UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una aplicación web de gestión colaborativa para un club de triatlón

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jose Enrique Pérez Rubio

Tutor/a: Manuela Albert Albiol
Victoria Torres Bosch

2016 - 2017

Resumen

Se ha desarrollado una intranet para sustituir el actual método de contacto y navegación de los usuarios el cual es un foro. La nueva aplicación cuenta con más funcionalidades que no estaban disponibles anteriormente. La página web está desarrollada en web2py, un *framework* de Python.

Como patrón de diseño para la implementación se utilizará el conocido Modelo Vista Controlador (MVC), arquitectura estándar hoy en día el cual separa los datos y la lógica de las vistas del usuario. Este diseño facilita el desarrollo y mantenimiento de las aplicaciones.

Palabras clave: triatlón, intranet, web2py, framework, Python. MCV

Abstract

This Intranet has been developed to replace the current users contact and navigation method, nowadays it is a forum. The new application has more functionality than previously available. This web page is developed in Python web2py's *framework*.

As design for the implementation we'll be using the Model View Controller (MVC), standard architecture because it separates the data and the logic from user's view. This design improves the development and maintenance of applications.

Keywords: triathlon, intranet, web2py, framework, Python, MVC

Agradecimientos

Antes de nada, me gustaría dar las gracias a:

Mis padres, por alentarme a continuar mi educación y han trabajado siempre muy duro para poder brindarme la oportunidad que ellos nunca tuvieron para poder continuar mis estudios. Una deuda que nunca podrá ser saldada.

A mis profesores por aportarme los conocimientos necesarios para afrontar los desafíos que se me han planteado a lo largo de mi vida en especial a Rosa del colegio Eliseo Vidal, por su dureza constancia al no echar la toalla con ninguno de sus alumnos y Don Bernardus (como él nos obligaba a llamarle) del IES Cid Campeador en 3º de la ESO por motivarme nuevamente cuando perdí el interés por los estudios por causas de la vida.

TABLA DE CONTENIDOS

1. Introducción 10
1.1. Motivación 10
1.2. Objetivos 10
2. Herramientas y tecnologías empleadas 12
2.1. XAMPP [1] 12
2.1.1. Apache [2] 13
2.1.2. MySQL [3] 13
2.2. PHP [4] 13
2.3. Python [5] 14
2.4. Web2py [6] 15
2.4.1. Modelo 17
2.4.2. Controlador 17
2.4.3. Vista 18
2.4.4. Seguridad 19
3. Metodología 20
3.1. Análisis de necesidades del usuario 20
3.2. Diseño y evaluación 21
3.3. Implementación y evaluación 22
4. Preparación del entorno de trabajo 22
4.1. Python 22
4.2. XAMPP 22
4.3. Web2py 24
4.4. Entorno de programación 26
5. Implementación 26
5.1. Modelo (Base de datos) 26
5.1.1. Código (db.py) 27
5.1.1.1. Conexión con la BBDD 27
5.1.1.2. Creación de tablas 28
5.2. Vistas 31
5.2.1. Index 34
5.2.2. Formularios 36
5.3. Controlador 37

5.4. Plugins 38
5.4.1. Bootstrap 38
5.4.2. Autosize [9] 38
5.4.3. Datatables [10] 38
5.4.4. Switch [11] 40
5.4.5. Datepicker [12] 40
5.4.6. Duallistbox 41
5.4.7. Select2 42
6. Manual de usuario 43
6.1. Creación del proyecto 43
6.2. Crear una cuenta de usuario 44
6.3. Preparar una nueva temporada 46
6.4. Vista como usuario 46
6.5. Vista como entrenador 48
7. Conclusiones, problemas y contratiempos 49
8. Bibliografía 50

ÍNDICE DE FIGURAS

Figura 1: Panel de control XAMPP con Apache y MySQL funcionando 12
Figura 2: Código Python ejemplo 15
Figura 3: Código html y php de ejemplo 16
Figura 4: Código Python ejemplo 16
Figura 5: Flujo de trabajo de una solicitud en web2py 16
Figura 6: Conexión con la base de datos almacenada en una variable..... 17
Figura 7: Definición de una tabla 17
Figura 8: Código ejemplo de una función definida en el controlador boletin.py 18
Figura 9: Ejemplo código vista de formulario para los boletines 18
Figura 10: Proceso del desarrollo centrado en el usuario..... 20
Figura 11: Instalación de XAMPP..... 23
Figura 12: Entorno phpMyAdmin para administrar la BBDD 24
Figura 13: Servidor web2py en funcionamiento recibiendo peticiones 25
Figura 14: Creación de una nueva aplicación. 25
Figura 15: Entorno de desarrollo Aptana Studio..... 26
Figura 16: Base de datos con sus relaciones en SQL Designer 27
Figura 17: Variable para la conexión con la BBDD 27
Figura 18: Extensión de la tabla creada por defecto que contiene datos de usuario. 29
Figura 19: Configuración en la definición de tablas. 29
Figura 20: Configuración para el correo. 30
Figura 21: Tabla que define unos campos para añadirlos en las tablas deseadas. 30
Figura 22: Ejemplo de código para definir una tabla. 31
Figura 23: Ejemplo de introducción de datos por defecto. 31
Figura 24: Interfaz cargada en el layout.html vista desde un perfil usuario. 32
Figura 25: Código para usar la plantilla web 32
Figura 26: Código que obtiene las alertas 33
Figura 27: Carga de alertas en el layout.html 34
Figura 28: Pantalla principal de la aplicación..... 35
Figura 29: Página index de eventos que muestra el grid. 36
Figura 30: Página form de eventos 37
Figura 31: Estructura del controlador. 38
Figura 32: Plugin Autosize: Ejemplo de uso 38

Figura 33: Plugin Datatable: configuración. 40
Figura 34: *Plugin* datatables filtrando por un valor 40
Figura 35: *Plugin* switch: configuración y vista 40
Figura 36: *Plugin* datepicker con calendario y hora 41
Figura 37: *Plugin* duallistbox: ejemplo desactivado 41
Figura 38: *Plugin* duallistbox: ejemplo activado 41
Figura 39: *Plugin* duallistbox: configuración. 42
Figura 40: *Plugin* Select2: ejemplo de búsqueda 42
Figura 41: Directorios de web2py. 43
Figura 42: Tablas en la base de datos creadas de manera automática. 44
Figura 43: Login de la página 44
Figura 44: Email de verificación. 44
Figura 45: Cuenta bloqueada a la espera de ser revisada. 45
Figura 46: Listado de los usuarios registrados. 45
Figura 47: Sección de administración en el perfil de usuario 46
Figura 48: Listado de noticias. 47
Figura 49: Imágenes secundarias de una noticia y como modificarlas o borrarlas 47
Figura 50: Como acceder al perfil de usuario o cerrar la sesión. 48
Figura 51: Asignar entrenamiento a usuarios 48

1. Introducción

1.1. Motivación

El reto planteado para el trabajo final de grado, consiste en el desarrollo de una aplicación web para el equipo de triatlón de la Universidad Politécnica de Valencia

Hasta el momento se empleaba un foro, en el que los usuarios del club se registraban, para acceder a los datos catalogados en entradas separadas. Esto era muy limitado al no poder adaptar la programación a las necesidades reales.

1.2. Objetivos

La intención del proyecto es cambiar dicho foro por una aplicación web en la cual los usuarios tendrán acceso para llenarla de contenido. Algunos de los datos recogidos serán empleados en otra página de presentación del club a la que accederán los usuarios no registrados que quieran conocer el equipo.

En un futuro, este trabajo será ampliado por otro alumno que realice su trabajo final de grado con objetivo de añadir nuevas funcionalidades, como una sección de stock, que permita comprar a los usuarios equipamiento de temporada.

Parte de los objetivos del proyecto era realizar la aplicación con Web2py, emplean distintas tecnologías y lenguajes para la realización del proyecto. Estas tecnologías son:

- **Web2py**: *Framework* escrito y programado en Python.
- **MySQL**: Para la base de datos.
- **Front-end**: HTML, CSS, javascript y Ajax.
- **Python**: Lenguaje de programación para la parte back-end.
- **Bootstrap**: Para hacer la web *responsive*.

La página debe ser *responsive*, esto significa que el diseño visto y manejado por el usuario, debe de adaptarse independientemente del dispositivo para ofrecer una correcta visualización. Esto permite al desarrollador reducir los tiempos de desarrollo y evitar la creación de páginas duplicadas al no necesitar varias que dependan de un tipo de resolución y/o dispositivo. Al poder trabajar desde un equipo o un dispositivo móvil no se requiere el desarrollo de una aplicación especializada para dispositivos móviles.

Al igual que en el foro, la aplicación se divide en secciones las cuales mediante una serie de roles definidos (Usuario, entrenador, usuario entrenador, ayudante y administrador) serán o no accesibles dependiendo el usuario que se encuentre registrado en la aplicación permitiendo limitar el control total a todo el mundo.

Secciones que ha de contener:

- Una sección donde administrar los miembros registrados en la que solo los usuarios con rol más elevado podrán acceder y una sección donde todo el mundo podrá modificar su propio perfil privado.
- Una sección donde poder gestionar los colaboradores.
- La posibilidad de crear asambleas, eventos o la difusión de boletines en formato PDF a los usuarios.
- Poder administrar los entrenamientos, noticias, colaboradores y carreras de una temporada.
- Poder poner los entrenamientos en formato PDF para una serie de usuarios dependiendo del entrenador.

2.Herramientas y tecnologías empleadas

Existen multitud de lenguajes de programación dedicados al ámbito web, este proyecto está realizado en Python. Hasta entonces nunca había trabajado con otro lenguaje más que con PHP en clase.

En este capítulo hablare sobre las distintas tecnologías empleadas en el proyecto con una breve descripción.

2.1. XAMPP [1]

Es un paquete de instalación independiente de plataforma, software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MariaDB, PHP, Perl. El programa se distribuye bajo la licencia GNU y actúa como un servidor web libre.

En este proyecto únicamente se va a emplear Apache para acceder a la configuración de MySQL por la interfaz web que nos brinda phpyMyAdmin. Se puede optar por no usar Apache y acceder mediante otros programas a la base de datos, pero he querido aprovechar la facilidad que brinda XAMPP con estos paquetes.

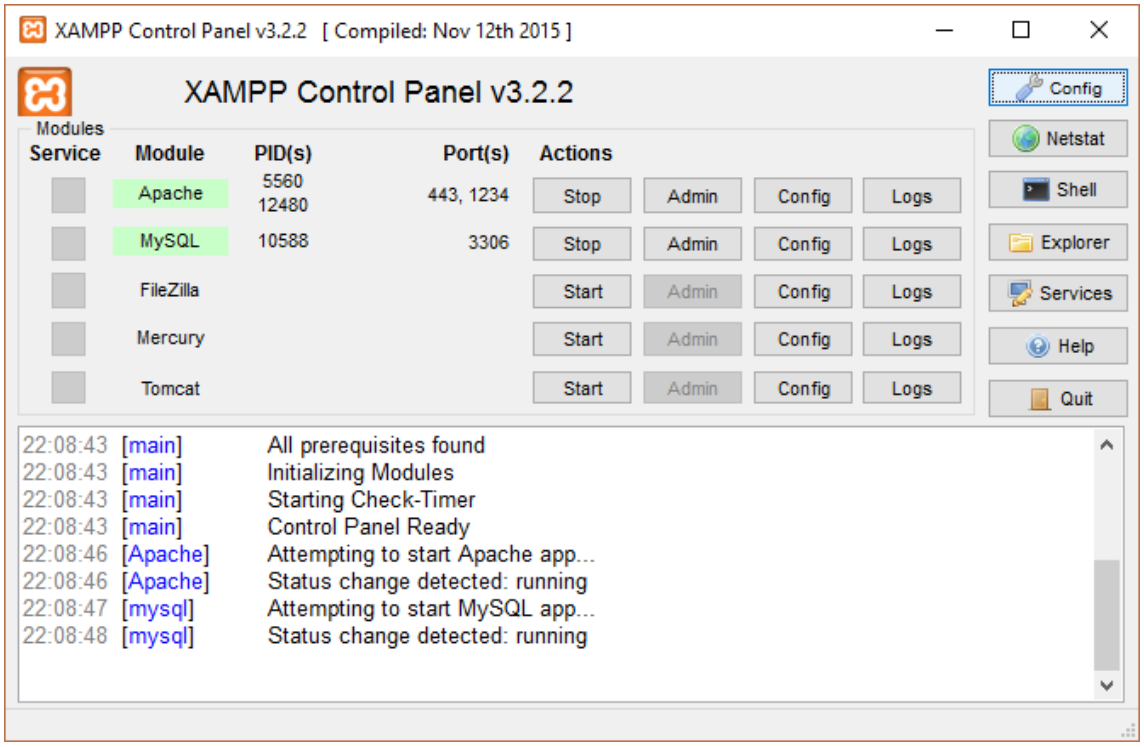

Figura 1: Panel de control XAMPP con Apache y MySQL funcionando

En la imagen anterior se puede observar que el servidor apache está funcionando sobre el puerto 1234 ya que el puerto por defecto me daba problemas con Skype. Otra opción es configurar en el propio Skype otro puerto para las llamadas y dejar el que viene por defecto en apache.

2.1.1. Apache [2]

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP.

Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de previsualizar y probar código mientras éste es desarrollado.

