Trabajo Fin de Grado

Sistema de integración de eventos sobre tecnología en Zaragoza

Autor

Andrés Julián López

Director

Francisco Javier Zarazaga Soria

Escuela de Ingeniería y Arquitectura

2015

Sistema de integración de eventos sobre tecnología en Zaragoza

Resumen

El presente proyecto tiene como objetivo ofrecer un punto de encuentro en el que acceder a toda la información acerca de eventos sobre tecnología en la ciudad de Zaragoza. Para conseguir este objetivo, es necesario poder recopilar información de fuentes de datos externas. Estas fuentes de datos son las webs de aquellas entidades que organizan eventos tecnológicos en Zaragoza. Algunas entidades ofrecen muchas facilidades para que podamos obtener cierta información estructurada de su web como es el caso del Ayuntamiento de Zaragoza que ofrece una API para tal fin. Y otras no ofrecen tantas facilidades y es necesario recurrir a técnicas como el web-scraping desde una página web más o menos estructurada para extraer información de dichos eventos.

Otro aspecto a tener en cuenta es que resulta inviable tomar en consideración a todas las entidades que organizan eventos en Zaragoza, y es por ello, que se ofrecerá también una vía para permitir que terceros nos den a conocer sus eventos. Para ello, se ha especificado un sencillo protocolo de comunicación que utiliza la popular red de microblogging Twitter como canal. Mediante este protocolo, cualquier tercero, ya sea humano o máquina, será capaz de hacernos llegar eventos que de otra manera pasarían desapercibidos para nuestro sistema. Con esta aproximación, se consigue que esta plataforma no se encuentre cerrada a unos pocos organizadores, sino que cualquiera pueda publicar aquí sus eventos.

Finalmente, y no por ello menos importante, se ofrece al público una interfaz web en la que cualquiera pueda tener al alcance del ratón una visión rápida sobre los eventos que tienen lugar en Zaragoza en las próximas fechas. Esta interfaz esta accesible a través de la URL http://ztic.unizar.es

Índice

0. Resumen ejecutivo

Desde hace un par de años en la ciudad de Zaragoza se vienen organizando múltiples eventos relacionados con las Tecnologías de la Información y las Comunicaciones (Betabeers, etopia, Cachirulo Valley, Agile Aragón, ...). Cada uno de ellos cuenta con su propio canal de comunicación para informar a los interesados en participar (páginas web propias, cuentas de Twitter, etc). El objetivo de este Trabajo Final de Grado es desarrollar un punto de encuentro que posibilite un canal unificado de comunicación no invasivo sobre los sistemas que actualmente hay en marcha.

Para ello, es preciso recopilar información sobre dichos eventos tomándola directamente desde sus fuentes de origen o mediante intermediarios. Estos intermediarios son agentes ajenos a los organizadores del evento, pero que publican información sobre dicho evento.

Una vez recopiladas las fuentes de nuestros eventos, tenemos que adaptar y transformar esa información en bruto recibida. El objetivo es ofrecer un modelo de datos común a todos los eventos recopilados independientemente del origen que éstos tengan.

Finalmente, hay que ofrecer a los usuarios una interfaz sencilla de consulta en la que consultar los eventos próximos que sucedan en la ciudad de Zaragoza. Esta interfaz esta accesible a través de la URL http://ztic.unizar.es

1. Contexto del TFG

En la ciudad de Zaragoza, así como en otras ciudades, se ha detectado un interés creciente por las nuevas tecnologías. Este interés ha hecho que surjan multitud de eventos relacionados e integrados en el ámbito de las conocidas como TIC (Tecnologías de la Información y la Comunicación).

Estos eventos se suelen organizar desde entidades independientes tales como asociaciones culturales, empresas, comunidades de desarrolladores, entidades públicas (ayuntamiento, casas de juventud, …). Cada una de estas entidades tiene su propio medio de difusión de actividades y eventos. En el contexto actual, no existe una plataforma común en la que se agrupen estos eventos que comparten una temática más o menos relacionada.

No obstante si hay algo que tienen en común: el medio de difusión, aunque no el canal. Todos estos eventos se encuentran publicados en internet, “si no, no existen”. Cada uno de los organizadores de estos eventos, tiene su propia forma de publicar sus eventos. La forma de comunicación de eventos por cada organizador es completamente distinta. Desde la utilización de modelos de datos estructurados, proporcionando incluso una API, hasta la publicación de eventos a través de entradas nuevas en una web, o incluso, mediante tweets.

2. Motivación y objetivos

Este proyecto tiene como objetivo ofrecer un punto de acceso en la Web en la que poder conocer todos los eventos de tecnología que suceden en Zaragoza. Para ello, el sistema debe de ser capaz de recopilar información de diversas fuentes y unificarla de forma sencilla para el usuario.

Para ello hay que realizar un estudio previo sobre cuáles son los principales organizadores de eventos de Zaragoza. Posteriormente, se deben seleccionar aquellos que aporten más eventos y que a su vez sean muy relevantes. Una vez seleccionadas todas las fuentes, hay que hacer una segunda selección sobre aquellas fuentes que ofrezcan a través de internet y públicamente información periódica y fidedigna acerca de sus eventos (hay algunos casos en los que no se lleva a cabo el mantenimiento de esta información que generalmente se encuentra desactualizada).

Una vez seleccionadas las fuentes, el siguiente paso es analizar la forma en que está estructurada la información de cada una de ellas. Hay que estudiar qué datos ofrecen acerca de los eventos, cuáles de ellos nos van a resultar interesantes y cuáles no. De este modo, construimos nuestro modelo de datos para conseguir formar uno que aporte el nivel de información suficiente a los usuarios acerca de los eventos.

Una parte importante de este proyecto consiste en otorgar la mayor independencia posible a todos los componentes por los que estará formado este proyecto. Básicamente, consta de tres componentes principales:

- **Procesos ETL (Extract-Transform-Load):** Es el encargado de obtener la información, transformarla y adaptarla para posteriormente almacenarla.
- **Proceso storage:** Es el proceso de almacenar localmente la información para que esté siempre disponible, y para poder recuperar eventos pasados.
- **Proceso view:** Es el proceso encargado de mostrar a los usuarios los datos contenidos en el storage de forma que les resulten útiles.

La decisión de separar los tres modelos, tiene la finalidad de proporcionar una fácil escalabilidad del sistema en un futuro. De este modo será posible cambiar cualquiera de los tres elementos sin que los otros se vean afectados. Se consigue que así sea posible añadir nuevas fuentes de datos sin necesidad de correcciones ni adaptaciones en el resto de los componentes. También es posible de este modo cambiar la presentación al usuario o incluso, presentar la información de varias formas, como por ejemplo mediante web o mediante una API pública.

3. Trabajo desarrollado

3.1. Análisis de las fuentes de información.

La primera labor para realizar un trabajo de estas características, consiste en recopilar información acerca de publicadores de eventos sobre los que se puedan recolectar datos sobre eventos. De esta forma vamos analizando qué sitios en Zaragoza pueden resultarnos interesantes para recoger eventos.

Tras una serie de reuniones con profesores y alumnos del Grado en Ingeniería en Informática, y el acceso a algunas otras entidades a través del Director de este TFG, se establecieron las dos entidades que se consideraban representativas para formar parte de este proyecto:

3.1.1. Ayuntamiento de Zaragoza

El Ayuntamiento de Zaragoza ofrece en su web información sobre muchos de los eventos que se organizan en la ciudad. Además ofrece gratuitamente un API muy completa en la que publican información sobre la ciudad. Esta API forma parte del proyecto de Datos Abiertos del Ayuntamiento de Zaragoza y ofrece datos como la de todo tipo como: datos climáticos, sobre contaminación, taxis libres en la ciudad, etc. y entre ellos la agenda, que es el que nos interesa para este caso.

