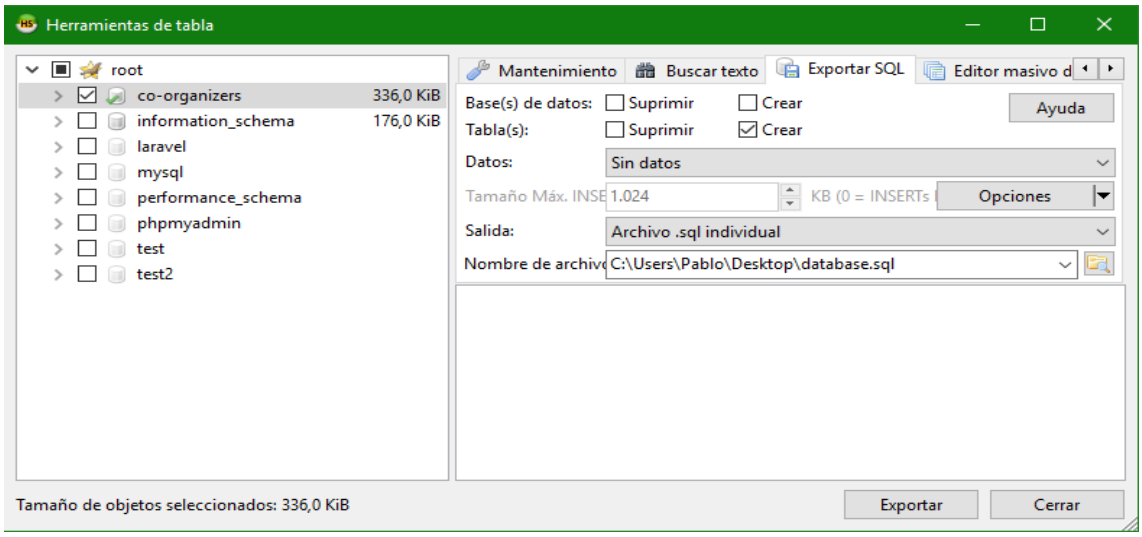
   Una vez hecho esto el repositorio de la plataforma web ya se encontrará en el servidor de producción. Para finalizar volvemos a cerrar el acceso mediante SSH.

5.2 Crear base la base de datos Mysql en Cpanel

En este apartado explicaremos la creación de una base de datos MySQL en cPanel y la importación de la base de datos de la web para poder usarla desde el servidor de producción.

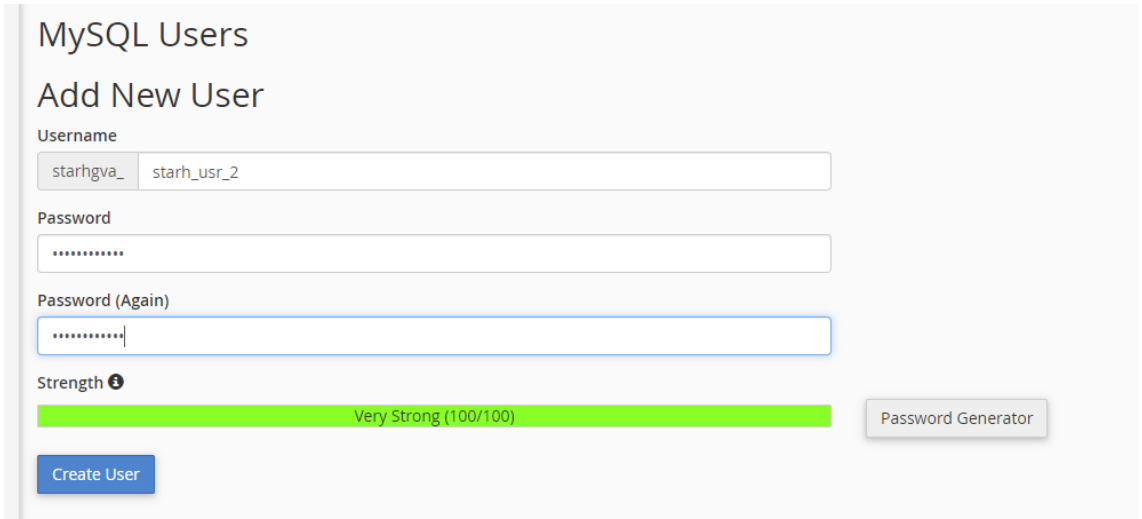
Pasos a seguir:

1. En primer lugar, será necesario exportar mi base de datos llamada 'Co-organizers' con el programa 'heidiSQL' como un archivo .sql sin datos.