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la *Apache Software Foundation* dentro del proyecto HTTP Server (httpd).

La mayor parte de la configuración se realiza en el fichero apache2.conf (Ubuntu) o httpd.conf (Otros), en mi caso que he necesitado cambiar el puerto de apache porque daba problemas con el de Skype he añadido en el fichero citado anteriormente la siguiente línea “Listen 1234” para que escuche las llamadas por un nuevo puerto. Cualquier cambio en este archivo requiere reiniciar el servidor, o forzar la lectura de los archivos de configuración nuevamente.

2.1.2. MySQL [3]

MySQL es un sistema de gestión de bases de datos relacional multiplataforma desarrollado por Oracle Corporation y está considerada como la base datos open source más popular del mundo, otras bases de datos famosas son Oracle y Microsoft SQL Server, pero estas no son gratuitas.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

MySQL es muy utilizado en aplicaciones web, como Joomla, Wordpress, Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla.

2.2. PHP [4]

Antes de comenzar a comentar acerca de PHP, quiero recalcar que no se emplea en este proyecto, pero al ser un lenguaje del que he hecho uso, quiero hablar acerca de él y exponer una serie de comparativas frente a Python las cuales me parecen interesantes

conocer por cualquier programador que dude que lenguaje emplear para desarrollar su trabajo.

Es de los primeros si no el primer lenguaje que permitió la creación de páginas dinámicas en internet. Pese al paso del tiempo, ha sabido actualizarse y actualmente, es un lenguaje muy usado en todo el mundo, es utilizado por grandes empresas como Facebook, Wikipedia, etc. Cuenta con *Frameworks* muy potentes como Laravel el cual he podido usar y puedo decir que es muy potente y fácil de aprender.

Ventajas	Inconvenientes
- Fácil de aprender. - Un lenguaje que se actualiza con el paso del tiempo. - Muchos *Frameworks*: Laravel, Symfony, Zend, etc. - Gran demanda en el mercado laboral. - Compatibilidad con la mayoría de hostings. - Tiene una gran comunidad.	- No compila los archivos por lo que se vuelve lento. - Muchas líneas de código comparado con Python - En el mercado laboral existe mucha gente y con mayor experiencia.

2.3. Python [5]

Como muy bien describe Wikipedia, es un lenguaje de programación interpretado (su código no necesita ser interpretado por un compilador) cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Tras mi experiencia trabajando en el proyecto, puedo afirmar que el código debe estar bien estructurado y tabulado correctamente ya que si no es así no funcionará reportando errores en el código fuente. Este es un error bastante típico (que yo mismo sufrí) a la hora de comenzar con python. Mi recomendación es que se empleen tabulaciones y no espacios a la hora de estructurar el código.

```python
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '    %s [label="%s' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for  in n, childenumerate(ast[1:]):
            children.append(dotwrite(child))
        print ,'    %s -> {' % nodename
        for  in :namechildren
            print '%s' % name,
```

Figura 2: Código Python ejemplo

Ventajas	Inconvenientes
- Rápido para programar. - Profesionales mejor pagados. - No se necesita un servidor web externo. - Gestor interno de paquetes. - Tiene una gran comunidad.	- Pocos hostings compatibles. - Se necesita un *framework* para trabajar o por lo contrario se hace difícil. - Difícil de realizar un despliegue en un servidor virtual.

2.4. Web2py [6]

Es un *framework* de código abierto para un ágil desarrollo de aplicaciones web dinámicas y seguras basadas en bases de datos.

Se puede comprobar la facilidad que brinda al desarrollador con una comparativa de código con PHP.

```php
<html><body><h1>Registros</h1><?
  mysql_connect(localhost, usuario, clave);
  @mysql_select_db(database) or die( "Imposible recuperar datos");
  $consulta="SELECT * FROM contactos";
  $resultado=mysql_query($consulta);
  mysql_close();
  $i=0;
  while ($i < mysql_numrows($resultado)) {
    $nombre=mysql_result($resultado, $i, "nombre");
    $telefono=mysql_result($resultado, $i, "telefono");
    echo "<b>$nombre</b><br>Teléfono:$telefono<br /><br /><hr /><br />";
    $i++;
  }
?></body></html>
```

Figura 3: Código html y php de ejemplo

En el código anterior, se puede ver un ejemplo de código PHP el cual recupera desde una BBDD información para mostrarla en una página HTML. El código está integrado en el HTML por lo que la corrección de errores y lectura se vuelve difícil con el crecimiento del código.

La funcionalidad del código se puede escribir en web2py con solamente dos líneas de código almacenadas en su controlador.

```
def index():
    return HTML(BODY(H1('Registros'), db().select(db.contactos.ALL)))
```

Figura 4: Código Python ejemplo

Fue diseñado para guiar al desarrollador para que emplee buenas prácticas a la hora de programar como por ejemplo el uso del patrón Modelo Vista Controlador (MVC).

Su núcleo de librerías, incluyendo la Capa de Abstracción de la Base de Datos, el lenguaje de plantillas, y el conjunto completo de ayudantes pesan 1.4MB. El código fuente completo incluyendo aplicaciones de ejemplo e imágenes pesa 10.4MB por lo que podemos decir que no ocupa apenas espacio en el servidor.

Este separa la representación de los datos (el modelo) de la representación de los datos (la vista) y de los algoritmos y flujo de operación (el controlador).

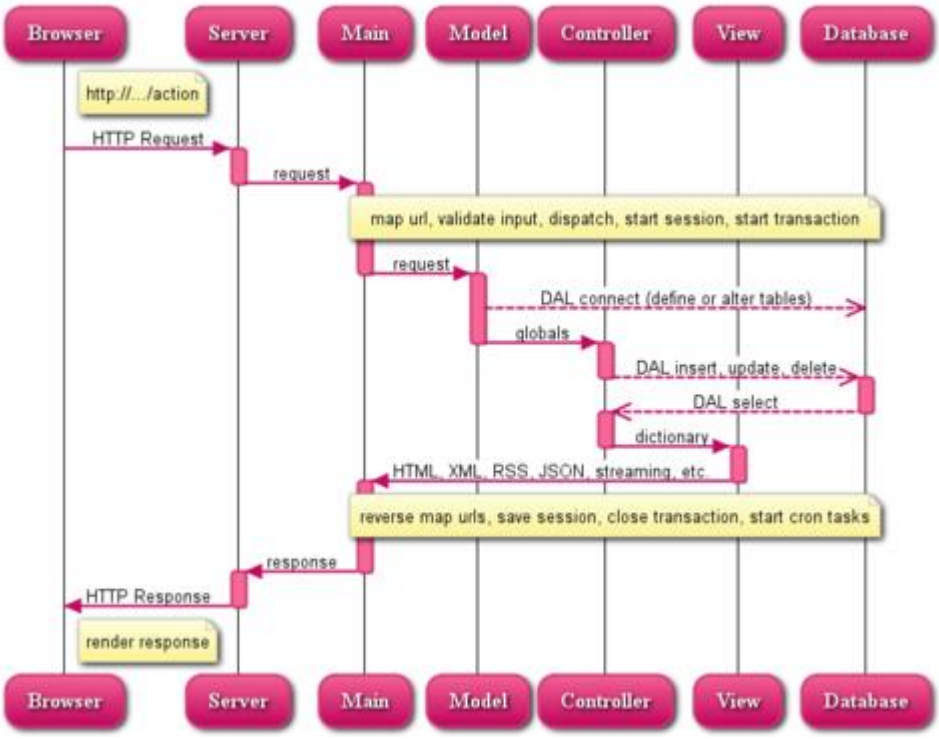

Figura 5: Flujo de trabajo de una solicitud en web2py

Las flechas entrecortadas representan la comunicación con el motor de la base de datos (o los motores). Las consultas a la base de datos se pueden escribir en SQL puro (no está recomendado pero no indica que no se deba hacer) o usando la Capa de Abstracción de la Base de Datos (la forma recomendada), de manera que el código de las aplicaciones no depende de una base de datos específica.

Los componentes del Modelo Vista Controlador son lo que conforman la aplicación del usuario.

2.4.1. Modelo

Este es la representación de la información y su tratamiento. La información tratada es enviada al controlador. En el proyecto, el archivo "triatlón/models/db.py" es el modelo, en este se definen la conexión con la base de datos, tablas que va a tener y valores por defecto. Algunas reglas de visualización como la introducción de valores por defecto también están definidos en dicho modelo.

La siguiente imagen muestra el código que conecta con la base de datos

```
56  if not request.env.web2py_runtime_gae:
57      db = DAL("mysql://root@localhost:3306/triatlon", pool_size=1, migrate_enabled=True, db_codec='utf-8')
```

Figura 6: Conexión con la base de datos almacenada en una variable

El siguiente código muestra la definición de una tabla en la que se tienen varias referencias. Miembro, temporada y entrenador están haciendo referencia al identificador de otra tabla creada anteriormente a esta. La referencia es el parámetro que se puede ver tras el nombre del campo (en el caso de la temporada es db.temporada).

```
334 db.define_table("participa",
335         Field("miembro",    db.auth_user,       notnull=True),
336         Field("temporada",  db.temporada,       notnull=True),
337         Field("entrenador", db.auth_user,       default=None),
338         firma_creacion_modificacion,
339         singular="Participa", plural="Participan")
```

Figura 7: Definición de una tabla

No importa si la base de datos que se ha empleado se reemplaza por MySQL, PostgreSQL, MSSQL, FireBird, Oracle, DB2, Informix, Interbase, Ingres, o Google App Engine (tanto para SQL como para NoSQL) al estar usando la Capa de Abstracción de la Base de Datos la aplicación no depende de una base de datos específica, este la traduce a un idioma que pueda entender la base de datos final.

Una vez que se ha definido y creado una tabla, web2py además genera una interfaz de administración de la base de datos completamente funcional, llamada appadmin, para poder acceder a esa base de datos y a sus tablas.

2.4.2. Controlador

En este se definen las funciones de la página. Uno de los posibles controladores es "default.py". En web2py las URL se asocian a módulos y llamadas a funciones de sus controladores.

Poniendo un ejemplo de url:

http://triatlon:1234/boletin/form/5

- triatlon: es el nombre de la aplicación
- 1234: hace referencia al puerto
- boletin: es el nombre del controlador al cual se va a acceder, en la carpeta controllers debe existir un boletin.py pero no es necesario que exista una vista ya que esta puede ser generada como en el ejemplo de la figura 4 anterior.
- form: función específica que se encuentra en su controlador. Dentro del controlador tiene que existir dicha función definida.
- 5: argumento que contiene el ID único que corresponde a un boletín almacenado en la BBDD.

```
def form():
    registro = db.boletin(request.args(0))
    formulario = SQLFORM(db.boletin, registro)
    formulario.add_button('Volver', URL(request.controller, 'index'))

    if formulario.process().accepted:
        almacenar_notificacion('Registrado', 'Se ha registrado correctamente')
        redirect(URL(request.controller,"index"))

    elif formulario.errors:
        almacenar_notificacion('Error', 'El formulario tiene errores')
    else:
        almacenar_notificacion('Nuevo', 'Por favor complete el formulario')

    return dict(formulario=formulario)
```

Figura 8: Código ejemplo de una función definida en el controlador boletin.py

2.4.3. Vista

Es la que se encarga de estructurar los datos recibidos por el controlador para aplicar estilos y estructuras que faciliten la visualización y tratamiento de los datos por parte de un usuario final.

De igual forma que existe un controlador, tiene que existir una vista que tenga el mismo nombre con la terminación html si esta no es generada y devuelta en su controlador.

```
{{extend 'layout.html'}}
<h1>Formulario boletines</h1>
{{=formulario}}
```

Figura 9: Ejemplo código vista de formulario para los boletines

2.4.4. Seguridad

Según la lista de vulnerabilidades de aplicaciones de software publicadas por el proyecto de Seguridad de Aplicaciones Web Abiertas (OWASP), web2py aborda una lista de las vulnerabilidades que son:

- **Cross Site Scripting (XSS):** Evita que se ejecuten scripts maliciosos en la página del usuario. Web2py, por defecto, "escapa" todas las variables procesadas en la vista.

- **Vulnerabilidades de inyección:** Las inyecciones ocurren cuando información provista por el usuario es enviada como parte de una instrucción o consulta. La información hostil del atacante engaña al intérprete para que ejecute comandos no esperados o para que modifique información. Web2py incluye una Capa de Abstracción de la Base de Datos que hace imposible la inyección de SQL.

- **Ejecución de archivos maliciosos:** Sólo se muestran las acciones, no se permite ejecutar funciones diferentes a las mostradas, así se evita que se ejecute archivos maliciosos.

- **Referencia directa a objetos:** Web2py no expone objetos internos de la implementación y valida las URL.

- **Fugas de información y manejo de errores inapropiado:** No se muestra al usuario ninguna información relevante a los errores.

- **Administración de sesión y autentificación rota:** Web2py maneja las sesiones activas y cookies, lo que evita que los desarrolladores cometan errores al tener que realizar dicho manejo por su cuenta.

- **Almacenamiento criptográfico inseguro:** Web2py puede encriptar los datos usando MD5 o HMAC la + SHA-512 para proteger los datos como las contraseñas almacenadas.

- **Falla en restringir acceso URL:** El control de acceso basado en roles evita que intrusos accedan a direcciones URL directamente, además que se puede elegir la visibilidad de las funciones por usuarios o sesiones.