Además, se permitiría que los organizadores de eventos que no tuvieran página web publicada, pudieran tener algún método para hacernos saber sus eventos.

3.1.2. Betabeers

Como dice su propia web, "Betabeers es una comunidad de desarrolladores que organizan encuentros mensuales para compartir conocimientos sobre tecnología y Startups". En Zaragoza organizan eventos mensualmente. En su propia web ofrecen un listado de todos los eventos que organizan. Incluso los que se encuentran organizados por ciudades. Para cada evento nos muestra el título, la fecha y horarios en el que se organizará, la dirección en la que tendrá lugar y una descripción sobre la temática del mismo. Ofrecen un esquema de información fijo y repetitivo.

3.1.3. Twitter

Adicionalmente a estas dos entidades, se ha identificado que Twitter se usa habitualmente por muchas personas para comunicar información sobre actividades y eventos. La popular plataforma de microblogging ofrece muchas posibilidades, a la hora de intercambiar información de forma sencilla. La idea es aprovechar la potencia de esta plataforma para que los organizadores de eventos que quieran que un nuevo evento que organizan aparezca reflejado en esta plataforma puedan hacerlo de una forma sencilla. De este modo podemos conseguir que eventos que de otra forma no se podrían capturar, tengan una forma sencilla de informarnos sobre eventos y que se encuentren publicados aquí.

3.2. Diseño e implementación de la estructura de datos de eventos

La base de datos que recopila los eventos tiene una estructura muy básica. Consta de dos tablas llamadas evento y organizador donde se guarda la información sobre los eventos y los organizadores de eventos respectivamente. Todos los campos, a excepción de las claves primarias, son no obligatorios. Esto es así, porque al tratarse de información obtenida de fuentes distintas, es imposible exigir que todos los campos sean obligatorios para todos los eventos.

EL siguiente esquema muestra el diagrama Entidad-Relación de la Base de Datos.

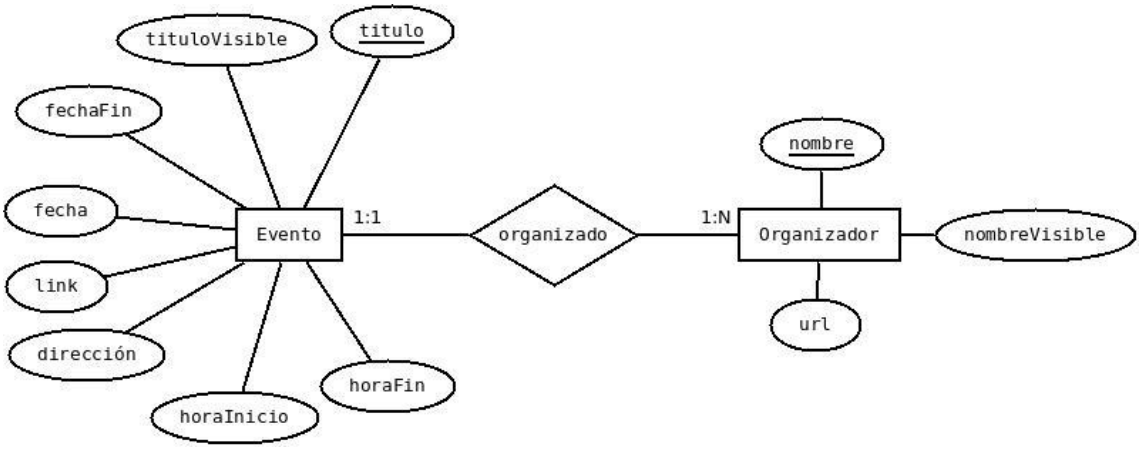


FIGURA 1: ESQUEMA ENTIDAD-RELACIÓN DE LA BASE DE DATOS

A continuación, se explican los campos de las dos tablas:

3.2.1. Tabla evento

- **titulo.** Es el título administrativo y clave primaria y guarda el título del evento recogido de la web. Aunque en otros contextos, no sería el campo más indicado para utilizar como clave, aquí es la mejor forma que tenemos de identificar unívocamente un evento y que además no se repita.
- **tituloVisible.** Es el título no administrativo, que aunque a priori es igual que el título, permite a los administradores de la web corregir o modificar el título del evento que será visible en la web, sin que esto afecte al comportamiento que tiene la clave primaria de título.
- **fecha.** Indica la fecha en la que tendrá lugar el evento, o en caso de ser un evento que dura varios días, la fecha de inicio.
- **fechaFin.** Es la fecha en la que finaliza el evento, en caso de eventos de varios días.
- **horaInicio.** La hora a la que dará comienzo el evento.
- **horaFin.** La hora en la que está prevista la finalización del evento.
- **link.** Enlace a alguna web que contenga información más detallada sobre el evento.
- **direccion.** La dirección en la que tendrá lugar el evento.

- **organizador.** Referencia a la otra tabla que contiene información sobre el organizador del evento.

3.2.2. Tabla organizador

- **nombre.** Nombre del organizador y clave primaria de la tabla.
- **url.** Contiene un link a la web de la entidad, organismo, asociación, … que organiza el evento.
- **nombreVisible.** Nombre a mostrar en la web del organizador del evento. Sigue los mismos principios que tituloVisible de la tabla evento, solo que aquí además, permite cambiar el nombre de un organizador permitiendo que los eventos pasados y futuros capturados mantengan la asociación con el mismo organizador.

3.3. Desarrollo del crawler de datos.

Para conseguir extraer información de sitios web necesitaremos utilizar lo que se conoce como crawler. Un crawler o araña web es un software encargado de recorrer de manera automática páginas web para obtener información o mantener una copia de los datos de la web[1]. En este caso tendremos que obtener información sobre eventos de tecnología de Zaragoza. Es la parte del sistema que se encarga de realizar el proceso ETL. A continuación, se detalla cada parte del proceso ETL.

3.3.1. Extraction

Es el proceso que consiste en obtener información desde Internet. Para ello hay que filtrar para reducir al máximo el número de peticiones.

Ayuntamiento de Zaragoza

El Ayuntamiento de Zaragoza ofrece información acerca de eventos que tienen lugar en Zaragoza. Estos eventos, se encuentran publicados a través de sus web, sobre la que podríamos utilizar la misma técnica que la empleada para Betabeers. Pero al Ayuntamiento de Zaragoza ofrece también una API pública en la que se pueden consultar, a parte de otros datos sobre la ciudad, la información sobre los eventos. Ofrecen los datos en distintos formatos, entre ellos JSON, que ha sido elegido para este proyecto por su sencillez y reducida verbosidad.

Realizando una consulta sobre la API nos devuelve todos los eventos de la ciudad de Zaragoza, ya sean futuros, actuales o pasados. Esta consulta, se puede parametrizar, se pueden realizar filtros, se puede especificar el formato de salida de los datos, y se puede limitar el número de eventos que se listan. Hay que parametrizar la consulta para conseguir que liste todos los eventos, porque sin parámetros solo lista 50 eventos.

[1] https://es.wikipedia.org/wiki/Ara%C3%B1a_web

Betabeers

Para obtener información de Betabeers se ha aprovechado su página oficial (http://betabeers.com). En su web tienen una sección donde se listan los eventos de Betabeers en cada ciudad. En el apartado de Zaragoza se listan todos los eventos que se celebran en esta ciudad. El listado de eventos se encuentra paginado, por lo que es necesario pasar páginas para poder listar todos los eventos.

