Grupo3 Documentación

Ingeniería Web

Curso 2023 - 2024

20/05/2024

—

Aitor Salazar
Iker Martin
Iker Larrazabal

Índice

Índice figuras

1. Introducción

Este documento contiene la información detallada sobre el proyecto realizado, con esta documentación queremos proporcionar una información lo más detallada posible sobre el proyecto, abordaremos desde el diseño del proyecto hasta la implementación final, pasando por la arquitectura, diagramas, tareas realizadas, etc…

2. Objetivos del proyecto

2.1. Resumen para la dirección

Nosotros hemos elegido hacer una aplicación que permita gestionar los proyectos de la empresa Deusto Tech S.L.
Esta aplicación está basada en Django, un framework de desarrollo web en Python, y utilizará JavaScript (JS), CSS y HTML para la parte visual y la interacción con el usuario.

2.2. Tareas principales

Análisis y diseño de la aplicación:

- Definir la estructura de la base de datos y los modelos de datos necesarios para almacenar la información de Clientes, Prioridad, EstadoTarea, Empleado, Proyecto y Tarea.
- Diseñar la interfaz de usuario para los modelos que vamos a gestionar en la aplicación.
- Identificar las funcionalidades extras que queremos implementar y evaluar si son viables y cómo vamos a hacerlas.
- Configurar el servidor y la base de datos.

Desarrollo del backend de la aplicación:

- Crear los modelos de datos y establecer las relaciones entre ellos.
- Implementar las operaciones CRUD (crear, leer, actualizar y eliminar) para los modelos que queremos gestionar.
- Desarrollar la estructura de clases y las rutas necesarias para gestionar las peticiones del cliente.

- Comprobación y procesamiento de los campos antes de su envío al servidor.
- Envío de datos de un formulario mediante AJAX para su almacenamiento en BBDD y posterior visualización del resultado.

Desarrollo del frontend de la aplicación:

- Crear las diferentes páginas HTML para visualizar las diferentes acciones de los modelos a gestionar, además del archivo CSS para aplicar estilos a estas páginas HTML.
- Añadir eventos a diferentes componentes de las páginas HTML para que estos tengan funcionalidad a través del archivo JavaScript.
- Capturar eventos DOM de las páginas HTML.

2.3. Planificación temporal

Este es un resumen del reparto de tareas del equipo:

- Aitor:
  - Diseño e implementación de la estructura de clases con las operaciones visualizar y crear en todos los modelos.
  - Creación de todas las estructuras HTML.
  - Paginación del listado de los modelos.
  - Envío de emails con JavaScript.
  - Implementación de login con protección de vistas.
- Iker:
  - Diseño e implementación de la estructura de clases con las operaciones actualizar y borrar en todos los modelos.
  - Implementación de CSS.
  - Creación de API y utilizado Fetch para obtener los datos y mostrarlos en el DOM de la ListView de cliente.
  - Comprobación de algunos campos con JavaScript antes del envío al servidor.
  - Capturar eventos DOM y alterar la página con JavaScript.
  - Añadir eventos JavaScript a diferentes componentes.

3. Especificación de requisitos del sistema

3.1. Catálogo de requisitos

Funciones y requisitos mínimos que nos han encargado a nuestro equipo de desarrollo de software:

1. Gestión de proyectos:
    a. Creación de proyectos con los siguientes campos: nombre, descripción, fecha de inicio, fecha fin , presupuesto, cliente y responsable.
    b. Visualización del listado de equipos
    c. Ver el detalle de cada proyecto con toda su información.
    d. Edición de proyectos
    e. Dar de baja a los proyectos

2. Gestión de tareas:
    a. Creación de tareas con los siguientes campos: nombre, descripción, fecha de inicio, fecha fin, responsable, proyecto, nivel de prioridad, estado tarea y notas
    b. Visualización del listado de tareas.
    c. Ver el detalle de cada tarea con toda su información.
    d. Edición de tareas.
    e. Dar de baja a las tareas
    f.
3. Gestión de empleados:
    a. Creación de tareas con los siguientes campos: dni, nombre, apellidos, email y teléfono
    b. Visualización del listado de empleados.
    c. Ver el detalle de cada empleado con toda su información.
    d. Edición de empleados.
    e. Dar de baja a los empleados

4. Gestión de clientes:
    a. Creación de tareas con los siguientes campos: nombre, teléfono y dirección
    b. Visualización del listado de clientes.
    c. Ver el detalle de cada cliente con toda su información.
    d. Edición de clientes.
    e. Dar de baja a los clientes

5. Requisitos generales:
    a. El dni del empleado debe de ser único
    b. Las tareas tienen 4 estados: abierta, asignada, en proceso y finalizada
    c. Debe de existir un sistema de niveles de prioridad en las tareas(alta, media, baja).
    d. La aplicación debe permitir la asignación de empleados(responsables) a las tareas y poder realizar notas
    e. La aplicación debe permitir la inclusión de la tarea en un proyecto concreto
    f. La aplicación debe de permitir la inclusión de un cliente en un proyecto
    g. La aplicación debe de permitir la inclusión de un empleado(responsable) en un proyecto

Inconvenientes actuales del cliente y necesidades de la nueva aplicación que el cliente nos ha transmitido:

- Actualmente, el cliente no tiene ninguna herramienta o software con el que puedan gestionar los diferentes aspectos claves de los proyectos.