3. Metodología

Antes de comenzar con un proyecto, se tienen que seguir una serie de pautas. Siguiendo las enseñanzas que se cursan en la asignatura Directiva Centrada en el Usuario (DCU) se deben seguir una serie de pasos mostrados en la siguiente imagen.

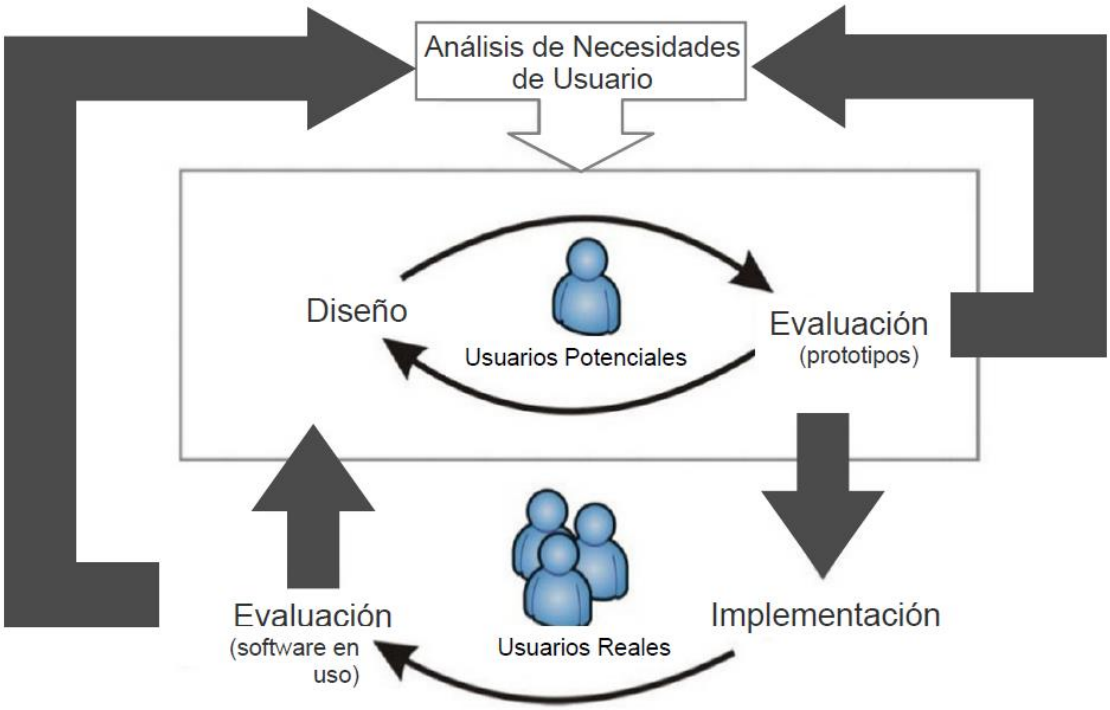

Figura 10: Proceso del desarrollo centrado en el usuario

En la figura anterior, se ve el esquema del proceso del DCU, el cual se procederá a explicar sus distintos pasos aplicados en el proyecto.

3.1. Análisis de necesidades del usuario

Se deben identificar los usuarios a los cuales va dirigida la aplicación, para qué va a ser usado y las condiciones con el fin de determinar los objetivos para satisfacer las necesidades. Pueden realizarse mediante entrevistas, cuestionarios, lluvia de ideas, etc.

El estudio de necesidades ya estaba realizado desde un primer momento por la tutora la cual ya indica para el proyecto que secciones quiere en la aplicación para cada tipo de usuario y las necesidades de estos. Es una aplicación para un grupo muy reducido y especializado de usuarios los cuales ya tienen una plataforma con la cual trabajar, pero se desea realizar una migración a otra más eficiente que cumpla con las necesidades del equipo.

3.2. Diseño y evaluación

Se plantea en el proyecto que la aplicación sea sencilla, de un fácil uso para usuarios menos experimentados en el sector tecnológico y autoadaptable independientemente del dispositivo y la resolución.

La parte centrada en el diseño es muy importante ya que es lo primero que va a ver el usuario al entrar en la aplicación y la cual le ofrecerá una experiencia agradable o de rechazo, por lo que se debe cuidar con detalle las imágenes, disposición y colores.

Bien, partiendo con los datos que tenemos, sabemos que va a ser para deportistas de la UPV la cual va a ser gente joven con edad entre los 20 y 35 años que no tienen amplias nociones en el manejo de nuevas herramientas informáticas por lo que se tendrá que facilitar la información importante lo máximo posible sin abordar al usuario con información o imágenes innecesarias para que no pierda el interés.

Se crearon 3 interfaces únicamente con funcionalidad de navegación como prototipo (opté por este método el cual pese a ser más costoso ya que se tienen que crear las interfaces con código en lugar de dibujarlas en papel por poner un ejemplo, resulta más atractivo para el usuario que la está probando y se asemeja más a la funcionalidad que se pretende conseguir) para estudiar su respuesta con 10 usuarios que cumplían los requisitos de no tener avanzadas nociones informáticas y estar en el rango de edad. Se les asignaron una serie de tareas que debían realizar para estudiar su respuesta por la aplicación:

- Crear una cuenta de usuario
- Acceder a las noticias
- Descargar un entrenamiento
- Modificar su perfil de usuario
- Cerrar la sesión

Una de las interfaces se descartó por contener demasiados pasos para realizar las tareas solicitadas ya que los usuarios navegaron entre los menús demasiado tiempo, esta constaba de un menú superior que se escondía al deslizar la página para que ocupase todo el contenido.

La cantidad de páginas visitadas hasta llegar al destino fueron apuntadas y se realizó una media entre los 10 usuarios.

- Crear una cuenta de usuario = 1 página
- Acceder a las noticias = 2.1 páginas
- Descargar un entrenamiento = 2.5 páginas
- Modificar su perfil de usuario = 3.6 páginas
- Cerrar la sesión = 1.3 páginas

Entre las dos interfaces restantes mediante valoraciones de los usuarios y velocidad en la realización de las tareas que se pusieron a prueba, destacó una. La cual tenía una media cuasi perfecta.

- Crear una cuenta de usuario = 1 página

- Acceder a las noticias = 1 páginas
- Descargar un entrenamiento = 1 página
- Modificar su perfil de usuario = 1.4 páginas
- Cerrar la sesión = 1 página

Al finalizar la prueba se hizo una breve encuesta a los usuarios en la cual se preguntó si han visto intuitiva la aplicación y que cambios les gustaría que se aplicasen. No dieron ninguna respuesta relacionada con el diseño, por lo que como última instancia fue presentada a la tutora para que diese el visto bueno.

La tutora aprobó el diseño de la aplicación al ser intuitivo, limpio y elegante.

La interfaz resultante dispone de un menú lateral izquierdo para la navegación entre las ventanas sin submenús. El contenido de las páginas

3.3. Implementación y evaluación

Es la fase en la que se desarrolla toda la aplicación con los detalles y funcionalidades. Posteriormente pasará a ser probada por usuarios reales para comprobar su correcto funcionamiento en un entorno real.

Tras las pruebas, se produce la evaluación del proyecto. Si no se han presentado problemas durante las pruebas, la aplicación se puede catalogar como completada, pero si por el contrario produce fallos y no es del gusto del cliente, se deberá volver al paso de evaluación y diseño para detectar que ha fallado y posibles maneras de corregirlo.

4. Preparación del entorno de trabajo

4.1. Python

Como se ha mencionado anteriormente, web2py hace uso del lenguaje de programación Python para el desarrollo. Por ello, se ha de descargar una versión de Python desde la web oficial (preferiblemente) para instalarla en el equipo. La versión empleada en este proyecto ha sido la 3.6.0 con fecha de salida 23/12/2016. La instalación es tan sencilla como seguir los pasos que va mostrando el instalador.

4.2. XAMPP

La versión empleada en el proyecto ha sido la 3.2.2. Una vez descargada desde la página web oficial, se procede a su instalación.

La instalación es sencilla y guiada, esta contiene una serie de paquetes establecidos en la cual se pueden desmarcar todas las opciones que no se van a emplear en el proyecto ya que solo necesitamos phpMyAdmin (para la gestión de la BBDD), Apache (como servidor para phpMyAdmin), MySQL (como base de datos). Los pasos de instalación son tan sencillos como presionar en siguiente y elegir la ruta de instalación.

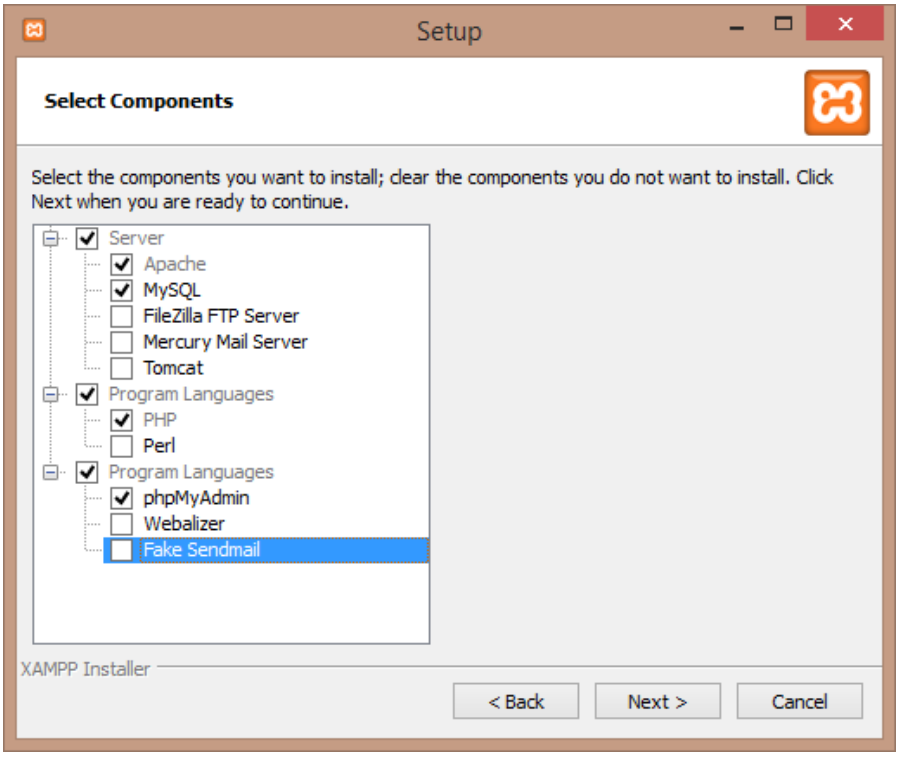

Figura 11: Instalación de XAMPP

Apache crea una ruta por defecto en la que contiene las aplicaciones web (puede ser cambiada por el fichero de configuración) llamada “htdocs”, no es necesario meter el proyecto en dicha ruta ya que como bien se ha mencionado anteriormente solo lo usaremos para utilizar la interfaz que nos proporciona para la gestión de la base de datos.

Una vez esté instalado, mediante phpMyAdmin se ha de crear una base de datos para el proyecto con codificación utf8_general_ci. Esta codificación tiene una gran precisión a la hora de ordenar correctamente con Unicode en muchos idiomas comunes, es rápido en comparaciones, ordenaciones,... y puesto que solo vamos a emplear el castellano es perfecto.

No es necesario definir las tablas ya que estas serán definidas dentro del proyecto en el fichero applications\triatlon\models\db.py.

Figura 12: Entorno phpMyAdmin para administrar la BBDD

4.3. Web2py

Este es el servidor que va a albergar nuestra aplicación web. La instalación es tan simple como descargarse y descomprimir un archivo desde la web oficial (no tiene instalación). Cuando esté descomprimido, la carpeta resultante contiene un ejecutable llamado web2py el cual inicia el servidor.

Al arrancar el servidor, se debe de seleccionar una IP de la lista que muestra. Para trabajar en local ha de ser seleccionada la primera opción, se debe de introducir un número de puerto con el cual acceder y finalmente una contraseña para acceder a secciones más elevadas de administración en el propio web2py desde donde se puede acceder a la administración de todos los proyectos que se tienen alojados por lo que si no se trabaja en local, es recomendable el uso de una contraseña segura.

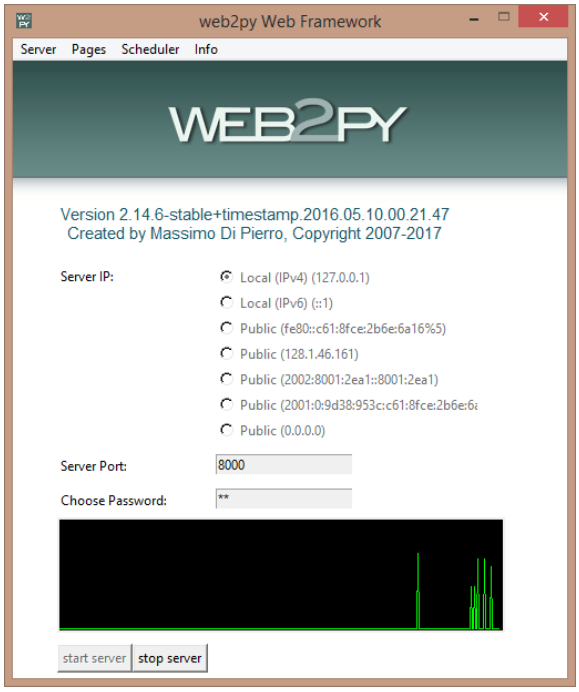

Figura 13: Servidor web2py en funcionamiento recibiendo peticiones

Web2py proporciona una interfaz administrativa muy completa accesible mediante la contraseña anteriormente definida. En ella se puede navegar entre los distintos proyectos que se tienen creados (automáticamente se crean 3 proyectos: *admin*, *examples* y *welcome*), solo falta crear un nuevo proyecto para comenzar con la implementación.

