Ingeniería Web – Proyecto Web Colaborativo

Reto 3: Gestión de proyectos

Curso: 2º Grado en Industria Digital (Semestre 2º)

Materia: Ingeniería Web

Estudiantes: Asier Sojo Álvarez de Eulate

Unai Gibello González

Grupo: IW-04

Profesor: Jon Vadillo Romero

Facultad de Ingeniería

UNIVERSIDAD DE DEUSTO

VITORIA - GASTEIZ, MAYO DE 2020

Índice

1 CAPÍTULO 1: INTRODUCCION

Este proyecto de web colaborativo pretende dar solución al siguiente problema presentado para simular una necesidad real de una empresa dedicada al desarrollo proyectos. La situación que se presenta es la siguiente:

El área de IT de la empresa Deustotil Tech S.L. ha decidido subcontratar el desarrollo de un software para la organización y planificación de proyectos de la empresa. Hacía tiempo que los responsables de proyecto demandaban una herramienta para gestionar mejor los proyectos y se ha decidido encargar a nuestro equipo su desarrollo.

Se trata de una empresa de servicios de consultoría en la cual se trabaja por proyectos. Es decir, el cliente contacta con la empresa, se acuerda la realización de un proyecto con fechas de inicio, fin y presupuesto fijado, y la empresa encarga el proyecto a un jefe de equipo. El jefe de equipo define las tareas del proyecto y las va asignando. La idea es que el nuevo software ayude a realizar la gestión necesaria, en la cual interesa registrar especialmente la información referente a proyectos, tareas de estos, los empleados y los clientes de la empresa.

2 CAPÍTULO 2: OBJETIVOS DEL PROYECTO

2.1 TAREAS PRINCIPALES

- ***Gestión de proyectos***: creación, visualización (listado y detalle), modificación y baja.
- ***Gestión de tareas***: creación, visualización (listado y detalle), actualización e incluir notas).
- La empresa también está abierta a cualquier funcionalidad extra que se quiera incorporar:
  - ***Gestión de empleados***
  - ***Gestión de clientes***

2.2 PLANIFICACION TEMPORAL

TAREA	Horas planif.	Horas Reales	Semanas								Observaciones
			30/mar	06/abr	13/abr	20/abr	27/abr	04/may	11/may	18/may	
Estructura Django	25		15	10	0	0	0	0	0	0	Las horas planificadas en ciertas semanas difieren de las reales debido al desconocimiento en algunas de las tareas encomendadas. En otros casos las horas planificadas fueron más que las invertidas finalmente, aunque no fuese lo habitual. Algunas tareas han podido llevar más trabajo por la búsqueda de información o por el propio proceso de aprendizaje.
		25	10	15							
HTML	16		0	10	0	0	0	0	3	3	
		18		8					5	5	
CSS	18		0	0	6	6	6	0	0	0	
		12			4	4	4				
JS / Ayax	32		0	0	0	0	0	12	12	8	
		40						15	15	10	
Documentación	13		0	0	0	0	0	0	3	10	
		14							2	12	
TOTAL	104	109	15	20	6	6	6	12	18	21	
			10	23	4	4	4	15	22	27	

3 CAPÍTULO 3: ESPECIFICACION DE REQUISITOS DEL SISTEMA

3.1 CATALOGO DE REQUISITOS

Para la primera entrega se buscaba una primera estructura de la aplicación en el entorno de desarrollo django. En una primera instancia toda la apariencia de los datos de la web se realizo mediante los forms de html y CSS.