Para poder listar todos los eventos es preciso recorrer todas las páginas del apartado eventos de Zaragoza. Para conseguirlo, se utiliza un mecanismo de screen scrapping[2] que permite recorrer todos los links de la web que hagan matching con una expresión regular. Además tiene en cuenta las páginas ya visitadas para no volver a pasar por una página ya analizada. De esta forma, conseguimos que, al pasar a la segunda página del listado, no siga el link que vuelve a la primera página.

Twitter

Los dos rastreadores anteriores sirven como punto de partida para permitir que el sistema continúe creciendo. Pero por muchas webs que podamos rastrear, nuestro sistema siempre estará incompleto. Siempre existirá una web que no hayamos tenido en cuenta, o un eorganizador que no tenga un sitio web. Para ello, se ha creído conveniente desarrollar un mecanismo para permitir que terceros (humanos o máquinas) sean capaces de hacernos saber sus eventos.

Para ello se ha especificado un sencillo protocolo de comunicación unidireccional que permita la publicación de eventos por parte de terceros. Se trata de un protocolo unidireccional ya que sólo permite a terceros enviar eventos sin necesidad de interacción por nuestra parte. De hecho ni siquiera recibirán ningún tipo de *acknowledgement* (ACK) para confirmarles que hemos parseado su Tweet. La codificación del protocolo es pública a través de la web y permite que en un solo Tweet se codifiquen los campos mínimos necesarios para crear un evento. Estos campos son: título, fecha, hora, dirección y link a alguna URL que amplíe información sobre el evento. Los campos se encuentran separados por el carácter | (pipe) y el Tweet debe contener el Hastag #ZTic para que sea capturado por nuestro sistema.

Los datos de Twitter se extraen mediante una librería de Python que nos permite realizar consultas y manejar nuestra cuenta de Twitter a través de una API. Para poder utilizar esta librería es necesario tener una cuenta de Twitter y registrarse en su página de Developers[3] como desarrollador. Así conseguimos obtener los tokens y las keys para utilizar la API. Para listar los eventos se utiliza el método GetSearch que nos permite realizar una búsqueda en texto sobre Twitter. De este modo, buscamos los eventos que contengan el Hastag #ZTic para buscar los eventos que sigan el patrón que

[2] https://es.wikipedia.org/wiki/Screen_scraping
[3] https://dev.twitter.com/

nos permite añadir eventos sugeridos por los usuarios de Twitter, como se explica en el Anexo I.

Cuando realizamos una búsqueda, la consulta nos devolverá todos los eventos de Twitter que contengan dicho Hastag (limitado a 100 tweets). Hay que tener en cuenta que nos devolvería tanto tweets nuevos, como tweets que ya hayamos procesado previamente. Los tweets, tienen un identificador único llamado "id". Gracias a este id, podemos filtrar la búsqueda para que nos muestre sólo tweets con id mayor que un id dado. Para poder utilizar esta funcionalidad, necesitamos de un mecanismo que nos permita recordar cuál fue el último elemento procesado. La forma elegida para preservar este id es guardar en un fichero de texto el id del último tweet procesado por nuestra aplicación. De este modo, en la siguiente consulta se mostrarán sólo los tweets posteriores al último procesado por la aplicación.

3.3.2. Transformation

Es el proceso mediante el cual la información recopilada es adaptada para que encaje con nuestro modelo de datos. Este proceso conlleva aplicar técnicas de:

- Filtrado de datos, para limpiar resultados que no se han podido filtrar en el proceso de extracción
- Conversión de datos, transformando formatos o tipos de datos
- Unión y división de campos.

Ayuntamiento de Zaragoza

En el ayuntamiento de Zaragoza, a través de su API, conseguimos un documento JSON[4]. En este documento, se encuentran mezclados eventos de tecnología con eventos de otra índole. El principal problema de esta API es que los datos no guardan una estructura definida. Es decir, no todos los eventos tienen los mismos campos, ni siquiera existe una guía donde se explique el significado de cada campo de un evento.

Aunque los nombres de los campos son más o menos intuitivos, en algunos casos hay que analizar el contenido del campo para conseguir saber realmente a qué hacen referencia. Además tenemos que tratar de rellenar todos los campos que tenemos asociados a nuestro evento en nuestra base de datos, para que la información que ofrezcamos sea lo más completa posible.

Como los eventos se encuentran mezclados, a priori es muy difícil separar los eventos tecnológicos de los eventos no relacionados. Existe un campo multivaluado de evento llamado "temas" en el que se incluyen todos los temas a los que va asociado un evento. El problema es que no existe documentación acerca de los diversos temas que pueden caracterizar al evento, a pesar de que cada tema, lleva asociado un título y un ID. Para ello, el primer paso fue listar todos los temas de todos los eventos evitando

[4] https://es.wikipedia.org/wiki/JSON

repeticiones. Los eventos que tiene actualmente publicados el Ayuntamiento de Zaragoza superan los 35.000 y algunos de ellos datan de hace más de cinco años. Una vez obtenido el listado, y analizando los temas se concluyó que el único tema al que podían ir asociados estos eventos es a uno llamado "Tecnología y Ciudadanía". Este fue un proceso inicial de configuración del sistema. En el caso de acceder a canales de eventos de otras entidades sería necesario seguramente hacer un procedimiento análogo.

Betabeers

En Betabeers todo lo que hemos extraído en el punto anterior es una página en HTML donde se encuentran todos los eventos de una de las páginas. En cada página aparece un listado de 20 o menos eventos.

Procesar el documento en HTML como texto plano resulta bastante complejo, porque se trata de un código muy verboso. Y, además, tiene unas reglas gramaticales que no todas las webs cumplen haciendo aún más dificultoso el proceso. Por suerte, Scrapy (entorno de desarrollo utilizado y que se explica más adelante) posee una funcionalidad que permite extraer partes de un documento HTML realizando una búsqueda por XPath[2].

Gracias a la búsqueda por XPath, podemos encontrar los bloques <li> que contienen los eventos individualmente, e incluso, nos permite iterar sobre todos ellos. De esta forma, entramos en un bucle en el que tenemos una variable que contiene el código HTML de cada evento. A partir de ahí, vamos extrayendo los datos del evento (título, fecha, dirección, link,...) a excepción de la hora de finalización del evento y el año del evento.

La hora de finalización y el año del evento se encuentran dentro de la página del propio evento. Para poder capturar estos datos, es necesario seguir en enlace capturado en el párrafo anterior. De este modo, conseguimos un nuevo documento HTML en el que se encuentra toda la información sobre el evento antes capturada, y además, la hora de finalización del evento, la fecha ampliada con el año y una descripción sobre los temas a tratar durante el evento. El procedimiento a seguir para capturar la hora de finalización y año es similar al del resto de datos, solo que en este nuevo documento HTML no es necesario iterar. Por lo que basta con encontrar el XPath adecuado y ya tenemos la hora de finalización.

La mayoría de los campos obtenidos, no se encuentran listos para ser consumidos. Hay que realizar algunos ajustes para que sea posible estandarizarlos al formato de nuestro modelo de datos:

- El título es un campo de texto, que se queda como está.
- Fecha y horas de inicio y finalización. Se encuentran todos en una misma cadena de texto. Hay que separar las horas de inicio y fin y ponerlas en formato

adecuado para la consulta SQL. En la fecha es necesario convertir el mes, que se encuentra en formato texto en castellano a su número correspondiente.