Este proyecto se ha definido con el nombre de triatlon.

Figura 14: Creación de una nueva aplicación.

4.4. Entorno de programación

Como entorno de trabajo para programar se puede utilizar cualquier plataforma en la que el programador se encuentre cómodo. Algunas ofrecerán más beneficios dependiendo el lenguaje que se emplee. En mi caso he optado por emplear Aptana Studio[7]. Es un entorno de desarrollo integrado de software libre basado en eclipse y desarrollado por Aptana Inc., puede funcionar bajo Linux, Windows o Mac proveyendo soporte para lenguajes como: PHP, Python, Ruby, CSS, Ajax, HTML y Adobe AIR. Con la descripción citada, se puede observar que está muy orientado para los lenguajes de programación web por lo que es una buena elección para este trabajo.

Otra opción de entorno podría ser Sublime Text[8], trata de otro entorno de programación gratuito y multiplataforma al igual que Aptana. Sublime también cuenta con una versión de pago cuya única diferencia, es que desactiva un “popup” que aparece en la versión gratuita al realizar un número determinado de guardados. Existen muchos entornos para el desarrollo de aplicaciones web, pero cada cual puede elegir el que quiera.

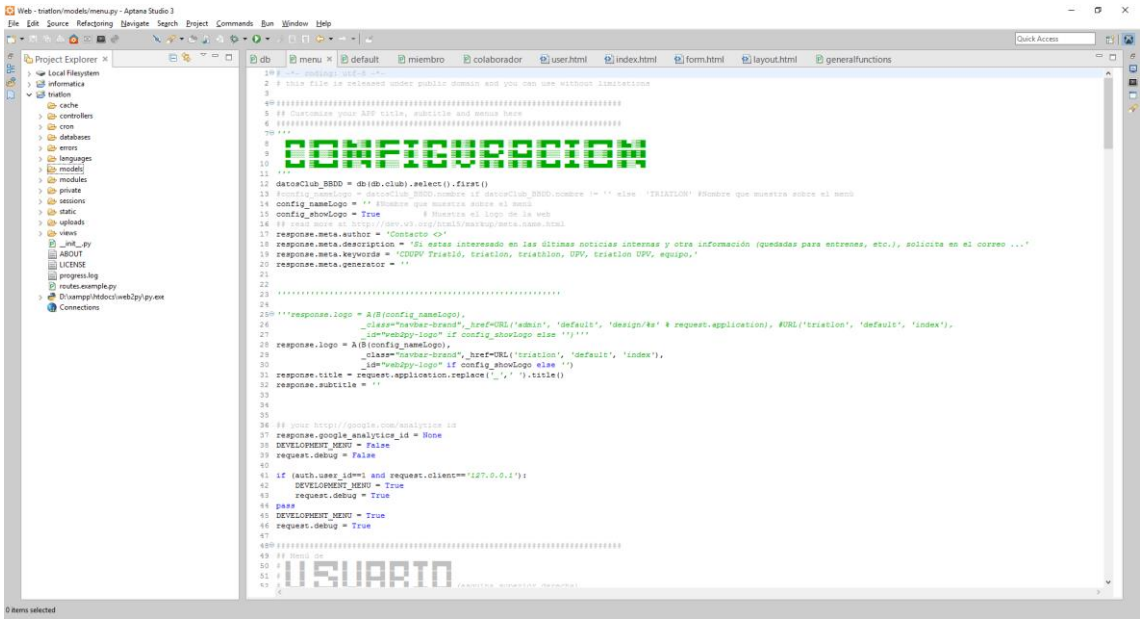

Figura 15: Entorno de desarrollo Aptana Studio

5. Implementación

5.1. Modelo (Base de datos)

A la hora de crear la BBDD, se ha empleado una aplicación web llamada “WWW SQL Designer” para hacer el diseño de esta. Este ofrece una interfaz muy simple pero práctica

para su diseño. Una vez realizado un diseño, este se puede guardar en el propio explorador usando las cookies o ser exportado a XML, lo que ha sido una buena funcionalidad para este proyecto ya que podíamos mantener contacto por correo mi tutora y yo enviándonos modificaciones de la BBDD de una manera rápida y fácil y visual.

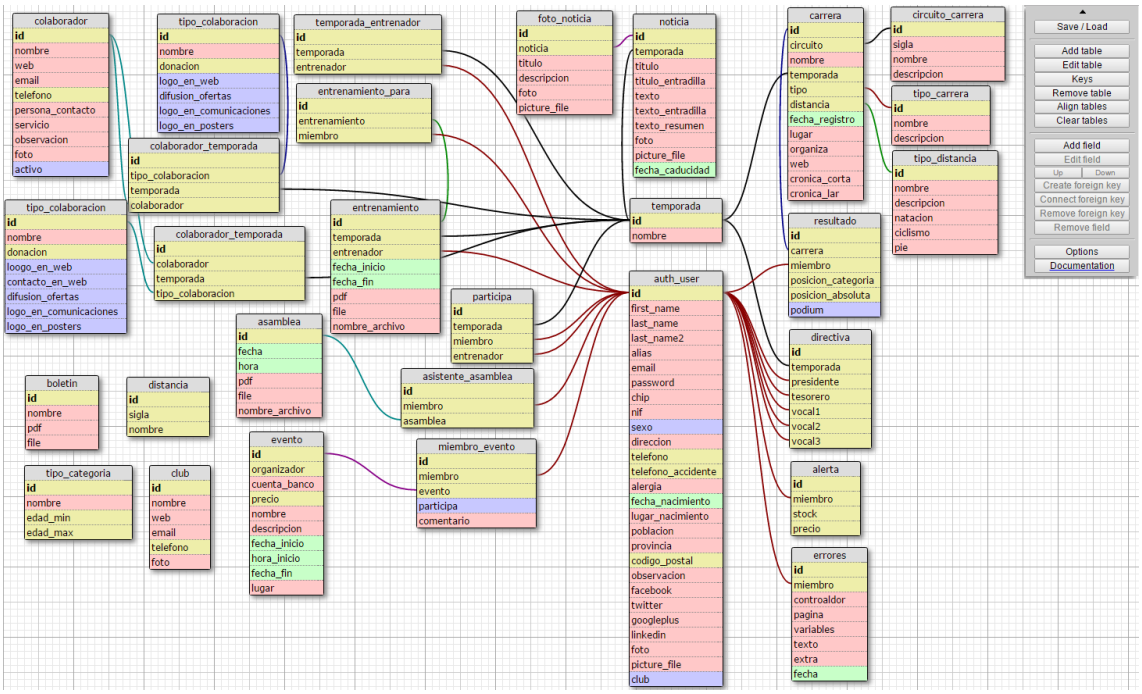

Figura 16: Base de datos con sus relaciones en SQL Designer

La base de datos crecerá en un futuro ya que será un nuevo trabajo final de grado para otro alumno que se ocupará de añadir nuevas funcionalidades como una sección de stock y permitir la compra de equipamiento por lo que recomiendo el uso de esta aplicación.

5.1.1. Código (db.py)

Lo primero de todo es realizar la conexión con la base de datos, recordemos que la hemos llamado “triatlon” y es una base de datos de MySQL.

5.1.1.1. Conexión con la BBDD

```
db = DAL("mysql://root@localhost:3306/triatlon",
pool_size=1, migrate_enabled=True, db_codec='utf-8')
```

Figura 17: Variable para la conexión con la BBDD

El objeto DAL o Capa de Abstracción de la Base de Datos, representa una conexión con la base de datos, esta conexión será almacena en la variable “db”.

Por defecto web2py trabaja con la codificación de caracteres UTF8. Si se desea trabajar con otra codificación se debe indicar en este punto.

5.1.1.2. Creación de tablas

Web2py trabaja con una serie de tablas con el prefijo "auth". Esta serie de tablas, almacenan una serie de datos personales y permisos relacionados con los usuarios.

- **auth_user:** almacena el nombre del usuario, dirección de correo electrónico, contraseña y estado (pendiente de registro, aceptado, bloqueado)
- **auth_group:** almacena los grupos o roles para usuarios en una estructura muchos-a-muchos. Por defecto, cada usuario pertenece a su propio grupo, pero un usuario puede estar incluido en múltiples grupos, y cada grupo contener múltiples usuarios. Un grupo es identificado por su rol y descripción.
- **auth_membership:** enlaza usuarios con grupos en una estructura muchos-a-muchos.
- **auth_permission:** enlaza grupos con permisos. Un permiso se identifica por un nombre y opcionalmente, una tabla y un registro. Por ejemplo, los miembros de cierto grupo pueden tener permisos de actualización para un registro específico de una tabla determinada.
- **auth_event:** registra los cambios en las otras tablas y el acceso otorgado a través de CRUD a objetos controlados con RBAC.
- **auth_cas:** se usa para el Servicio Central de Autenticación (CAS). Cada aplicación web2py es un proveedor de CAS y puede opcionalmente consumir el servicio CAS. En este proyecto no es empleado.

En este proyecto se hace uso para el registro de los usuarios la tabla "auth_group" pero ha sido personalizada para añadirle nuevos campos a los que crea por defecto.

```
auth.settings.extra_fields['auth_user']= [
    Field("last_name2",   "string",  default=None,  length = 25,   label=T("Last
surname")),
    Field("nif",        "string",  default=None,  length = 9,    label = T("DNI")),
    Field("chip",       "string",  default=None,  length = 9,    label=T("Chip")),
    Field("sexo",       "boolean", default=False,             label=""),
    Field("direccion",   "string",  default=None,  length = 150, label=T("Address")),
    Field("telefono",    "integer",  default=None,  length = 12,  label = T("Phone")),
    Field("telefono_accidente","integer",default=None,  length = 12,  label = "Tel.
Accidente"),
    Field("alergia",   "text",    default=None,  length = 250, label="Alergias",
      widget = SQLFORM.widgets.text.widget, readable=False),
    Field("fecha_nacimiento","date",   default=None,          label=T("Birth date"),
      requires = IS_DATE(format=("%d-%m-%Y"))),
    Field("lugar_nacimiento","string",  default=None,  length = 50,  label = T("Place
of birth")),
    Field("poblacion",    "string",  default=None,  length = 50,  label=T("Town")),
    Field("provincia",    "string",  default=None,  length = 50,  label=T("Province")),
    Field("codigo_postal", "string",  default=None,  length = 50,  label = T("Postal
code")),
    Field("observacion",   "text",    default=None,  length = 250,
label=T("Observations"),
```

```
        widget = SQLFORM.widgets.text.widget, readable=False),
    Field("facebook",     "string",  default=None,  length = 50),
    Field("twitter",      "string",  default=None,  length = 50),
    Field("googleplus",   "string",  default=None,  length = 50),
    Field("linkedin",     "string",  default=None,  length = 50),
    Field("foto",         "upload",
        uploadfolder=request.folder+"uploads/fotos/usuarios", autodelete=True,
uploadfield=True,
        requires          = IS_EMPTY_OR(IS_IMAGE(extensions=('jpeg', 'png', 'JPEG',
'PNG', 'jpg', 'JPG')))),
    Field("picture_file",  "blob"),
    Field("club",         "boolean", default=False, label="Activo"),
    Field("talla_sudadera", "integer", default=0, length = 1, requires =
IS_IN_SET(varTallas,      zero=None)), #zero="< "+T("Choose one")+" >")),
    Field("talla_camiseta", "integer", default=0, length = 1, requires =
IS_IN_SET(varTallas,      zero=None)),
    Field("talla_pantalon", "integer", default=0, length = 1, requires =
IS_IN_SET(varTallas,      zero=None)),
    Field("talla_calzado", "integer", default=0, length = 2, requires =
IS_IN_SET(varTallasCalzado, zero=None)),
    ]
```

Figura 18: Extensión de la tabla creada por defecto que contiene datos de usuario.

Para que los usuarios puedan tener un alias con el que identificarse y poder acceder a la aplicación, se puede definir de la siguiente manera:

```
#username = True          - Permite logear con el nombre de usuario
auth.define_tables(username=True, signature=False)
```

Figura 19: Configuración en la definición de tablas.

El club tiene una cuenta de correo Gmail que se empleará para enviar correos a los usuarios para validar su cuenta de correo con la que se han registrado en el servidor o que deseen recuperar su identificador y/o contraseña en el caso de que no la recuerden. Este les enviará un correo a su cuenta con la que se han registrado con una serie de indicaciones que contienen un acceso directo para realizar la acción solicitada.

El servidor de correo que se ha empleado funciona por el protocolo smtp y su configuración con web2py es sencilla.

Antes de nada, hay que importar la librería de correo.

```
from gluon.tools import Auth, Ser... Mail
```

En este caso de ha decidido utilizar Gmail pero se puede emplear cualquier otro servidor de correo. Véase también que se puede modificar el mensaje que se envía.