Para esta entrega final del proyecto, se nos pide cambiar el método de acceso a los datos en la BBDD. Previamente a implementar JavaScript, la aplicación ya era capaz de gestionar completamente los objetos, crear, mostrar, actualizar y borrar. Sin embargo, tal y como pasa hoy en día en los desarrollos, llega una nueva tecnología, método o lenguaje y es necesario adaptarse a dicha novedad. En nuestro caso se trata del lenguaje JavaScript y metodología AJAX. Esta metodología permite una conexión asíncrona con el servidor web, haciendo que la web, que carga su contenido dinámico vía JavaScript, realice peticiones al servidor y éste le devuelva las consultas. Todo esto, a ojos del usuario no cambia nada, sin embargo, hace que la navegación y manipulación de webs dinámicas sea mucho más rápida y sencilla ya que la web no necesita actualizarse ni cambiar la URL, lo que en verdad cambia es el DOM.

Sabiendo esto, como requerimientos tenemos los siguientes:

- Carga y modificación de DOM de datos mediante JavaScript, empleando el método *fetch* sobre una API previamente desarrollada.
- Envío de datos de un formulario mediante AJAX para su almacenamiento en BBDD.
- Añadir una funcionalidad extra que emplee JavaScript. En nuestro caso, se trata de un select, que según qué opción esté escogida, muestra una tarea con la prioridad seleccionada. Esta acción se realiza sin recargar la web, simplemente refrescando el DOM.
- Más funciones Python: En la entrega anterior, se incluyeron ya varias funcionalidades extra, adelantándonos de alguna manera a este requisito. Estas funciones, que ya estaban previamente fueron:
    - Envío de mails
    - Envío de mensaje WhatsApp
    - Carga de imagen en un campo habilitado para ello. En nuestro caso, foto de perfil para cada empleado.

3.2 DESCRIPCION DE LA INTERFAZ DEL SISTEMA

La aplicación, mediante las diferentes secciones, permite al usuario visualizar los elementos que seleccione. Podrá ver en un listado todos los elementos que forman parte del sistema. Con un formulario sencillo, que se despliega con un botón, podrá crear y añadir a la BBDD un nuevo elemento.

3.2.1 PERFIL DE LOS USUARIOS

Consideramos que la gestión de nuestra aplicación no requiere de un perfil muy alto. Cualquier persona con los conocimientos básicos de navegación web podrá adaptarse y emplear sin problemas nuestra aplicación.

4 CAPÍTULO 4: ESPECIFICACION DEL DISEÑO

4.1

4.1.1 PRINCIPALES FUNCIONES DEL SOFTWARE

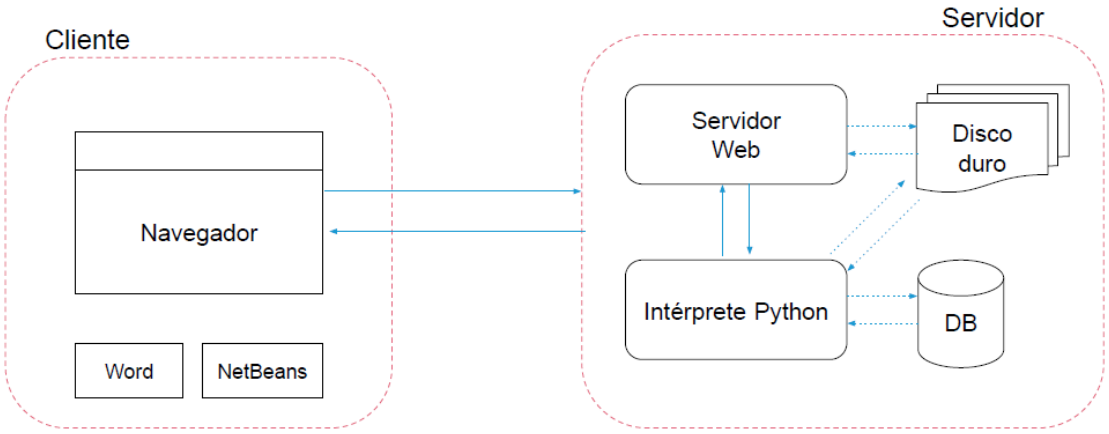
Visualizar tareas, empleados, clientes y proyectos. Una vez que se visualizan, se permite ver en detalle, con todos los campos disponibles de cada objeto. También permite dar de alta nuevos elementos que se guardarán en la BBDD.