- La falta de un sistema centralizado dificulta el seguimiento y la asignación de tareas a los empleados.

- El cliente necesita una solución más eficiente y organizada para gestionar los proyectos de sus clientes.

- El cliente busca mejorar la comunicación interna y agilizar el proceso de resolución de proyectos, evitando la duplicación de tareas.

3.2. Modelo lógico de datos

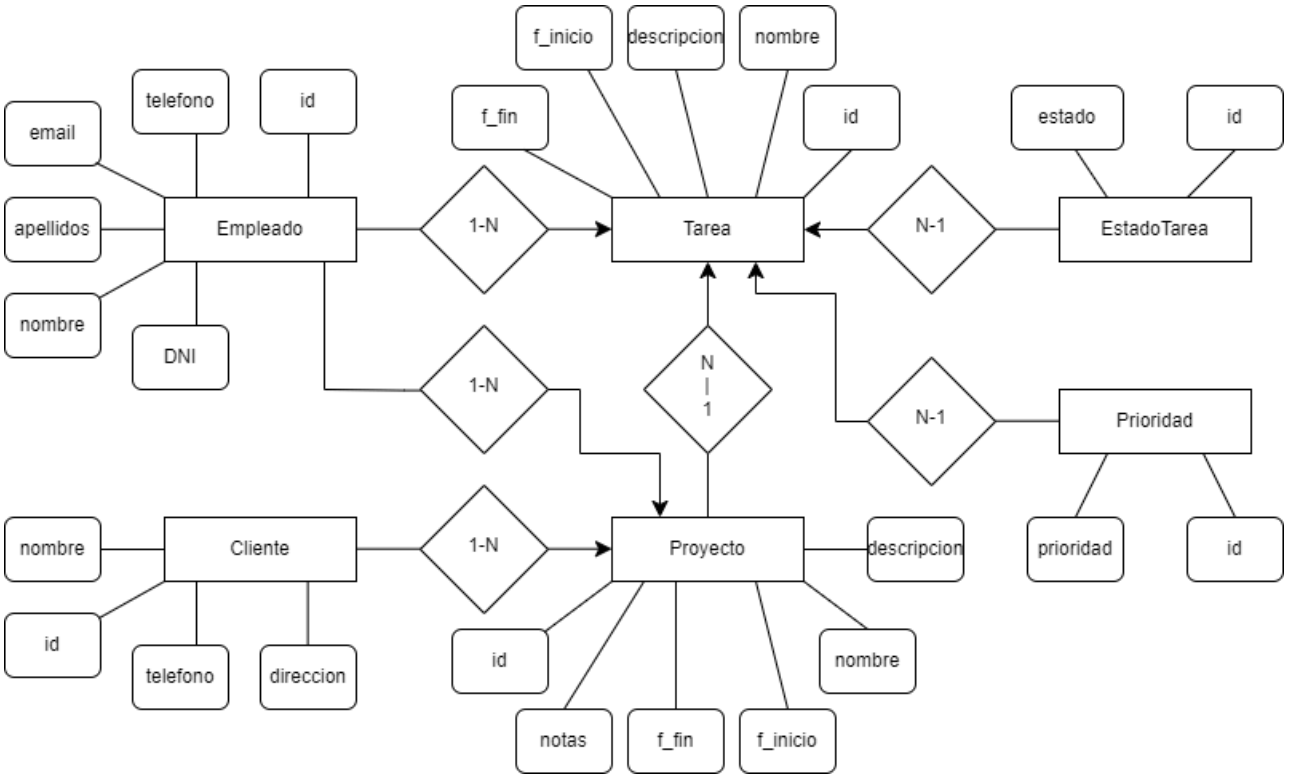

Figura 1: Diagrama ER

3.3. Descripción de la interfaz del sistema

En la aplicación, hay 4 tipos diferentes de vistas:

- **Login:** Esta vista es la primera vista que se visualiza nada más arrancar la aplicación y es obligatoria si quieres acceder a las demás.
- **Vista principal:** Está vista es la que se visualiza nada más loguearte. Contiene cuatro tablas con los datos más relevantes de cada una con un par de botones con los que puedes interactuar para ocultar la tabla o redireccionarte a una vista concreta.
- **Subvistas:** El diseño de las vistas es completamente igual en todas ellas y su única diferencia es la información que se muestra en cada una. En todas ellas se muestran todos los campos del modelo en cuestión con una paginación para que la visualización de la información sea más fácil. Aparte dos de las subvistas cargan y muestran los datos de una API que contiene la información.