- La dirección y el link no es preciso modificarlos.
- El organizador del evento, en este caso se introduce estáticamente ya que es siempre el mismo: Betabeers.

Twitter

En la parte de extracción de Twitter habíamos conseguido que los tweets se filtren por el hastag #ZTic y que se muestren sólo los no procesados previamente. A pesar de ello, tenemos que conseguir que la aplicación capture sólo los eventos que cumplen con el formato de mensajes especificado en el Anexo I.

En este caso se trata de un formato de eventos muy simplificado que tan sólo contiene título, fecha, hora, dirección y link. No obstante, es necesario verificar que los contenidos proporcionados son consistentes. Por ejemplo, que la fecha del evento se encuentre en un rango adecuado (de hoy a dentro de un año), que la URL sea coherente, etc.

3.3.3. Load

La forma de almacenar los eventos es igual para todos los orígenes de datos. Los datos se almacenan en una base de datos MySQL formando una estructura sencilla en la que se relacionan eventos con organizadores como se explica en el Apartado 3.2.

3.4. Desarrollo de la interfaz de visualización de eventos.

3.4.1. Estructura de la web

Hasta ahora simplemente habíamos recopilado los datos en una base datos MySQL. La base de datos, no es claramente una forma atractiva de mostrar los eventos recopilados al usuario. Es por ello, que hay que crear un frontend que permita a los usuarios acceder al catálogo de eventos de una forma sencilla. La forma elegida para esta visualización es mediante la web. Por tanto, se pretende crear una interfaz sencilla que permita a los usuarios visualizar todos los eventos que haya recopilado nuestra aplicación.

Un esquema del aspecto de la web:

FIGURA 2: ESQUEMA BÁSICO DE LA WEB

La estructura de la web se plantea con una interfaz y navegación muy sencillas. Las páginas principales contendrán listados con eventos. Los componentes que contiene el sitio web son los siguientes:

Página de inicio

La página de inicio será la página principal donde se mostrarán por orden cronológico todos los eventos futuros y los eventos que hayan tenido lugar en los 15 días a la fecha de la visita de la página.

Página de eventos pasados

Se trata de una página de apariencia similar a la página de inicio que permita consultar todos los eventos pasados a modo de histórico.

Página de organizadores

Una página donde aparecen todos los organizadores de eventos que existen registrados en el sistema. Ofrece a los usuarios un link a la página del organizador (si existe) y haciendo click en el título del organizador nos lleva a la página de eventos de organizador. Cuando nos encontramos en esta página, en el menú principal junto al enlace a Organizadores aparece un desplegable que nos lista todos los organizadores y, si hacemos click sobre uno de ellos nos lleva a la página de eventos de dicho organizador.

Página de eventos de organizador

Esta página nos muestra todos los eventos que han sido publicados por un mismo organizador. En el menú principal aparece un nuevo elemento cuando nos encontramos en esta página con el nombre del organizador y al hacer click nos lleva a su sitio web.

Menú principal

Una barra superior nos muestra los diferentes apartados de la página web. Los enlaces disponibles son: Inicio, Organizadores y Anteriores. Se trata de un menú dinámico y cambia según la página de la web en la que nos situemos. Cuando nos encontramos en una página de las que figuran en el menú, dicho elemento aparece resaltado para facilitar la navegación.

Evento

Es uno de los componentes principales. Este elemento aparece en todas las páginas que muestran eventos. Se trata de un cuadro que contiene los elementos principales de un evento. El título aparece destacado, y al hacer click sobre éste nos lleva a la página web externa del evento. Aparecen también los campos dirección, que nos lleva a la ubicación del evento en *Google Maps*; y los campos fecha y horario, que al hacer click nos generan una plantilla para añadir a nuestro calendario de *Google Calendar* el evento.

Pie de página

El pié de página se ha utilizado para acceder a otras secciones de la web que no interesan tanto al público en general. En concreto se ha añadido una sección de “Acerca de ZTic” donde se explica brevemente el proyecto y una sección titulada “Publicar eventos desde Twitter” que explica el protocolo a seguir para publicar un evento en la web desde Twitter.

3.4.2. Componentes de Symfony

Para el desarrollo de la web se ha empleado Symfony ya que permite una gran versatilidad. Gracias a Symfony, podemos crear plantillas sobre componentes de la web que podremos reutilizar en todas las páginas que compondrán la web. De este modo, podemos utilizar la misma cabecera en todas las páginas a modo de menú.

El modelo web de Symfony consta de los siguientes componentes principales:

Router

Es el encargado de mapear las peticiones GET de una ruta con el controlador que debe ejecutar la acción. Se encarga de saber qué controlador nos va a renderizar la página que hemos solicitado o devolver error en caso de que no exista. Por ejemplo si pedimos GET /organizadores llamará al controlador que se encarga del renderizado de la página que nos lista los organizadores.

Doctrine

Es un mapeador objeto relacional de PHP que nos permite consultar los eventos de la base de datos. Aporta gran versatilidad al código ya que, una vez configurado correctamente el mapeo, permite tratar las consultas sobre la base de datos como objetos de PHP. Así podemos fácilmente seguir la relación entre evento y organizador de una forma sencilla.

EventsController

Es el controlador y se encarga de preparar la respuesta a una petición web. Se encarga de llamar al render de una página estática, como puede ser la página de “Qué es ZTic” o bien de realizar consultas sobre la base de datos para que pasando el resultado de la consulta al render junto con la plantilla, genere el código HTML de la web que hemos solicitado.

Plantillas twig

Son la forma de dar forma al documento HTML generado. Permite gran modularidad de dos formas distintas:

- Podemos reutilizar partes del documento que vayan a ser comunes a varias secciones de la web, como es el caso del header y el footer. En este caso, el header y el footer se encuentran en todas las secciones de la web. De modo que se incorporan a la plantilla base para que todas las demás hereden esta base.
- Podemos reutilizar una misma plantilla para dibujar varias secciones de la web incluyendo elementos variables en la plantilla. En este caso, se puede observar que el formato de la sección de eventos pasados, es igual que la sección de eventos futuros, y que solo cambia el contenido y el título. De este modo, el controlador se encargará de pasar a la plantilla los eventos correspondientes y

el título y tendrán el mismo aspecto el listado de eventos pasados que de eventos futuros.

3.5. Despliegue de la aplicación

El despliegue de la aplicación también forma parte de este proyecto. Esta aplicación requiere estar ejecutándose periódicamente en una máquina real. Además requiere de una serie de componentes software para hacer funcionar aplicación.

3.5.1. Hardware y Sistema Operativo

Para ello, el Departamento de Sistemas de la Universidad de Zaragoza nos aprovisionó una máquina. Se trata en concreto de una máquina virtual de 2 cores, 2GB de RAM y 25 GB de disco duro. Como sistema operativo se ha seleccionado Debian por ser un SO Linux, muy robusto y estable.

3.5.2. Servidor Web

Como servidor web, se ha elegido Apache2. Se trata de un servidor HTML de código abierto muy extendido potente y flexible. Se configura de forma muy sencilla, para sites básicos, aunque permite configuraciones más complejas. En este caso, ha sido necesario configurar el ServerName que sirve para indicarle a Apache qué nombre DNS debe de escuchar para este sitio. Este parámetro se utiliza especialmente en configuraciones multi-site, es decir, alojar varias webs en un mismo servidor. Además ha habido que activar el módulo *mod_rewrite* que permite la reescritura de URLs. Gracias a esta reescritura de URLs podemos ocultar a los usuarios detalles acerca de la estructura de ficheros de la web, permitiendo que el router de Symfony se encargue de hacer de forma transparente las llamadas a los ficheros que componen las páginas.