La configuración de los parámetros es la siguiente:

```
# -------------------------------------------------------------------------
```

```
# Configuración del correo
# ---------------------------------------------------------------------
mail = auth.settings.mailer
mail.settings.server    = 'smtp.gmail.com:465' or 'logging' if request.is_local else
myconf.get('smtp.server')
mail.settings.sender    = 'triatlonupv.pruebas@gmail.com'
mail.settings.login     = 'triatlonupv.pruebas@gmail.com:**********'

mail.settings.tls = myconf.get('smtp.tls') or False
mail.settings.ssl = myconf.get('smtp.ssl') or False

auth.messages.verify_email = 'Antes que nada, es necesdaria la verificación del
correo. \n\ Entra en el link %(link)s'
auth.messages.reset_password = 'Entra en el link %(link)s para reiniciar la
contraseña'
```

Figura 20: Configuración para el correo.

Para tener una seguridad extra y un control a la hora de que se registren datos en un formulario para ser almacenados en la base de datos, se me ocurrió el crear una serie de registros que almacenasen el identificador de usuario y a qué hora ha creado un registro, los cuales, no serán visibles en la web ya que se indica que van a ser ocultos y no modificables.

Aparte de estos dos campos, un campo también puede ser modificado por lo que aparte de los citados anteriormente, se definen otros dos campos similares que almacenen que usuario ha realizado la modificación y la hora.

Esto crea una tabla, la cual no se almacena en la base de datos y es guardada en una variable para poder ser insertada en la creación de nuevas tablas.

```
firma_creacion_modificacion = db.Table(db, 'firma_creacion_modificacion',
   Field('created_on', 'datetime',   default=request.now,   writable=False,
readable=False),
   Field('created_by',  db.auth_user, default=auth.user_id,  writable=False,
readable=False),
   Field('modified_on', 'datetime',   update=request.now,    writable=False,
readable=False),
   Field('modified_by', db.auth_user, update=auth.user_id,   writable=False,
readable=False))
```

Figura 21: Tabla que define unos campos para añadirlos en las tablas deseadas.

En el siguiente ejemplo de una tabla real en el proyecto, se puede ver que se define el nombre de la tabla, los campos y luego se añade la variable anteriormente creada que contiene los datos extra de control. El campo “format” indica la cadena que se va a mostrar a la hora de referenciar tabla para mostrar un campo. En este caso, cuando se

muestre en un desplegable una temporada mediante un ID, esta mostrará el nombre de dicha temporada lo que posteriormente ahorrará muchas líneas de código cuando se desee crear el desplegable y obtener los valores.

```
db.define_table("temporada",
    Field("nombre", notnull=True, default=None),
    firma_creacion_modificacion,
    singular="Temporada", plural="Temporadas",
    format='%(nombre)s')
```

Figura 22: Ejemplo de código para definir una tabla.

Antes de comenzar a trabajar, al finalizar de crear las tablas automáticamente en la base de datos, se necesitan ciertos valores que estén ya por defecto en las tablas. Se hace una simple comprobación de que la tabla no contiene datos para poder introducir los valores.

```
if db(db.club).isempty():
    db.club.truncate()
    db.club.insert(
        nombre    = 'CDUPV Triatló',
        web       = 'http://cdupvtriatlo.org/',
        email     = 'cdupv.triatlo@gmail.com'
      )
```

Figura 23: Ejemplo de introducción de datos por defecto.

Por defecto se introducen los datos del club (nombre, web y email), tipos de colaboraciones para el equipo, datos de los distintos tipos de distancias y carreras, categorías según rangos de edades definidos para futuros cálculos de categorías en las carreras, etc. Aparte de todos estos datos, se crea un primer usuario administrador con privilegios de administración total.

Los distintos roles que tiene la aplicación son:

- **Usuario:** Rol por defecto sin privilegios, al crear una cuenta desde la aplicación es el rol asignado automáticamente.
- **Entrenador:** Usuario especial que puede subir entrenamientos y asignarlos a usuarios.
- **Usuario y entrenador:** Junta los permisos de los dos roles anteriores. Puede ser un entrenador y recibir entrenamientos de otro entrenador.
- **Ayudante:** Se ocupa de ayudar al administrador a gestionar la intranet. Tiene ciertos privilegios que le permiten acceder a zonas que los usuarios anteriores no pueden.
- **Administrador:** Tiene control total te la aplicación. Puede crear, acceder y borrar todo lo que desee. A diferencia de los ayudantes, ellos tienen ciertas limitaciones que este no tiene.

5.2. Vistas

Las vistas es la parte en la que los usuarios interactúan con la aplicación. Se ha separado el código que es igual para cada página el cual comprende desde el logo de la web, a los menús y el contenedor que mostrará los contenidos de cada página.

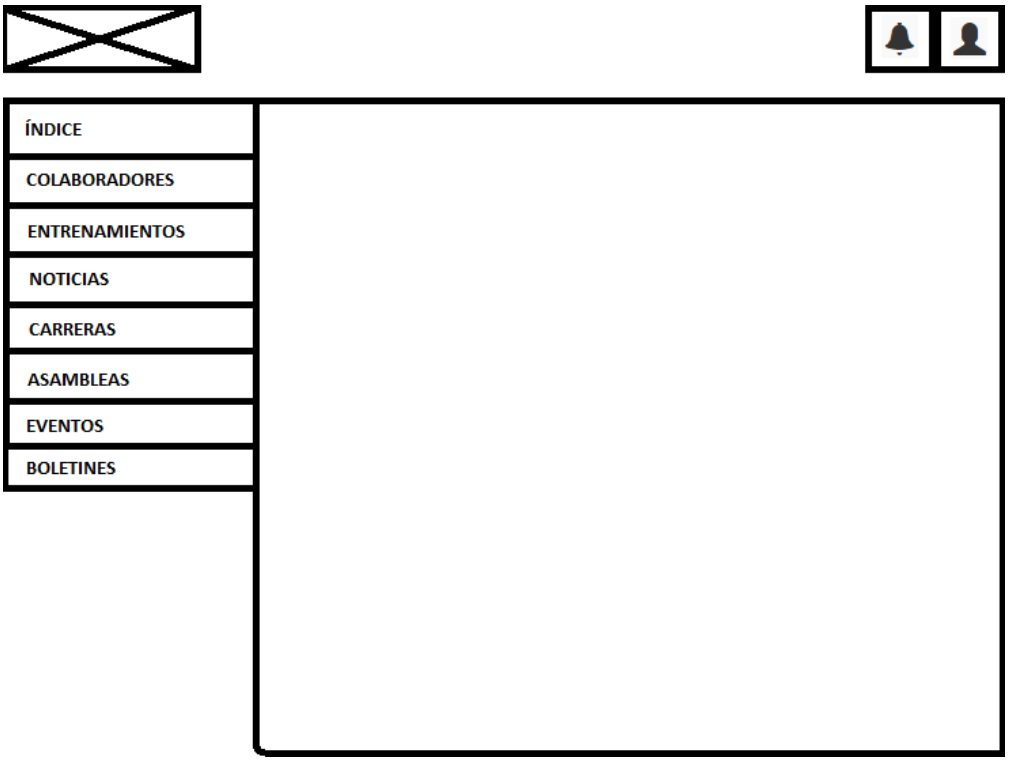

Figura 24: Interfaz cargada en el layout.html vista desde un perfil usuario.

La imagen anterior, representa la parte que comparten todas las ventanas de la aplicación ya sean vistas o formularios. En la parte superior se muestra la imagen del club y a la derecha dos iconos. Uno indica al usuario notificaciones sobre nuevas modificaciones y el otro icono, información del usuario para que pueda acceder a la modificación de su perfil y cierre de sesión.

El menú lateral izquierdo se genera en un archivo Python llamado menú, alojado en la carpeta de los modelos al igual que las alertas anteriormente mencionadas. Dependiendo del usuario registrado, el menú varía para mostrar nuevas funcionalidades, pero las alertas son iguales para todos los usuarios.

Cada página tiene su índice dentro de su carpeta en las vistas, este nombre tiene que coincidir con el nombre del controlador. Para cargar el layout al comenzar el documento se tiene que hacer una llamada para que sea usado como base común.

```
{{extend 'layout-grid.html'}}
```

Figura 25: Código para usar la plantilla web

Las alertas se cargan en el fichero de Python llamado menú.py

```
## Menú de
# ALERTAS
# ALERTAS
# ALERTAS (esquina superior derecha)
################################################

if auth.is_logged_in():
   usuario     = str(session.auth.user.id)
   alertaCont = 0
   alertasDatos = {}
   alertasModulos = {"entrenamiento"      :"entrenamientos",
                     "noticia"            :"noticias",
                     "carrera"            :"carreras",
                     "asamblea"           :"asambleas",
                     "evento"             :"eventos",
                     "boletin"            :"boletines"}
   try:
      alertasBBDD  = db((db.alerta.miembro == usuario) & \
                  (db.alerta.leido == False)).select(orderby=db.alerta.modified_on)
      # Inicializar los datos
      for key, val in alertasModulos.items():
         if (request.controller != key) or (request.function != "index"):
            alertasDatos[key] = [0, val, '']

      for alerta in alertasBBDD:
         fechaModificado = formato_fecha(alerta.modified_on) + " " +
formato_hora2(alerta.modified_on)
         controlador = alerta.controller
         alertasDatos[controlador] = [alertasDatos[controlador][0]+1,
alertasDatos[controlador][1], fechaModificado]

         alertaCont +=1
      pass
   except Exception, e:
      db.errores.insert(texto = e, extra = 'Error al obtener informacion de las alertas
para el menú superior')
   pass
pass
```

Figura 26: Código que obtiene las alertas

1.- Antes de cargar las alertas se comprueba de que sea un usuario logeado por lo que esté autentificado para poder acceder a su sesión de la cual obtendremos su número de identificación.

2.- Se leen desde la base de datos las alertas registradas para los módulos definidos que son los únicos que van a aparecer en las alertas y se guardan en una variable con un número que indica la cantidad de alertas.

Las alertas se muestran con el código que contiene el layout.html el cual es el siguiente:

```
<ul class="nav navbar-top-links navbar-right">
  {{ ''' Estas variables vienen de menu.py en modules '''}}
  {{if usuarioActivo:}}
        {{=LI(
        A(
                I(_class="fa fa-bell fa-fw"+("" if alertaCont >0 else " no-badge")),
                SPAN(alertaCont, _class="badge badge-new-alert") if alertaCont > 0
else SPAN(),
_href="#", _class="dropdown-toggle", **{"_data-toggle" : "dropdown"}),
        UL(
                LI(_style="margin-bottom:10px;"),

                # ALERTAS
                *[
                LI(
                        A(
                                DIV(
                                        #I(_class="fa fa-twitter fa-fw"),
SPAN(alertasDatos.get(alertaRow)[0], _class="badge", _style="margin-right:5px;"),
SPAN(alertasDatos.get(alertaRow)[1]),
SPAN(alertasDatos.get(alertaRow)[2],_class="pull-right text-muted small")
                ),_href=URL(alertaRow, "index")),_style="margin-bottom:10px;") if
alertasDatos.get(alertaRow)[0] > 0 else []
 for alertaRow in alertasDatos] if alertaCont > 0 else (LI(_class="divider"), LI("No hay
alertas",_style="margin-left:10px;"),LI(_class="divider"))
,_class="dropdown-menu dropdown-alerts animated
fadeInDown"),_class="dropdown") if auth.is_logged_in() else ""
pass}}
```

Figura 27: Carga de alertas en el layout.html

La mayoría de las páginas tienen un índice, un formulario y una vista. Todas ellas tienen el código anterior al comienzo del código.

Al final del documento se incluye código javascript correspondiente a su página. El código javascript global el cual es utilizado por todas las páginas se incluye en el layout.html y en web2py_ajax.html.

5.2.1. Index

Es la pantalla principal, el índice principal se compone de lo más relevante para el usuario que consta de una sección donde salen las últimas noticias. Estas aparecen en la parte izquierda ocupando gran parte del contenedor central de la página.

En la parte derecha se muestra una lista de los últimos entrenamientos que se han publicado. Por defecto se muestran tres indicando la fecha y un acceso directo para su descarga, esto facilita al usuario no tener que navegar entre menús para acceder a lo que más le interesa nada más registrarse, facilitándole los datos y ahorrándole tiempo.