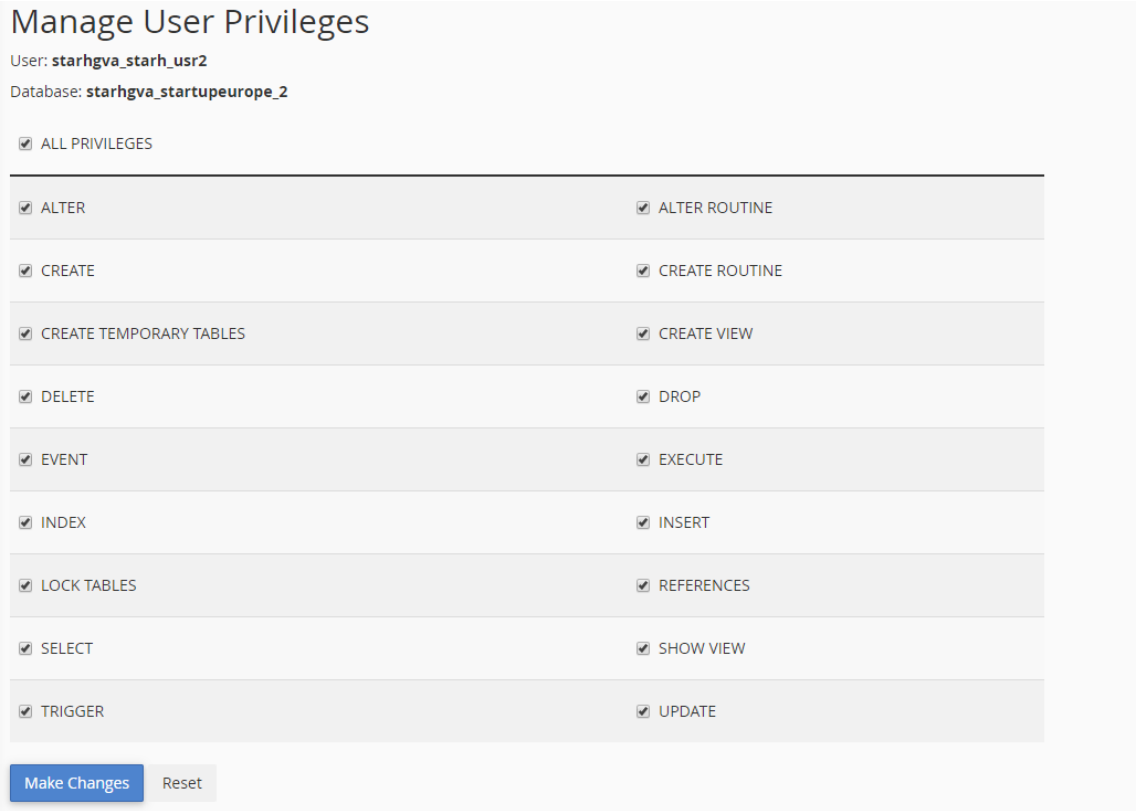


2. En el panel de Cpanel “MySQL® Databases” crearemos una nueva base de datos.

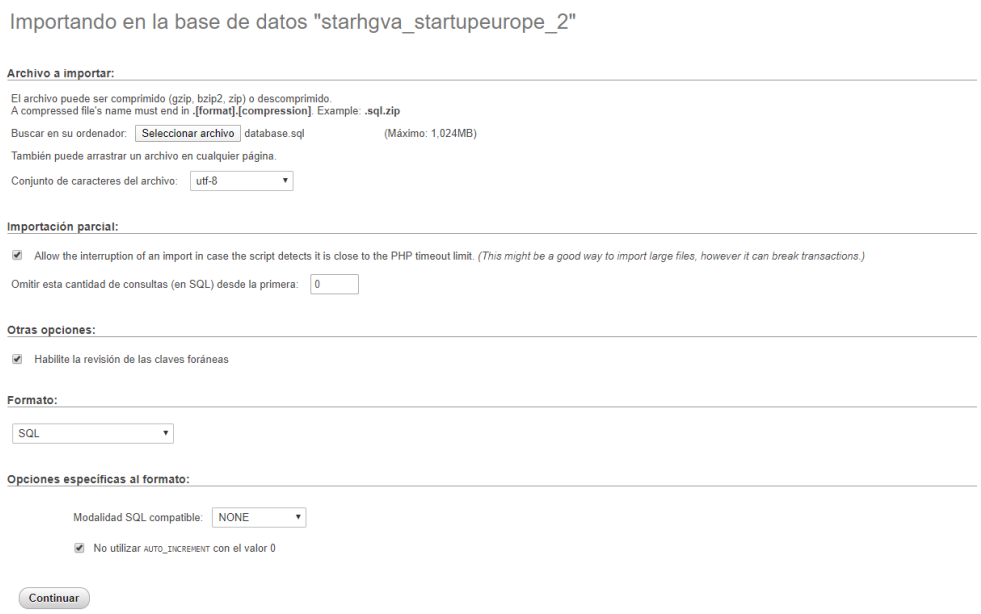
3. En el panel de Cpanel “MySQL® Databases” tendremos que crear un usuario para poder acceder a la base de datos.



4. Una vez creada la base de datos y un usuario tendremos que añadir a dicho usuario a la base de datos como usuario privilegiado. Es decir, darle permisos de lectura, escritura, ….