3.5.3. Ejecución automática

Para permitir la ejecución automática del crawler, se ha configurado el cron[1] del sistema para que lance los procesos rastreadores de la siguiente manera:

Un proceso es llamado todas las noches y se encarga de buscar actualizaciones en las webs. Se utiliza la franja nocturna previendo que habitualmente sea el momento que menos carga soportan las webs externas sobre las que recopilamos información. Además, las webs normalmente hacen públicos sus eventos con suficiente antelación para que las personas interesadas en ellos, puedan enterarse, organizarse y asistir. Por este motivo no es necesaria una mayor periodicidad a la hora de rastrear eventos.

Otro proceso se encarga de rastrear Twitter cada hora en busca de nuevos eventos. Los eventos publicados en Twitter se rastreen cada hora porque hay que tener en cuenta que la naturaleza de éstos eventos. Cuando alguien publique en Twitter un evento, éste puede tener lugar en pocas horas, o incluso estar teniendo lugar en el

momento. Por esta razón, los eventos publicados en Twitter tienen una periodicidad mayor que los publicados en la web.

Además, la API de Twitter en su versión gratuita pone un límite a la cantidad de operaciones que se pueden realizar con ella. Entre muchas otras cosas, tiene un límite de búsquedas que se pueden realizar por hora y otro de resultados de búsqueda que lista por hora. De este modo, realizando una búsqueda por hora, también nos garantizamos el exprimir al máximo la API de Twitter y evitando que algún Tweet se nos pueda quedar oculto por el camino debido a que hemos excedido las limitaciones de uso de la API de Twitter.

3.5.4. Debugging y logs

Para permitir detectar posibles fallos de la aplicación, ya sean internos (p.ej.: errores de programación) o externos (p.ej.: cambios en el diseño de las webs externas) es necesario guardar un registro de las salidas de los rastreadores. Estos rastreadores se ejecutan en *background* por lo que la única manera de analizar su comportamiento es mediante el uso de logs. Los procesos rastreadores, tienen redirigida su salida estándar y de errores a ficheros en directorio de logs del sistema (/var/log/ztic).

Esta solución, aunque muy útil, genera unos ficheros que pueden llegar a ocupar mucho espacio si no se tratan adecuadamente. Para solucionar este problema, se ha utilizado la herramienta de UNIX *logrotate* que simplifica la llamada “rotación de logs”. Esta rotación de logs consiste en limpiar ficheros de log antiguos para evitar que crezcan indefinidamente. En este caso, se ha optado por almacenar los logs durante 12 semanas (casi 4 meses). Además, se genera un nuevo fichero de log por cada semana y los ficheros anteriores se comprimen para reducir su tamaño. De este modo, tendremos un fichero de log en texto plano correspondiente a la semana actual y once ficheros de log correspondientes a las 11 semanas anteriores.

4. Tecnologías utilizadas

4.1. Apache2

Apache[3] es un servidor HTTP multiplataforma y de código abierto. Tiene su propia licencia de software llamada Licencia Apache 2.0 que es muy similar a una licencia de software libre, sólo que ésta no exige compartir el código en versiones modificadas. Apache es usado por más de la mitad de sitios web activos actualmente[4].

Apache2 consta de un core con las funcionalidades básicas de servicio HTTP, pero además tiene diversos módulos que le aportan nuevas funcionalidades. Uno de los módulos más utilizados, incluido en este proyecto, es el módulo llamado mod_rewite que permite la reescritura de direcciones, permitiendo ocultar al cliente detalles acerca de la implementación de la página web y utilizar URLs más limpias.

4.2. Symfony2

Symfony[5][6] es un framework para construir aplicaciones web desarrollado en PHP. Está pensado para desarrollar aplicaciones con MVC (Modelo - Vista - Controlador). Utiliza un diseño modular basado en herencia que aporta gran flexibilidad para el desarrollo de aplicaciones complejas.

4.3. Scrapy

Scrapy[7] es un framework escrito en Python para hacer crawlers de la web, aunque también permiten información de APIs. Es de código abierto bajo licencia BSD. La ventaja de scrapy es que utiliza python que ha demostrado ser un lenguaje muy potente y ligero. Y además, cuenta con una herramienta que permite utilizar reglas (rules) para seguir enlaces automáticamente y, muy importante, sin repetirlos. Esto quiere decir que en podemos recorrer un sitio web completo, siguiendo todos los enlaces que encuentre que apunten dentro de la misma web, sin pasar dos veces por una misma página.

4.4. MySQL

MySQL[8] es un gestor de bases de datos multiusuario y multi-hilo. Se licencia bajo GPL o con uso privativo. Se trata del segundo gestor de bases de datos más utilizado, sólo por detrás de Oracle. Se trata de un gestor de bases de datos muy rápido en lecturas. No resulta muy adecuado para sitios con gran concurrencia de escrituras.

4.5. Twitter

Twitter[9] es una plataforma de microblogging. Permite que los usuarios publiquen mensajes de una longitud inferior a 140 caracteres. Los mensajes son públicos y pueden ser vistos tanto por usuarios registrados como por usuarios no registrados. Aunque también permite enviar mensajes privados entre usuarios. Los mensajes se pueden etiquetar a un tema mediante los conocidos como “Hastag” que consisten en una almohadilla (#) seguida de una palabra.

5.- Lecciones aprendidas y conclusiones

Como resultado de este proyecto, se ha conseguido una web pública en la que de manera automática se recogen y publican eventos TIC en la ciudad de Zaragoza. Durante todo el proceso de desarrollo de este sitio web, se ha adquirido una visión sobre el mundo de internet en general y sobre los crawler de la web en particular.

Como resultado de la realización de este proyecto, he sido capaz de comprender cómo funcionan a un nivel muy básico muchos de los sitios de Internet. Cómo es capaz que un buscador sea capaz de servirnos en unas milésimas de segundo resultados de búsqueda sobre una cantidad inmensa de sitios web. La respuesta está en los crawler. Este proyecto, lejos de compararse en dimensiones con un buscador de Internet, cumple con algunos de los requisitos que éstos tienen. Es decir, posee mecanismo para recopilar información de manera automática de la web (sin intervención del ser humano), esto es lo que viene siendo el crawler básico. Y además, posee un mecanismo para "comunicar" al sistema manualmente un evento, a través de twitter.

Durante el proceso de desarrollo, se han debido de tomar diferentes decisiones que han resultado en el diseño final de la infraestructura empleada para este proyecto. De entre ellas, cabe destacar la tecnología empleada para crear la página web que muestra los eventos. A priori, se pensó en utilizar Drupal como CMS para realizar una web de una forma aparentemente más sencilla. Pero el problema que tiene Drupal es que está muy cerrado a su propia estructura interna, y no resultaba nada sencillo adaptar una fuente externa de información. Esta fuente externa de información es la base de datos que almacena los eventos. Además, Drupal resulta demasiado complejo para un proyecto como éste en el que premia la sencillez en el apartado web.