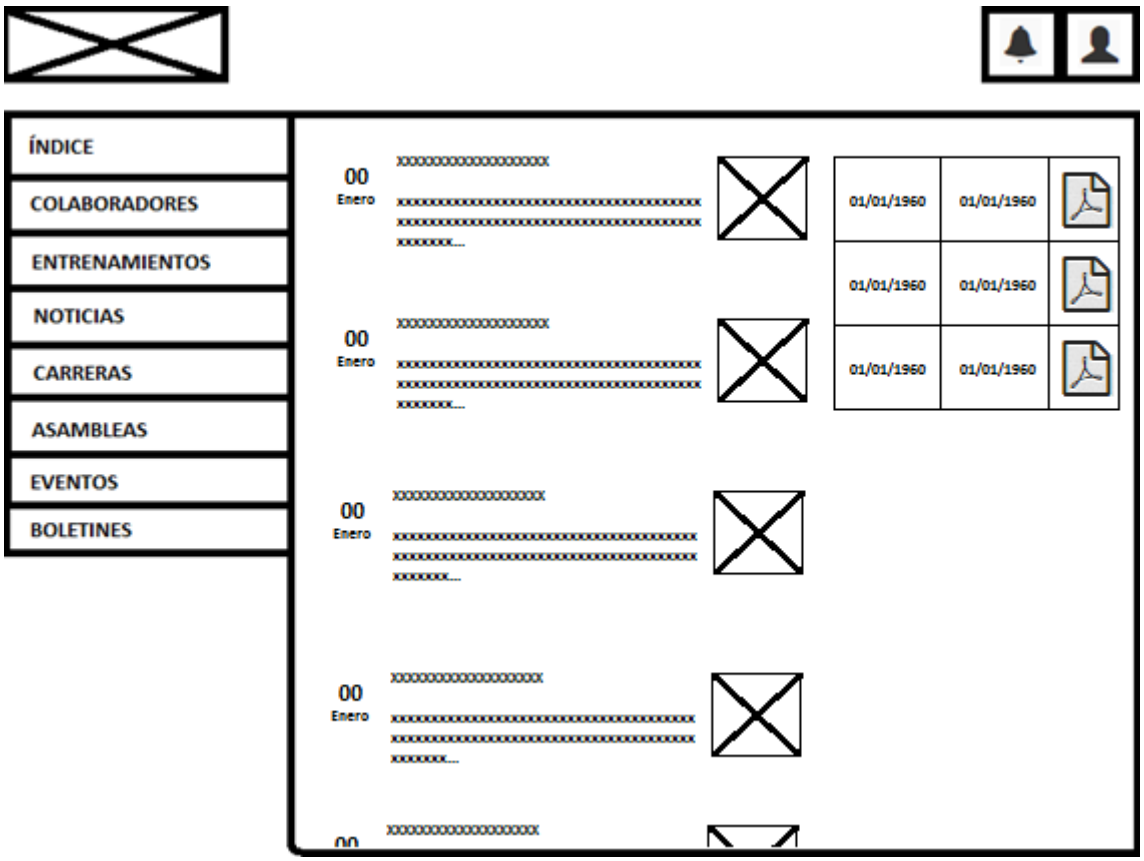

Figura 28: Pantalla principal de la aplicación

El resto de páginas tienen un índice propio. Pongamos como ejemplo la sección de eventos. Esta consta de una tabla que muestra algunos datos almacenados en la base de datos en su tabla correspondiente. Al final de cada registro tiene tres botones con los que se puede ver al detalle el registro, modificar o ser eliminado (dependiendo el rol de usuario pueden aparecer menos botones, ya que por poner un ejemplo, los eventos no pueden ser modificados o borrados por un usuario.

Sobre la tabla de rejilla hay un botón para crear uno nuevo que redirige a su correspondiente formulario que será visto más adelante, un desplegable para mostrar diferentes cantidades de datos en la tabla y un buscador que busca por coincidencias de palabras. La estructura de los índices es similar entre todos ellos.

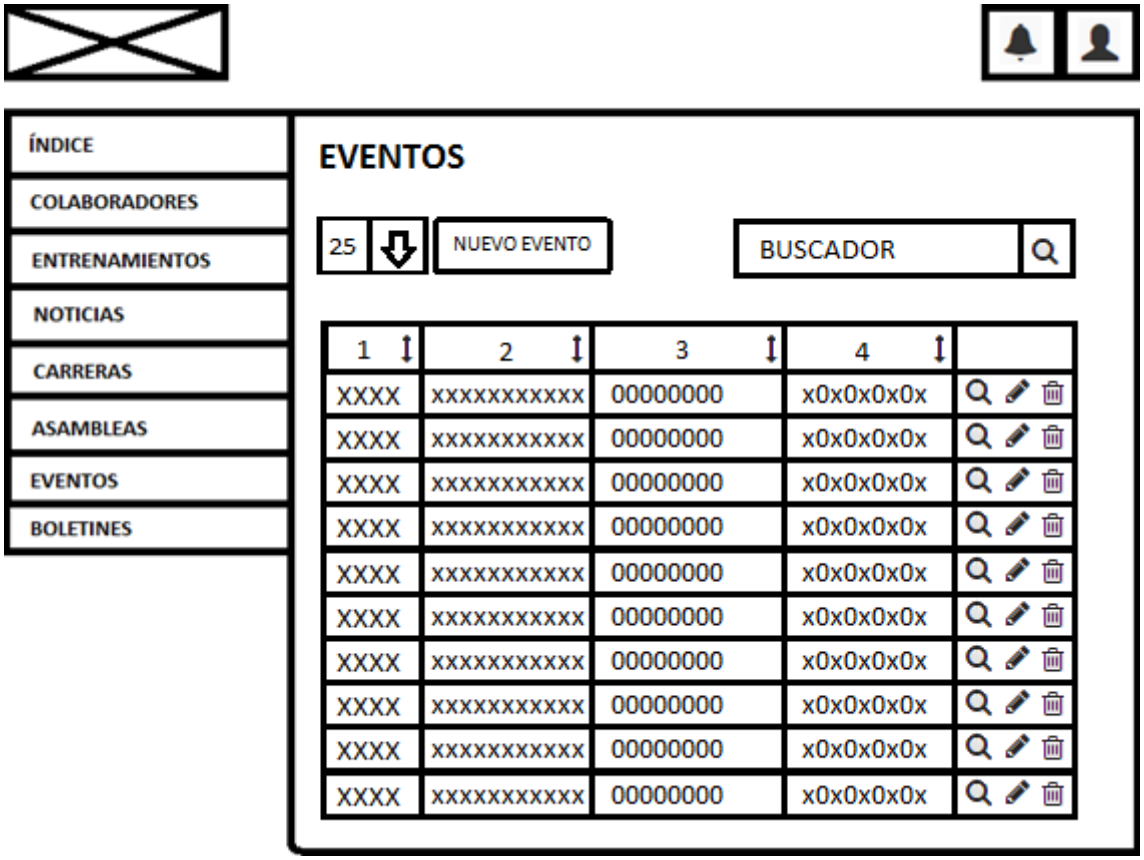

Figura 29: Página index de eventos que muestra el grid.

5.2.2. Formularios

Cada formulario tiene que tener su respectiva función en el controlador y su vista. Los formularios se han llamado “form” y su apariencia es sencilla.

Se dispone de un conjunto de campos en los que introducir los datos.

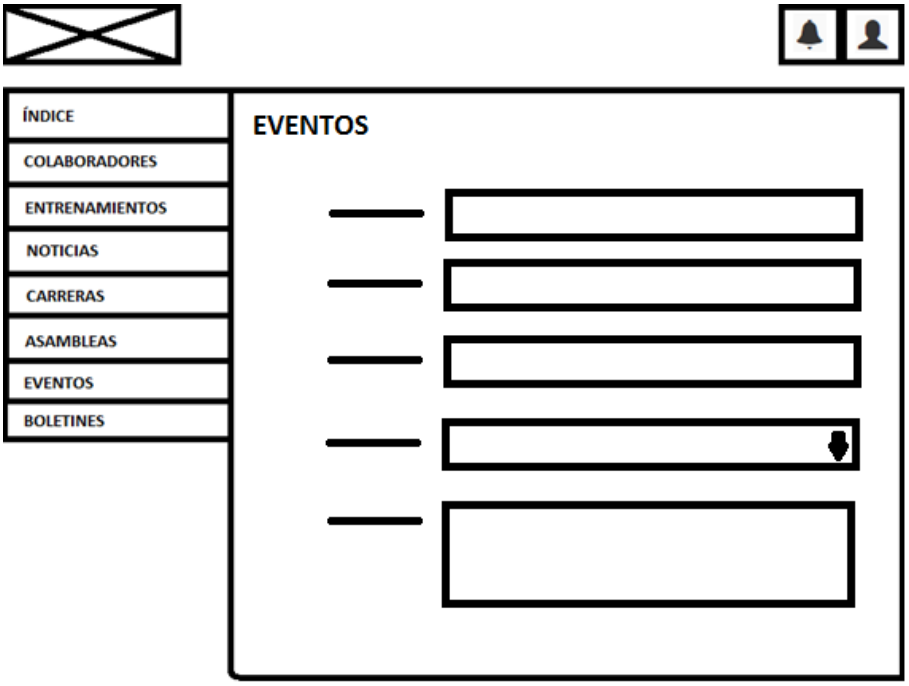

Figura 30: Página form de eventos

5.3. Controlador

La estructura de los controladores suele ser igual para todos excepto para algunas excepciones ya que pueden no tener la misma cantidad de funciones.

```
[Variables globales]
# Es la página por defecto

def index():
        |
        |
        return ...
‘’’
# Código correspondiente al formulario de entrada de datos
‘’’
def form():
        |
        |
        return ...

‘’’’
# Vista de los datos (no está activo en todos ya que esto es un añadido en pruebas que tiene que definirse su funcionalidad por otro alumno en un futuro TFG)
‘’’
def view():
```

```
        |
        |
        return ...
‘’’
# Código para el borrado de datos
‘’’
def destroy():
        |
        |
        return ...
```

Figura 31: Estructura del controlador.

Algunas páginas tienen otras funciones extra ya que necesitan llamadas de jQuery.

Estos controladores tienen llamadas a funciones que son usadas por otros por lo que dichas funciones han sido importadas a un archivo ubicado en la carpeta “models” para que puedan ser utilizadas por todos ellos.

5.4. Plugins

Web2py tiene un gran potencial para la creación de páginas web, pero la apariencia que tiene con la que viene por defecto puede ser mejorada mediante el uso de complementos.

5.4.1. Bootstrap

Es un *framework* incluido con web2py que permite la creación de interfaces web con la particularidad de adaptar la interfaz independientemente al tamaño del dispositivo de visualización, permitiendo la visualización en ordenadores, tabletas o teléfonos móviles. Esto nos permite que la web sea autoadaptable cumpliendo una de las principales condiciones del proyecto.

5.4.2. Autosize [9]

Plugin dedicado a la apariencia, este auto redimensiona el tamaño de los textarea basándose en la cantidad de texto que tiene en su interior. Conforme el texto incrementa, el contenedor responde creciendo también. Este *plugin* solo es para afectar a la apariencia por lo que no aporta funcionalidad extra.

```
// Textos
$('.text').autosize();
```

Figura 32: Plugin Autosize: Ejemplo de uso

5.4.3. Datatables [10]

Plugin para mostrar los datos en formato de tabla, se ha decidido emplear una alternativa a las tablas ofrecidas por web2py ya que este *plugin* simplifica mucho más la información y permite una mayor personalización. El buscador que ofrece, busca entre todos los valores que se encuentran en la tabla pese a que se encuentren ocultos por lo que nos permite poder hacer búsquedas de valores y mostrar menos información que es irrelevante. A términos de usuario final, esta es una gran ventaja ya que no tienen que preocuparse de construir una consulta para las búsquedas.

La invocación de este *plugin* se realiza en el código javascript al final del layout.html

```
$.extend($.fn.dataTable.defaults, {
      searching:  true,           //Buscador
      ordering:   true,           // Permitir la ordenación
      colReorder: true,
      autoWidth:  true,
      paging:     true,           // Habilitar el paginado
      info:       true,           // Informacion del grid
      select:     false,          // Poder seleccionar una fila
      processing: true,
      stateSave:  false,          // Guardar el estado de busqueda
      language: {
                  //"decimal":       ",",
                  "emptyTable":     "No hay datos disponibles",
                  "info":           "Mostrando desde _START_ a _END_",
                  "infoEmpty":      "Mostrando 0",
                  "infoFiltered":   "Total _MAX_",
                  "infoPostFix":    "",
                  "thousands":      ".",      // Separados de miles
                  "lengthMenu":     "Mostrar _MENU_",
                  "loadingRecords": "Cargando...",
                  "processing":     "Procesando...",
                  "search":         "Buscar:",
                  "zeroRecords":    "No se han encontrado coincidencias",
                  "paginate": {
                     "first":      "Primero",
                     "last":       "Último",
                     "next":       "Siguiente",
                     "previous":   "Anterior"
                  },
                  "aria": {
                     "sortAscending":  ": activar para ordenar la columna ascendentemente",
                     "sortDescending": ": activar para ordenar la columna
descendentemente"
                  },
                  select: {
                     rows: {
                        _: "(%d filas seleccionadas)",
```

```
                    0: "",
                    1: "(1 fila seleccionada)"
                }
            }
        },
    "dom":   "<'row'<'col-sm-8 col-xs-8'l><'col-sm-4 col-xs-4'f>>" +
        "<'row'<'col-sm-12'tr>>" +
        "<'row'<'col-sm-4'i><'col-sm-8'p>>"
});
```

Figura 33: Plugin Datatable: configuración.

Figura 34: *Plugin* datatables filtrando por un valor

5.4.4. Switch [11]

Transforma los cuadros de selección para otorgar una apariencia mucho más agradable. En el formulario del perfil de usuario se emplea para la selección de género del usuario.

```
$("[name='sexo']").bootstrapSwitch({
        size:'small',
        onText: "Hombre",
        offText: "Mujer",
        offColor: 'pink'});     // Cambia el color por defecto
```

Figura 35: *Plugin* switch: configuración y vista

5.4.5. Datepicker [12]

Datepicker ofrece un cambio con la forma de seleccionar una fecha ofreciendo un calendario desplegable asociado a un campo en un formulario.