5. En el phpmyadmin de cPanel accederemos a la base de datos 'starhgva_startupeurope_2' y en la pestaña de importar tendremos que importar la base de datos de la web.



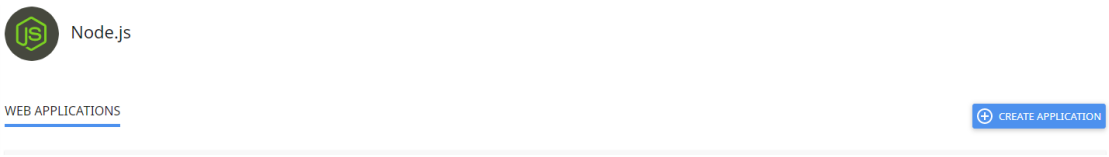
Una vez importada ya podremos usarla en la plataforma web SEW.

5.3 Configurar la aplicación web Node.js

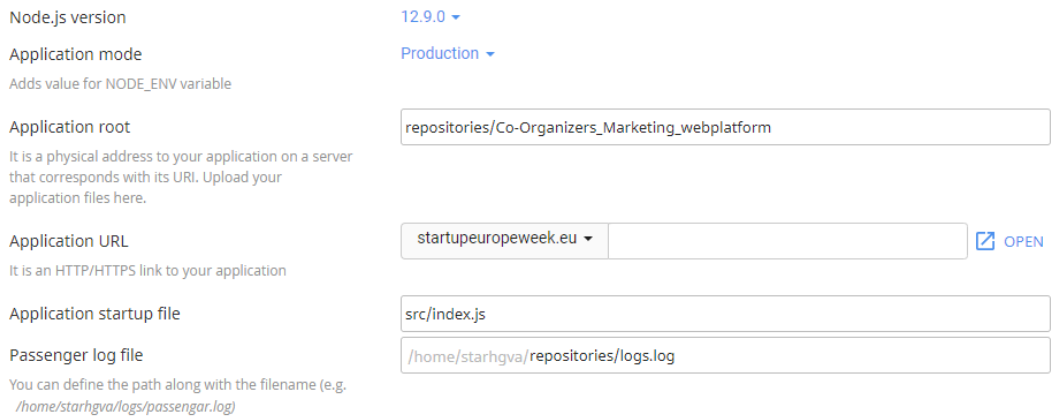
Por último, pondremos en marcha la plataforma web creando una aplicación Node.js desde el propio cPanel aprovechado que ahora da soporte para crear aplicaciones Node.js de una manera visual y muy intuitiva.