Otro aspecto que cabe destacar de este proyecto es la parte de recogida de información de sitios web preexistentes. Los sitios web, habitualmente no proporcionan ninguna información acerca de su estructura interna. Esto implica una gran carga de análisis y de ingeniería inversa para conseguir averiguar la estructura interna que sigue la web. Y también, en otros casos, hay que adivinar la forma de conseguir lo que quieres. Es el caso, por ejemplo, de la web del Ayuntamiento que, a pesar de que proporciona una API bastante versátil, no ofrece a priori, al menos según la documentación, una forma de listar todos los eventos con una sola petición. La petición estándar devuelve un máximo de 50 eventos por petición. Y, posee un parámetro para poder solicitar un número determinado de eventos. De modo que puedes pedir 70 o 127 eventos, pero no tiene un método publicado para obtenerlos todos. La única manera de obtener todos los eventos, fue buscando un fallo de este parámetro. El fallo consiste en solicitar -1 eventos, y así se consiguen listar todos los eventos.

6.- Estructura de los anexos

Se van a enumerar los anexos existentes

- Anexo I : Publicación de eventos en Twitter
- Anexo II : Tutorial para la creación de un nuevo crawler de datos
- Anexo III : Diagrama de componentes
- Anexo IV : Ejemplo de formato de la web
- Anexo V : Manual de uso de la web

7.- Bibliografía

[1] Cron (Unix). (n.d.). Retrieved November 17, 2015, from https://es.wikipedia.org/wiki/Cron_(Unix)

[2] XML Path Language (XPath)Version 1.0. (n.d.). Retrieved November 18, 2015, from http://www.w3.org/TR/xpath/

[3] Apache HTTP Server. (2015, November 9). Retrieved November 17, 2015, from https://en.wikipedia.org/wiki/Apache_HTTP_Server

[4] March 2015 Web Server Survey. (2015, March 1). Retrieved November 18, 2015, from http://news.netcraft.com/archives/2015/03/19/march-2015-web-server-survey.html

[5] The Symfony Cookbook. (2015). Symfony. http://symfony.com/pdf/Symfony_cookbook_2.7.pdf?v=4

[6] Symfony. (2014, September 14). Retrieved November 18, 2015, from https://es.wikipedia.org/wiki/Symfony

[7] Scrapy. (2015, July 29). Retrieved November 18, 2015, from https://en.wikipedia.org/wiki/Scrapy

[8] MySQL. (2015, November 9). Retrieved November 18, 2015, from https://en.wikipedia.org/wiki/MySQL

[9] Twitter. (2015, November 12). Retrieved November 18, 2015, from https://es.wikipedia.org/wiki/Twitter

Anexo I. Publicación de eventos en Twitter

Si conoces algún evento que haya sido incluido en nuestra web y quieres hacerlo público, puedes utilizar Twitter para darnos a conocer dicho evento. El tweet debe de cumplir con unos requisitos para pueda ser capturado por nuestra aplicación. A continuación te mostramos qué debes de hacer para que tu evento sea recogido automáticamente por nuestra aplicación.

Datos necesarios

Para publicar un evento en Twitter y que sea capturado por nuestra aplicación el tweet debe cumplir los siguientes requisitos:

- Debe contener el Hastag **#ZTic**.
- Se deben conocer del evento los siguientes datos:
  - Título del evento
  - Fecha en la que tendrá lugar el evento
  - Hora del evento
  - La dirección en la que tendrá lugar
  - Un link a alguna URL que amplíe información sobre el evento

Si no conoces alguno de estos datos no podrás publicar el evento.

Formato del tweet

El *tweet* debe de seguir un formato en el que se deben de incluir todos los campos mencionados en el apartado anterior. Los campos deben de seguir siempre el mismo orden, pero el *hastag* puede aparecer en cualquier lugar. Como elemento separador entre campos se debe utilizar símbolo | (pipe). El orden de los campos es el siguiente:

> Título del evento | Fecha del evento (DD-MM-AAA) | Hora del evento (HH:MM) | Dirección del evento | Link al evento

Es importante respetar los formatos de Fecha y Hora para que sean capturados correctamente por la aplicación

Ejemplo de *tweet*

A continuación tienes un ejemplo de un *tweet* que pueda ser capturado por nuestra aplicación:

> Presentación ZTic|20/12/2015|12:00|María de Luna 3|http://www.unizar.es/ **#ZTic**

Otro ejemplo:

> **#ZTic** Google en Zaragoza | 19/10/2015 | 18:00 | Echegaray y Caballero 10 | http://www.google.es/

Anexo II. Tutorial para la creación de un nuevo crawler de datos

A modo de ejemplo para la explicación, se muestra el código completo del crawler de Betabeers y en la página siguiente se detallarán los elementos que son necesarios modificar para crear un nuevo crawler.

```
from scrapy.http import Request
from scrapy.contrib.spiders import Rule, CrawlSpider
from scrapy.contrib.linkextractors.lxmlhtml import LxmlLinkExtractor
from ztic.items import ZticItem

class Betabeers(CrawlSpider):
    name = 'betabeers'
    allowed_domains = ['betabeers.com']
    start_urls = ['https://betabeers.com/community/events/?id=27&p=1']


    rules = [Rule(LxmlLinkExtractor(allow=(
                    r'(.*)betabeers.com/community/events/\?id=27&p=(.*)',
                )), callback='parse_events', follow=True),
             ]

    def parse_events(self, response):
        eventos = response.xpath('//div[@id="tab_box_events"]/ul').xpath('li')
        for sel in eventos:
            item = ZticItem()
            item['organizador'] = 'Betabeers';
            item['fecha'] =
sel.xpath('div[@style="float:left;width:80px"]/text()').extract()[1].strip()
            item['horaInicio'] =
sel.xpath('div[@style="float:left;width:80px"]/text()').extract()[2].strip()
            item['link'] = sel.xpath('div/a/@href').extract()[0].strip()
            item['titulo'] = sel.xpath('div/a/text()').extract()[0].strip()
            item['direccion'] =
sel.xpath('div[@style="float:left;width:75%"]/text()').extract()[1].strip()
            request = Request(item['link'], callback=self.parse_event)
            request.meta['item'] = item
            yield request

    def parse_event(self, response):
        item = response.meta['item']
        fecha = response.xpath('//div[@id="tab_box_info2"]/p/text()')[0].extract()
        item['horaInicio'] = fecha.split(',')[1].split('-')[0].strip()
        item['horaFin'] = fecha.split(',')[1].split('-')[1].strip()
        item['fecha'] = self.convierte_fecha(fecha.split(',')[0])
        yield item

    def convierte_fecha(self, cadena):
        campos = cadena.split()
        mes = {
               'enero':1,
               'febrero':2,
               'marzo':3,
               'abril':4,
               'mayo':5,
               'junio':6,
               'julio':7,
               'agosto':8,
               'septiembre':9,
               'octubre':10,
               'noviembre':11,
               'diciembre':12,
               }
        return str(campos[3]) + '-' + str(mes[campos[2]]) + '-' + str(campos[1])
```

En primer lugar hay que darle un nombre a la clase, conviene darle el mismo nombre que a la araña (spider).

```
class Betabeers(CrawlSpider):
    name = 'betabeers'
    allowed_domains = ['betabeers.com']
    start_urls = ['https://betabeers.com/community/events/?id=27&p=1']
```

En estas líneas se realiza la configuración básica de nuestro spider:

- **name**. Es el nombre interno que scrapy detectará como nombre del spider. Se recomienda darle el mismo nombre que a la clase.
- **allowed_domains.** Contiene un array con los dominios sobre los que scrapy podrá seguir links. De esta forma evitamos que siga enlaces fuera de la web que estamos rastreando.
- **start_urls.** Ahí debemos poner la(s) URL(s) sobre la(s) que queremos que empiece a rastrear

```
rules = [Rule(LxmlLinkExtractor(allow=(
                r'(.*)betabeers.com/community/events/\?id=27&p=(.*)',
            )), callback='parse_events', follow=True),
         ]
```