4.1.2 DESCRIPCION DEL ENTORNO DE DESARROLLO

El entorno de Desarrollo (IDE) empleado para desarrollar todo el Código ha sido PyCharm, con licencia Educativa. A través de este software hemos podido desarrollar sin problemas todo el Código en Python, HTML, CSS y JavaScript.

4.2 ARQUITECTURA FISICA Y ENTORNO TECNOLOGICO

4.2.1 DESCRIPCION GENERAL



4.3.1 ESPECIFICACION DE LAS INTERACCIONES

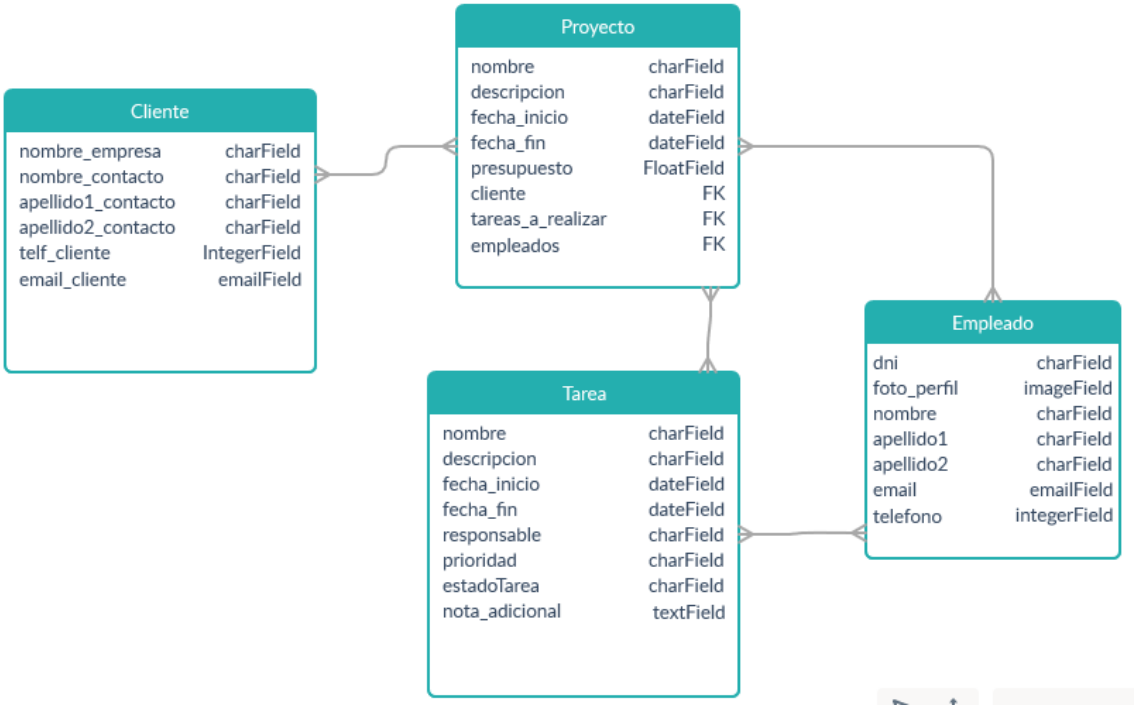
Una vez que tenemos nuestro servidor activo gracias el comando *"python manage.py runserver"* el acceso a nuestra aplicación web ya está habilitado, siendo la página de inicio la URL: *http://127.0.0.1:8000/index/home/*. Esta página se basa en dos archivos HTML, el index, que siempre estará fijo, y el especifico de esta dirección. La única carga de datos e interacción con el servidor que se realiza en esta URL es el contar cuantos empleados, clientes y proyectos hay en la BBDD. Para ello se emplea este HTML:

```
<section class="seccionesPortada">
  <img id="imagenProyectos" src="{% static 'img/imagenProyectos.jpg'
%}">
  <p class="textoPortada">Actualmente, en nuestra empresa tenemos {{
numProyectos }} proyectos en desarrollo</p>
</section>
<section class="seccionesPortada">
   <img id="imagenClientes" src="{% static 'img/imagenClientes.jpg'
%}">
   <p class="textoPortada">En nuestra cartera, contamos con {{
numClientes }} clientes de todo tipo</p>
</section>
<section class="seccionesPortada">
  <img id="imagenEmpleados" src="{% static 'img/imagenEmpleados.jpg'
%}">
  <p class="textoPortada">En nuestra plantilla contamos con {{
numEmpleados }} empleados de alta cualificación</p>
</section>
```