Pasos a seguir:

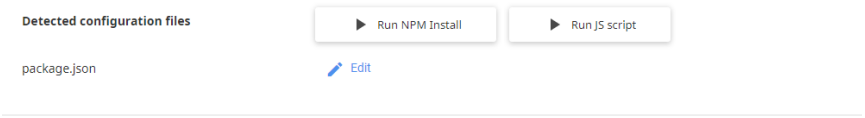
1. Al acceder a "Setup Node App", crearemos una nueva aplicación desde el botón que encontraremos en la parte superior derecha.



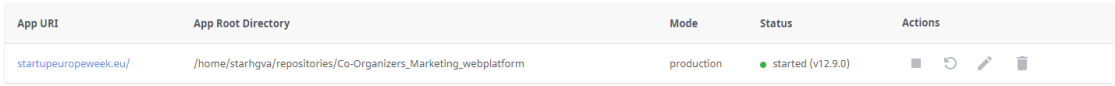
2. A continuación, tendremos que configurar la nueva app Node.js con las siguientes opciones:
   - **Node.js Version**: La visión de Node.js que se ha usado es la 12.9.0
   - **Application mode**: Cuando se estaba desarrollando la web, el modo que se usaba era "Development", para ponerla en producción el modo será "Production".
   - **Application Root**: Tenemos que especificar una carpeta que será la ruta de la aplicación. Esta será la dirección al repositorio que hicimos en el apartado 5.1
   - **Application URL**: Tenemos que asignarle una URL a la aplicación, este será el dominio de la web de SEW.
   - **Application startup file**: Tendremos que especificar la ruta al archivo que va a contener la aplicación.
   - **Passenger log file**: Tendremos que especificar la ruta al archivo que contendrá los registros de la aplicación.



3. Antes de iniciar la aplicación será necesario ejecutar el comando "npm install" para instalar todas las dependencias de la aplicación node.js.



4. Por último, iniciaremos la aplicación Node.js y ya estará accesible la plataforma web de SEW.



5.4 Referencias

[28] cPanel
https://en.wikipedia.org/wiki/CPanel

Capítulo 6. Conclusiones

6.1 Conclusiones del proyecto

Tras haber realizado este Trabajo Fin de Grado puedo sacar las siguientes conclusiones.

En primer lugar, he profundizado mis conocimientos sobre tecnologías que aprendí en la carrera, como el uso de JavaScript que ha sido fundamental para crear la plataforma web o el uso de bases de datos SQL, entender cómo crear una base de datos bien estructurada y cómo realizar las consultas a dicha base de datos. Todo ello ha sido fundamental para crear la plataforma web.

Realizar una API REST me ha sido de mucha utilidad para entender cómo se crean los servicios web, pues en la actualidad no existe proyecto o aplicación web que no disponga de una API REST, Twitter, YouTube, los sistemas de identificación con Facebook… hay cientos de empresas que generan negocio gracias a las APIs REST.

Trabajar con Node.js, al ser una tecnología que lleva un tiempo en el mercado, ha resultado muy cómodo a pesar de no estudiarlo durante la carrera. Hay muchísima documentación al respecto, mucha gente ya ha tenido los mismos errores que yo, por lo que buscar información sobre cómo solucionarlos ha sido relativamente fácil.

Realizar el Trabajo Fin de Grado en colaboración con una empresa real, en este caso Startup Europe Week, dándome estos las especificaciones sobre la web, qué requisitos debía satisfacer y qué diseño debía tener la plataforma web me ha resultado una buena experiencia al poder aplicar mis conocimientos en un caso práctico real. Aunque la comunicación con la empresa fue un poco tibia al principio fue mejorando a medida que el proyecto avanzaba sobre todo en las fases finales del proyecto.