Aquí configuramos las “rules” o reglas que debe seguir nuestro spider para rastrear:

- **allow.** contiene un array con una serie de expresiones regulares para indicarle qué links puede seguir. Scrapy cuenta con un motor interno que evita que volvamos a pasar dos veces por la misma URL, pero es muy importante que sea la misma. En este caso de ejemplo, estamos permitiendo cualquier protocolo (http/https) y la url con o sin www. al principio. El querystring ‘p’ hace referencia en este caso al número de página, ya que en caso de haber muchos eventos se encuentran separados en varias páginas. Hay que tener cuidado a la hora de marcar la página de inicio, ya que si nos fijamos en *start_urls* hemos utilizado el querystring p=1. En la web de betabeers si no ponemos el querystring ‘p’ nos lleva a la página 1, pero cuando el spyder haya pasado a la página 2, tendrá un enlace a la página 1 con querystring p=1, lo que resultaría en recorrer la página 1 dos veces.
- **callback.** Es la función a la que se llamará para parsear el documento HTML. Se llama una vez por cada página visitada.

```
    def parse_events(self, response):
        eventos = response.xpath('//div[@id="tab_box_events"]/ul').xpath('li')
        for sel in eventos:
            item = ZticItem()
            item['organizador'] = 'Betabeers';
            item['fecha'] =
sel.xpath('div[@style="float:left;width:80px"]/text()').extract()[1].strip()
            item['horaInicio'] =
sel.xpath('div[@style="float:left;width:80px"]/text()').extract()[2].strip()
            item['link'] = sel.xpath('div/a/@href').extract()[0].strip()
            item['titulo'] = sel.xpath('div/a/text()').extract()[0].strip()
            item['direccion'] =
sel.xpath('div[@style="float:left;width:75%"]/text()').extract()[1].strip()
            request = Request(item['link'], callback=self.parse_event)
            request.meta['item'] = item
            yield request
```

Esta es la función encargada de parsear el evento. En la primera línea de la función lo que se hace es seleccionar el elemento que contiene el listado de eventos.

response contiene el documento HTML completo. En este caso busca en todo el documento HTML mediante la función xpath() el elemento HTML con id=”tab_box_events” y dentro de él, elemento <ul>. Un vez seleccionado, busca todos los elementos <li> dentro del anterior <ul>. Cada uno de estos <li> (list ítem) contiene los datos necesarios para un evento. Al iterar sobre la variable *eventos*, la variable *sel* para cada iteración contiene solamente el HTML contenido dentro de cada uno de los <li> anteriores. De esta forma se simplifica mucho la búsqueda de elementos.

item será la variable que almacenará los campos de un evento. En primer lugar tenemos que decirle de que tipo de Item se trata. En este caso, es de tipo ZticItem, que se encuentra definido en el fichero “ítems.py”. A continuación, vamos rellenando los campos que debe tener un evento. El organizador, se introduce esta vez estáticamente, ya que siempre trabajamos con eventos de Betabeers. El resto de campos se buscan mediante XPath en el HTML de cada <li>.

Existen algunos campos que no se pueden obtener directamente del listado. Es el caso del campo fechaFin. Para conseguir este campo, tenemos que seguir el enlace que nos lleva a la página del evento. Esto se consigue mediante la antepenúltima línea. En esta línea, mediante la función Request, le decimos como primer parámetro qué URL debe visitar y como segundo parámetro *callback* la función que se encargará de parsear dicha URL.

Finalmente, tenemos que añadir a la variable *request* el ítem que hemos ido generando para que scrapy pueda pasárselo a la segunda función y finalmente para que scrapy se encargue del postprocesado del ítem (almacenarlo en la Base de Datos).

```
def parse_event(self, response):
    item = response.meta['item']
    fecha = response.xpath('//div[@id="tab_box_info2"]/p/text()')[0].extract()
    item['horaInicio'] = fecha.split(',')[1].split('-')[0].strip()
    item['horaFin'] = fecha.split(',')[1].split('-')[1].strip()
    item['fecha'] = self.convierte_fecha(fecha.split(',')[0])
    yield item
```

Esta función es la encargada de parsear la URL que hemos seguido del listado anterior que contiene los detalles de un evento. El funcionamiento es similar al de la función anterior, solo que ésta vez devolvemos el ítem y scrapy se encarga de buscar en el fichero pipelines.py la función ZticPipeline() que se encarga del postprocesado del elemento. En este caso, de almacenar el evento en la Base de Datos si no existía previamente.

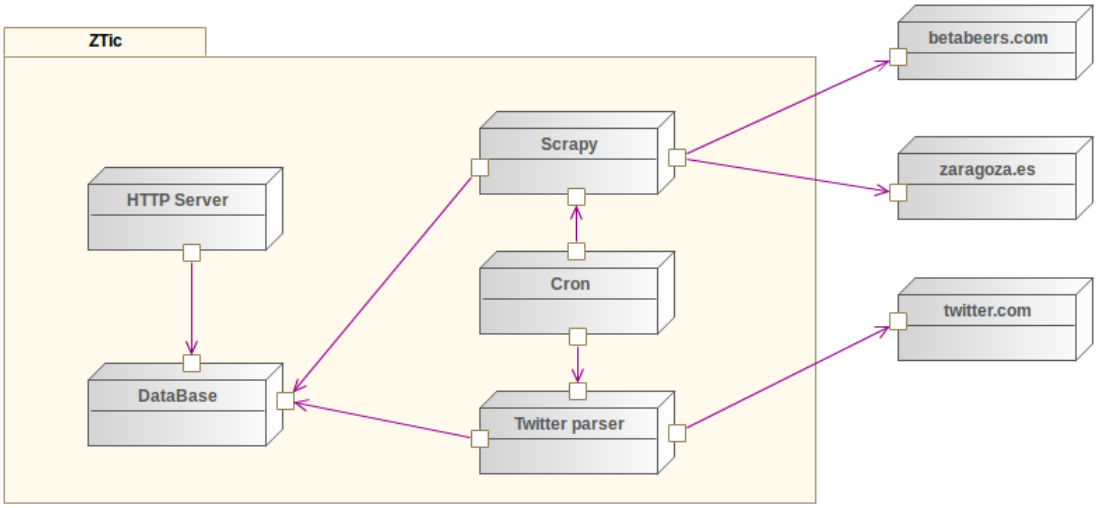
```
def convierte_fecha(self, cadena):
    campos = cadena.split()
    mes = {
           'enero':1,
           'febrero':2,
           'marzo':3,
           'abril':4,
           'mayo':5,
           'junio':6,
           'julio':7,
           'agosto':8,
           'septiembre':9,
           'octubre':10,
           'noviembre':11,
           'diciembre':12,
           }
    return str(campos[3]) + '-' + str(mes[campos[2]]) + '-' + str(campos[1])
```

Finalmente, se muestra el ejemplo de una función auxiliar que puede venir en el mismo fichero que puede ser muy útil para adaptar algunos datos obtenidos de la web. En este caso, nos encargamos de adaptar las fechas al formato de MySQL y convertir los meses que en la web están en texto en castellano a su número de mes.