Cualquier modificación del número de estos objetos se verá reflejado de manera automática en esta URL.

Para las URLs que listan los objetos de la BBDD, se emplea un script de JavaScript. Dicho script recoge todos los elementos de la API, que es un archivo JSON, y genera una tabla con unos campos genéricos del objeto. Si el usuario clica en el icono de ver más detalles, accederá al usuario con el ID seleccionado. En esta nueva URL, ya se muestran en una tabla todos los elementos que componen el objeto, es decir, es una vista detalle de dicho objeto.

4.3.2 DISEÑO DE LA ESTRUCTURA FISICA DE LOS DATOS



4.3.3 DEFINICION DE VISTAS

El diseño de la aplicación está montado sobre un HTML llamado index. Este archivo se emplea como base sobre todas las vistas. El html index posee la cabecera de la aplicación, un nav con las URLs necesarias para navegar sobre toda la web, títulos de página dinámicos según el contenido que cargue y a pie de página un footer.

Sobre este HTML se mandarán y se estructurarán los demás como por ejemplo “lista_clientes.html”. El contenido cargado proveniente del servidor, se aloja y adquiere todas las características del diseño de la web, configuradas en el archive css “main.css”.

5 CAPÍTULO 5: INCIDENCIAS DEL PROYECTO Y CONCLUSIONES

Muchas de las funcionalidades que se nos requerían desde el inicio del proyecto han sido explicadas en clase, de modo que no han surgido muchos problemas con ellas. A medida que nos veíamos más cómodos con Python, quisimos ampliar las funcionalidades, investigando y desarrollando código descubríamos de qué éramos capaces.

Por esta razón, vimos que añadir la funcionalidad de envió de emails y de mensajes de WhatsApp era bastante asequible. Sin embargo, crear un campo nuevo para que los empleados contasen con una foto de avatar trajo bastantes quebraderos de cabeza. Un imageField requería cambiar y habilitar secciones nuevas en el archive settings, también habilitar una nueva ruta y carpeta donde alojar esos archivos. Sin llegar a exagerar, esto supuso dos días de visualización de tutoriales, adecuar esos tutoriales a nuestro Código y hacer que funcionase sin ningún tipo de problemas.

Con esta última entrega no hemos estado tan cómodos con las funcionalidades a añadir. Ha supuesto trasladar todo lo hecho previamente a una nueva metodología de programación como AJAX, y un nuevo lenguaje, JavaScript. La carga y muestra de datos ya no era tan sencilla, había que desarrollar una API con los JSON de cada objeto, y conteniendo esta API, había que desarrollar el algoritmo JS que generase una tabla HTML con los objetos. Para hacer su navegación más sencilla, se creó un botón que mostraba y ocultaba el formulario de creación de un objeto nuevo. Con todo este formulario ya desarrollado no hemos sido capaces de lograr la function POST correctamente. Se envían los datos de una tarea, pero tras enviar el formulario se muestra el JSON generado, volviendo a la app, esa misma tarea sí que aparece en la BBDD y se muestra ya listada.