Se puede configurar para que muestre un calendario con fechas u horas.

```
$('.bootstrap-timepicker input').datepicker({
  //changeMonth: true,
  //changeYear: true,
  format: "dd-mm-yyyy",
```

```
    weekStart: 1,
    startView: 0,
    todayBtn: "linked",
    language: "es",
    daysOfWeekHighlighted: "0,6",
    autoclose: true,
    todayHighlight: true,
    toggleActive: true
});
```

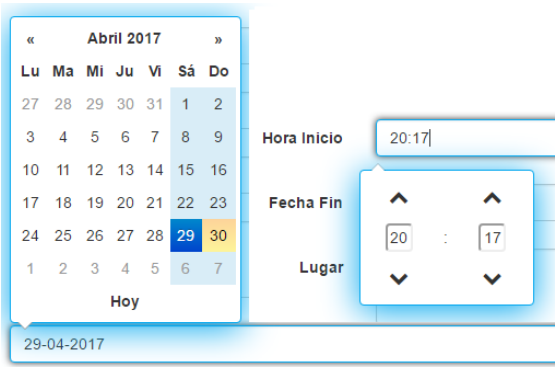

Figura 36: *Plugin* datepicker con calendario y hora

5.4.6. Duallistbox

Transforma los cuadros de selección múltiple ampliando su funcionalidad. Proporciona un buscador para una sencilla navegación entre los valores y separa los seleccionados de los que no lo están facilitando la introducción de datos.

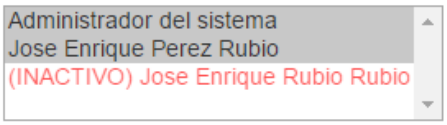

Figura 37: *Plugin* duallistbox: ejemplo desactivado

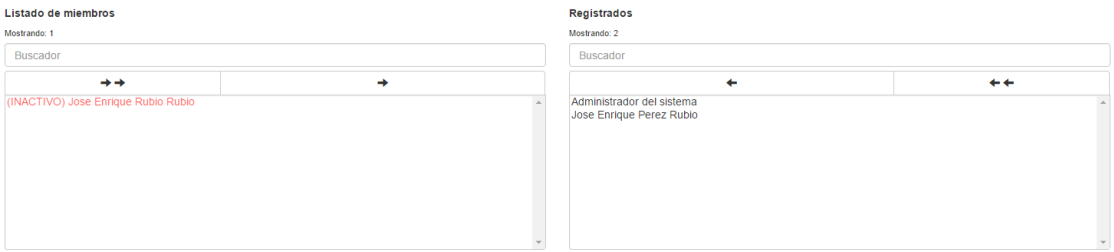

Figura 38: *Plugin* duallistbox: ejemplo activado

```
$('select[multiple="multiple"]').bootstrapDualListbox({
        nonSelectedListLabel     : '{{=T("List of members")}}',
        selectedListLabel        : '{{=T("Registered")}}',
        preserveSelectionOnMove  : '{{=T("moved")}}',
```

```
        infoTextEmpty           : '{{=T("Empty list")}}',
        infoText                : '{{=T("Showing all {0}")}}',
        infoTextFiltered        : '{{=T("{0} from {1}")}}',
        filterTextClear         : '{{=T("Show all")}}',
        removeSelectedLabel     : '{{=T("Remove selected")}}',
        removeAllLabel          : '{{=T("Remove all")}}',
        filterPlaceHolder       : '{{=T("Filter")}}',
        selectorMinimalHeight   : 200,
        moveOnSelect: false
});
```

Figura 39: *Plugin* duallistbox: configuración.

5.4.7. Select2

Este *plugin* se encarga de transformar los desplegables aportando un buscador en la parte superior.

```
$('select[class="generic-widget form-control row"]').select2({
                language: "es",
        placeholder: "Selecciona uno",
        noResults: "No se encuentran"
      });
```

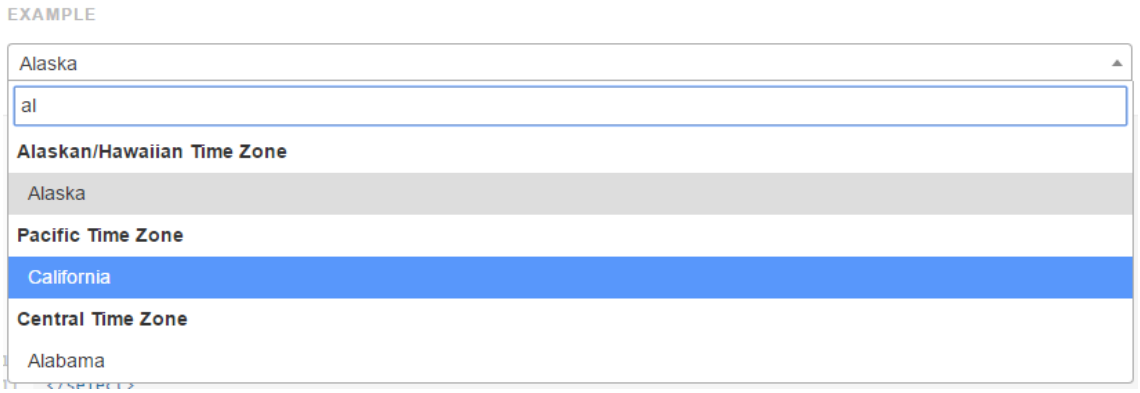

Figura 40: *Plugin* Select2: ejemplo de búsqueda.

6. Manual de usuario

6.1. Creación del proyecto

El trabajo consistía en la creación del proyecto por lo que se entregó fue un fichero .rar en el que se encuentra el proyecto comprimido.

Este debe ser descomprimido dentro de la carpeta “applications” la cual aloja los proyectos web.

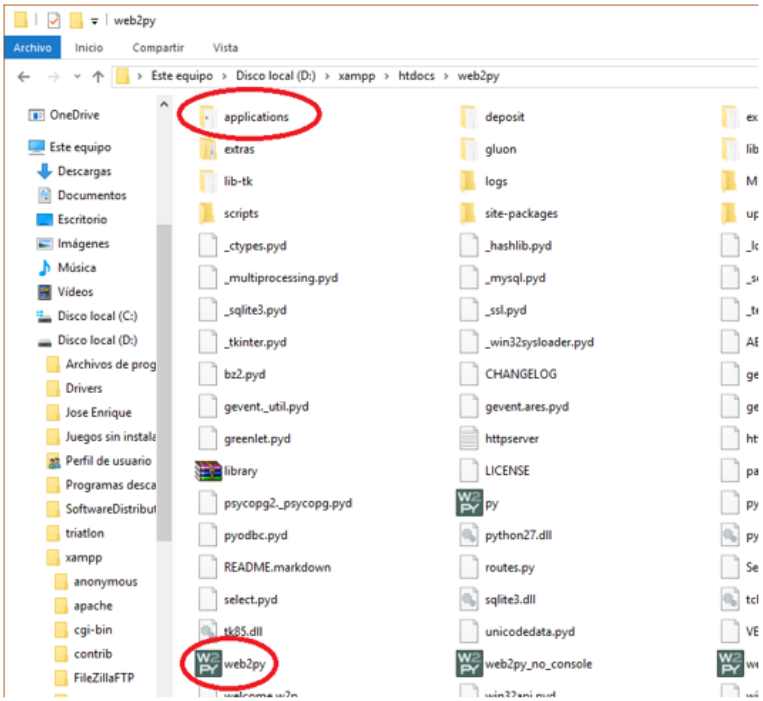

Figura 41: Directorios de web2py.

Una vez arrancado el servidor y la base de datos se puede acceder mediante un explorador de internet. En este caso se ha decidido acceder con Chrome ya que tiene un editor de código integrado el cual facilita la programación a la hora de consultar errores o la inserción de código de prueba.

Nada más intentar acceder a la web, esta genera de manera automática las tablas dentro de la base de datos ya que se encuentran definidas dentro de la aplicación. Se puede comprobar que ha creado las tablas y en las necesarias ha introducido valores por defecto.

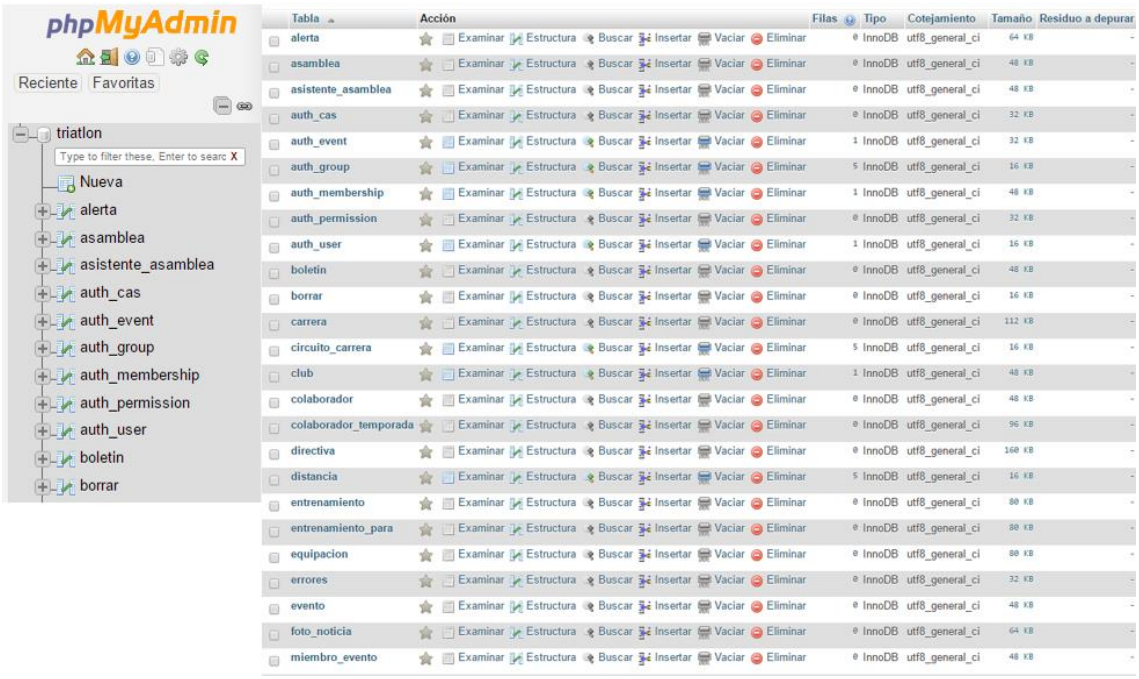

Figura 42: Tablas en la base de datos creadas de manera automática.

6.2. Crear una cuenta de usuario

Al acceder a la aplicación por primera vez, se acceder a la sección de autentificación y puesto que no se tiene una cuenta de usuario se necesitará crear una.

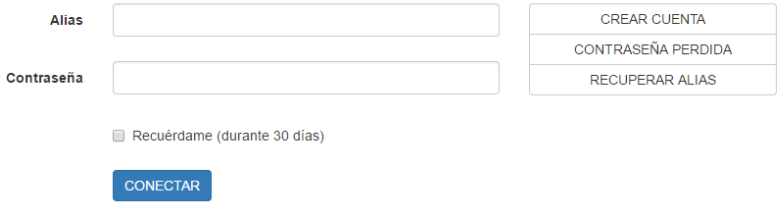

Figura 43: Login de la página.

Al crear una cuenta se debe de rellenar un formulario con una serie de campos que dan información sobre el usuario. Al terminar y enviar los datos, se recibirá un correo en la cuenta de correo introducida que sirve para verificar la dirección de correo electrónico.

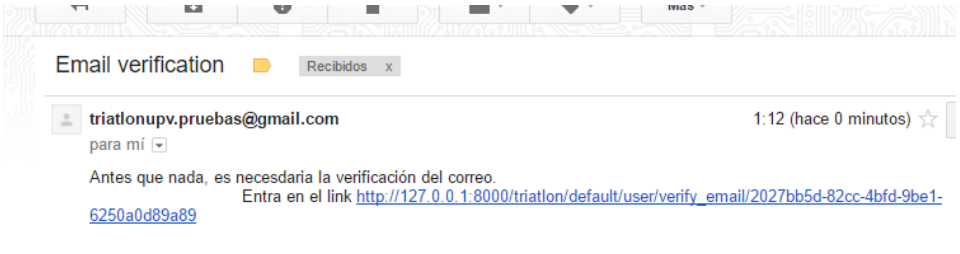

Figura 44: Email de verificación.

Al registrarse no se va a poder navegar por petición de requerimientos en el proyecto el cual se planteó que los usuarios solo pudiesen navegar por la aplicación con el previo consentimiento de un administrador por lo que la cuenta por defecto está bloqueada a la espera de ser revisada.

Figura 45: Cuenta bloqueada a la espera de ser revisada.

Un administrador puede ver todos los usuarios registrados en la sección de usuarios desde la cual puede ver los que se encuentran a la espera de ser desbloqueados.

Figura 46: Listado de los usuarios registrados.

Los usuarios que tienen la cuenta bloqueada aparecen semitransparentes y para facilitar su búsqueda, un atajo que tiene la aplicación, es escribir en el buscador la palabra “bloqueado” para que haga un filtrado de usuarios que se encuentran en ese estado.