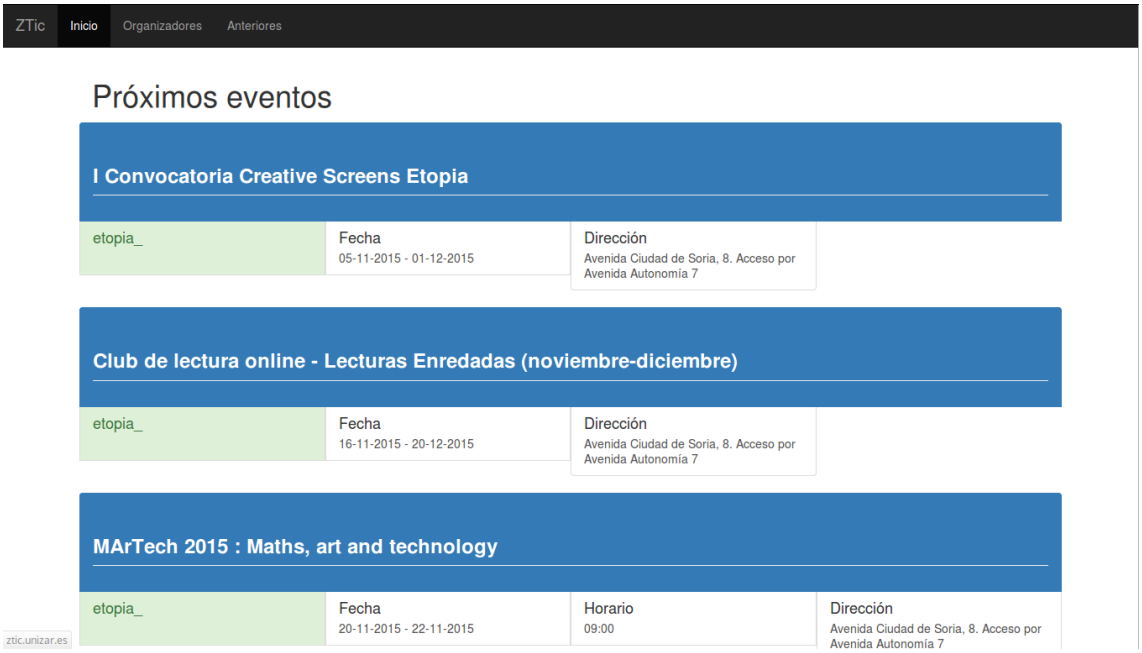
Anexo III. Diagrama de componentes

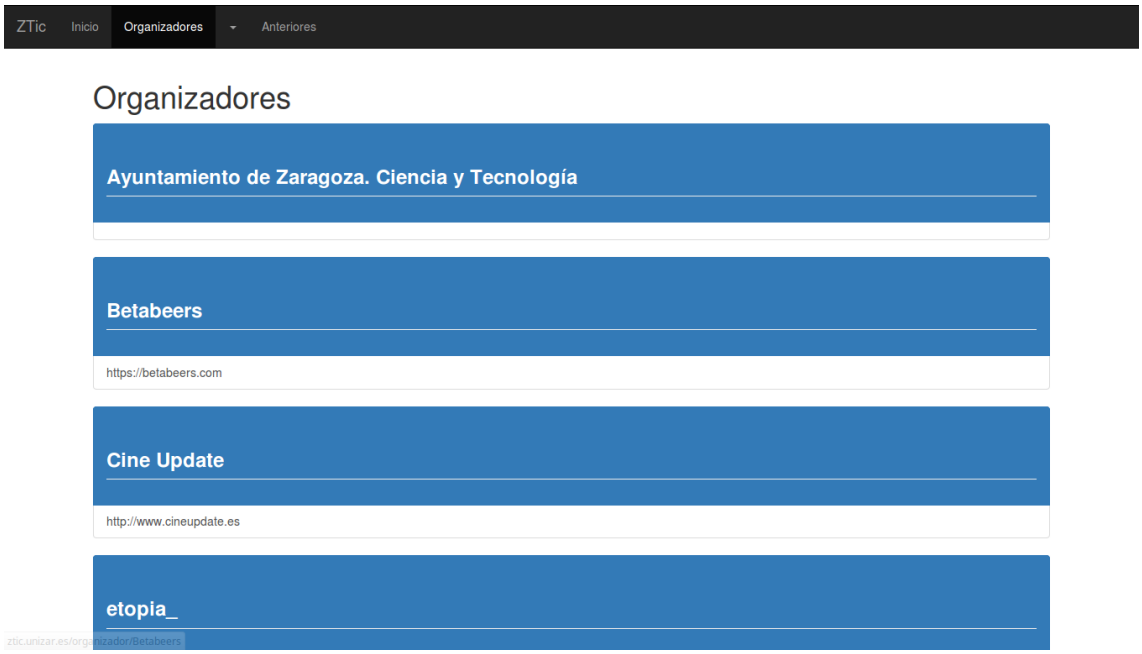
Este diagrama muestra las relaciones que existen entre los diferentes componentes del sistema. Las interacciones que existen son:

- La Base de Datos es atacada por el servidor HTTP en modo lectura.
- Scrapy y el parser de Twitter atacan a los servidores de las webs externas, transforman los datos y finalmente, guardan en la Base de Datos los eventos.
- Cron se encarga de llamar a los procesos de Scrapy y Twitter parser para que se ejecuten automáticamente con una periodicidad determinada.



Anexo IV. Ejemplo de formato de la web





Anexo V. Manual de uso de la web

Este anexo tiene como objetivo ofrecer un breve manual del usuario de la Web.

La web consta de básicamente de tres tipos de páginas:

- **Páginas estáticas.** Son las páginas de "Acerca de" y de "Cómo publicar en Twitter". Se trata de páginas con contenido estático en HTML que ofrecen información acerca de los creadores y un breve manual de cómo publicar eventos en Twitter, respectivamente. Ambas dos son accesibles desde el *footer* de la página, presente en todas las páginas del sitio.
- **Página de organizadores.** Contiene un listado con todos los organizadores de eventos registrados en el sistema. Debajo del título del organizador se muestra un enlace a la URL de éste. El enlace a esta página lo podemos encontrar en el menú superior de la web. Al hacer click sobre el título de un organizador nos lleva a la página de eventos de dicho organizador.
- **Páginas de eventos.** Son páginas dinámicas que muestran listados de eventos. Estas páginas son similares entre sí. Lo que las diferencia son los eventos que se muestran en cada una. Los eventos se encuentran filtrados según la página en que nos encontremos. Nos encontramos con tres tipos de filtros a los eventos:
    - Página inicio o próximos eventos: nos muestra el listado de los eventos que tuvieron lugar desde 15 días atrás y todos los eventos futuros. Esta página la tenemos accesible desde el menú principal situado en la parte superior de la web.
    - Página de eventos anteriores: nos muestra el listado de los eventos que ya han pasado y que se pueden consultar a modo de histórico. Tablién disponemos de un enlace llamado "Anteriores" en la barra superior de menú.
    - Página de eventos de un organizador: nos muestra los eventos pasados y futuros ordenados por fecha decreciente de todos los eventos organizados por éste. En la barra superior aparece el título del organizador y haciendo click sobre él nos lleva a la página web del organizador.

  Cada evento consta de cuatro partes diferenciadas:

    - Título del evento, al hacer click sobre él, se abrirá en una pestaña nueva a la página web externa del evento con información ampliada sobre éste.
    - Dirección del evento, al hacer click se abre en una pestaña nueva la posición del evento en Google Maps.
    - Fecha y horario del evento. Muestran la a fecha o el período que dure el evento y la hora de inicio u horario. Si hacemos click nos crea una

plantilla de Google Calendar para agregar el evento a nuestro calendario de Google.