En la modificación del usuario a la cual puede acceder un administrador aparece una sección en la cual puede aumentar o disminuir los permisos del usuario el cual se esté modificando, un botón para activar o desactivar la cuenta y un campo de texto donde añadir observaciones en el caso de que fuesen necesarias. Un ejemplo de observación podría ser que ha pagado la mitad de las tasas de un mes y este campo brinda una ayuda para recordar estas notas.

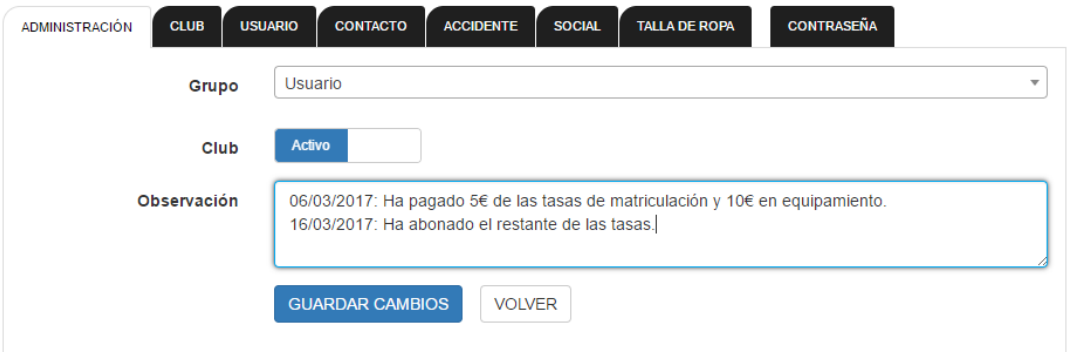

Figura 47: Sección de administración en el perfil de usuario.

6.3. Preparar una nueva temporada

El siguiente paso como administrador consiste en crear una temporada nueva. Se puede acceder al formulario de creación mediante el menú Temporadas en el lateral izquierdo.

Una temporada consta de un nombre que la identifica, una directiva, entrenadores, participantes inscritos y una equipación de temporada.

Según el grupo que tiene definido cada usuario se agrega por defecto como entrenador (el grupo se puede ver en la figura anterior). Los participantes se agregan por defecto excepto los entrenadores que actualmente no participan solo se dedican a entrenar pero existe la posibilidad que puedan llegar a participar en un futuro.

Si un usuario o entrenador no se encuentra registrado en una temporada no va a poder interactuar con esta.

6.4. Vista como usuario

Los usuarios tienen un menú diferente a los administradores y más simplificado el cual no tiene submenús.

- En la sección de inicio pueden ver las últimas noticias registradas y descargar los boletines de entrenamiento de una manera cómoda y fácil.
- En la sección de colaboradores se puede asignar un tipo de colaboración a los colaboradores registrados por un usuario con mayores privilegios de acceso. Un usuario no puede modificar el tipo de colaboración de otro usuario aumentando la seguridad ya que si un usuario se equivoca o lo quiere hacer de manera intencionada para perjudicar no va poder hacerlo.
- En la sección entrenamientos pueden ver los entrenamientos por temporadas para poder acceder a su descarga. A diferencia del listado que aparece en el menú principal, este ofrece un listado con todos los entrenamientos registrados y el otro solo muestra los últimos entrenamientos.
- En la sección noticias se pueden acceder a las noticias publicadas o crear nuevas. Esta tiene una foto principal y una fecha de caducidad. Al registrar la nueva

noticia aparece en un listado similar a todos los que se pueden encontrar en cada una de las secciones.

En la siguiente imagen se puede ver una nueva noticia creada. Esta noticia es marcada como nueva con un indicador azul en la parte izquierda y en la sección superior de alertas para el resto de usuarios. En la parte derecha de la imagen se encuentran tres botones de acceso rápido los cuales permiten ver la noticia, modificarla o borrarla.

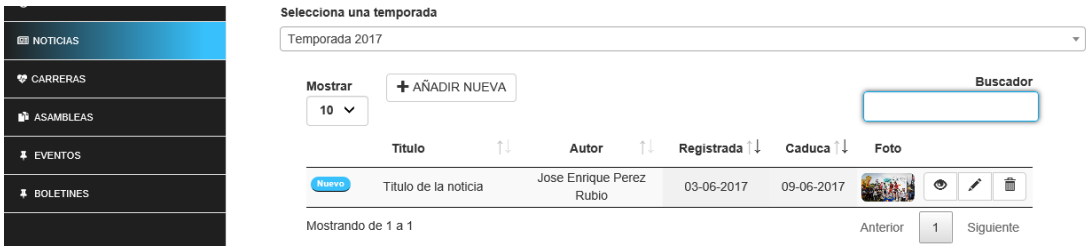

Figura 48: Listado de noticias.

El botón del lápiz sirve para modificar la noticia y en ella aparece una sección nueva para poder agregar nuevas imágenes además de la principal. Al agregar una imagen que no sea la deseada, se puede modificar o borrar, para ello se ha de posicionar el cursor sobre la imagen deseada y aparecerán dos botones nuevos semitransparentes.

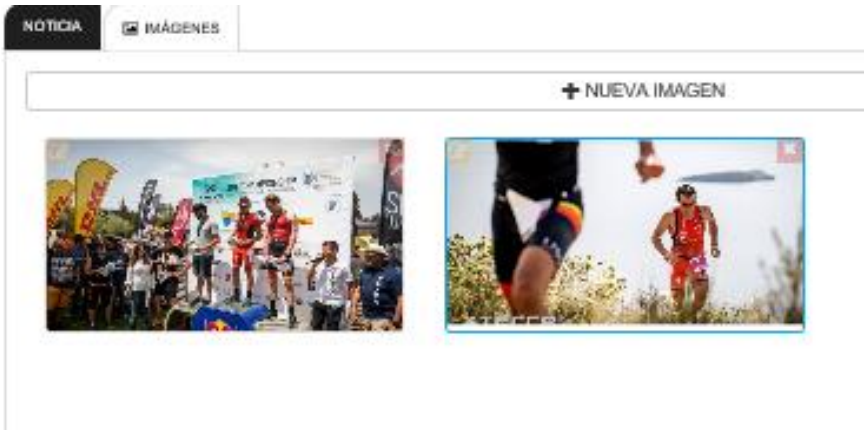

Figura 49: Imágenes secundarias de una noticia y como modificarlas o borrarlas

- En el menú se puede ver una sección de carreras en donde filtrando por temporada aparecen las carreras, en ella, los usuarios pueden agregar nuevas carreras y subir sus marcas en posición por categoría y posición absoluta tras competir. La categoría en la que participan se calcula de manera automática mediante la fecha de nacimiento guardada en el perfil.

- Las secciones de asambleas, eventos y boletines son similares en cuanto a funcionamiento y sirven para descargar o consultar información sobre estos. Los usuarios pueden crear nuevos eventos pero no asambleas o boletines.
- Para modificar los datos del perfil de usuario se debe acceder al botón situado en la parte superior derecha de la pantalla junto a las alertas de notificaciones. En este menú se puede acceder a los datos de usuario o cerrar la sesión.

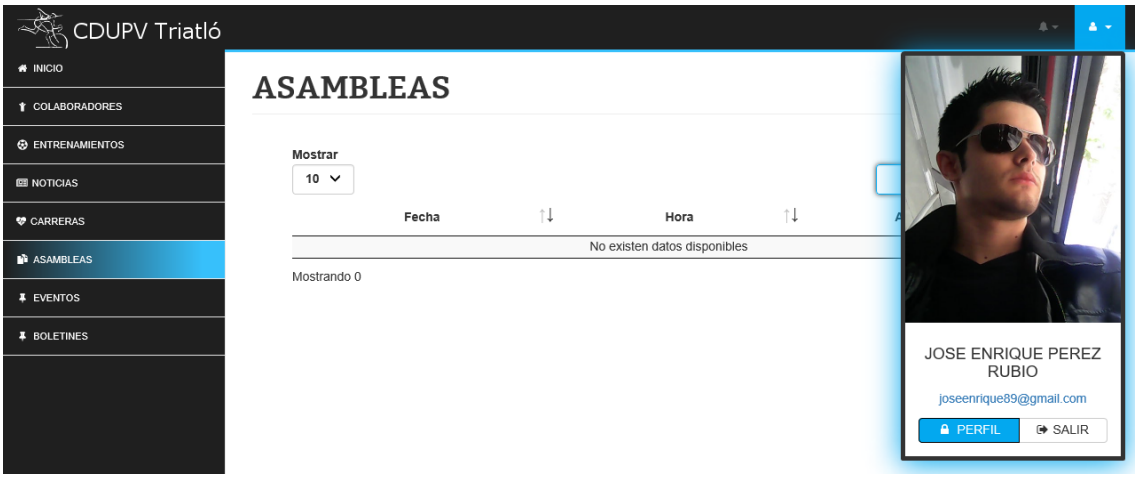

Figura 50: Como acceder al perfil de usuario o cerrar la sesión.

6.5. Vista como entrenador

- La diferencia entre el rol de entrenador y el rol de usuario es que estos pueden crear entrenamientos y asignarlos a usuarios.
  Al crear un nuevo entrenamiento este es compartido con todos los usuarios que se encuentren registrados pero se puede modificar esta lista en la sección de modificación del registro.

Figura 51: Asignar entrenamiento a usuarios

7. Conclusiones, problemas y contratiempos

En este último apartado de la memoria se incluye una sección en la que se describen dificultades y contratiempos surgidos durante el desarrollo del trabajo.

No todo va a ser perfecto siempre como es normal por lo cual han ido apareciendo ciertos problemas que han dificultado el desarrollo del trabajo retrasando en tiempos y haciendo más duro el proceso.

- **Falta de experiencia**: Esta es la primera vez que hago una página web y el hecho de tener que desarrollar una intranet con permisos hizo el trabajo más complicado ya que no es una ampliación sobre una web de la cual puedo tomar datos o con una base ya definida y un rumbo.
- **Desconocimiento del lenguaje**: Nunca antes había desarrollado en Python ni conocía Web2py. Antes de comenzar debí informarme de las reglas de programación de dicho lenguaje y del framework.
- **Problemas a la hora de consultar la API de Web2py**: Por si no fueran pocos los problemas, al comenzar con el trabajo, la página donde está muy bien explicada era inaccesible al lenguaje español e inglés por lo que tuve que recurrir a la página en otros idiomas e ir traduciéndola mediante traductores y diccionarios para entender su funcionamiento. Afortunadamente, un mes tras comenzar, arreglaron el problema que impedía ver la página en inglés y posteriormente en español.
- **Falta de información en internet:** Se presentaron casos concretos muy específicos los cuales no pude obtener ayuda por internet ni entre profesores de la universidad. En el caso del perfil de usuario mi idea fue el separar los campos en pestañas permitiendo una mejor organización y dando comodidad al usuario al no tener todos los campos seguidos creando una ventana más grande por la cual tenga que desplazarse que ofrezca una mala experiencia. Al consultar a un compañero de mi tutora, el cual ha trabajado con web2py, no pudimos resolver entre los 3 cuál era el problema por el que no se guardaban los datos ya que no viajaban hasta el controlador. Mi solución fue empezar de nuevo dicha sección planteándola de una manera diferente con dos pequeños ejemplos que vi en internet para juntarlos en una única solución.
- **Falta de tiempo en ocasiones:** La falta de tiempo siempre ha estado presente a la hora de dedicarle horas de trabajo ya que es difícil compaginar el trabajo, los estudios, academia y proyecto.

Pero pese a todo, he disfrutado de la experiencia y siempre que he tenido tiempo libre me ha gustado dedicarlo a intentar aprender cómo mejorar la aplicación. En esta aplicación aprendí acerca de nuevos lenguajes ya que había trabajado con PHP y como trabajar con plugins los cuales bien empleados hacen una página mucho más usable.

8. Bibliografía

[1] Wikipedia (Mar. de 2017), XAMPP. Disponible en: https://es.wikipedia.org/wiki/XAMPP

[2] Wikipedia (Feb. 2017), Apache. Disponible en: https://es.wikipedia.org/wiki/Servidor_HTTP_Apache

[3] Wikipedia (Mar. 2017), MySQL. Disponible en: https://es.wikipedia.org/wiki/MySQL

[4] Jose Mª Baquero (May. 2015), PHP. Disponible en: https://blog.arsys.es/comparamos-php-python-y-ruby-que-es-mejor

[5] Wikipedia (Mar. 2017), Python. Disponible en: https://es.wikipedia.org/wiki/Python

[6] Massimo Di Pierro y Alan Etkin (2017), Manual Web2py de referencia oficial. Disponible en: http://www.web2py.com/books/default/chapter/29/01/introduction

[7] Aptana Studio, Disponible en: https://es.wikipedia.org/wiki/Aptana_Studio

[8] Sublime Text, Entorno de programación, Disponible en: https://www.pabloyglesias.com/entornos-de-desarrollo-i-sublime-text-2/

[9] *Plugin* Autosize, Jack Moore Disponible en: http://www.jacklmoore.com/autosize/

[10] Sublime Text, Entorno de programación, Disponible en: https://www.pabloyglesias.com/entornos-de-desarrollo-i-sublime-text-2/

[11] Toogle switch jquery *plugin*, Disponible en: http://tinytools.codesells.com/ToggleSwitch

[12] Datepicker jquery *plugin*, Disponible en: https://bootstrap-datepicker.readthedocs.io/en/latest/markup.html