- **Correo:** Vista con un formulario sencillo que envía un correo a un mail concreto. Contiene un formulario con asunto y mensaje.

Todas las vistas excepto la del login tienen un menú de navegación superior e inferior con el que puedes cambiarte de vistas.

3.4. Representación jerárquica de la interfaz

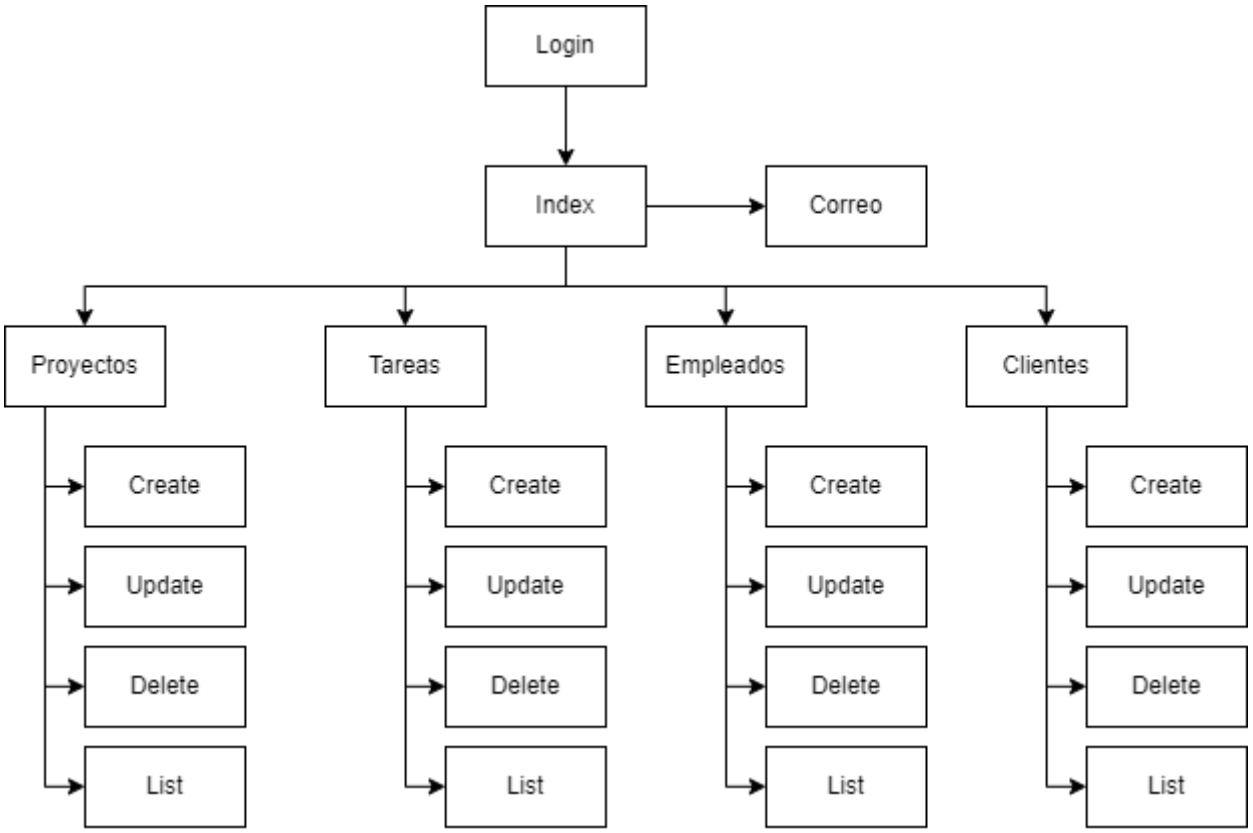

Figura 2: Diagrama transición estados

4. Especificación del diseño

4.1. Principales funciones del software

Las principales funciones del software que se van a diseñar son:

- Cambio de tamaño de elementos: Al hacer click en los botones disminuir tamaño y aumentar tamaño, se producirá el aumento o disminución del tamaño de los elementos en la página index.html.

- Validación de campos de formulario: Antes de enviar un formulario al servidor, se realizará la validación de campos como fecha de inicio y fecha de fin para asegurar que cumplan con los requisitos establecidos, en caso de no estar correcto alguno de estos campos, crearemos un mensaje en rojo señalando el porqué del error del envío.

- Captura de eventos y cambios en el DOM: Se capturaron eventos en el DOM y se realizaron cambios en el estilo o contenido de la página, por ejemplo, ocultando o expandiendo las tablas del index.html mediante un botón llamado ocultar y expandir.

- Carga y modificación de datos mediante JavaScript: Se utilizará la llamada a la API de Django mediante la función Fetch para obtener datos y mostrar los valores modificando el DOM en el modelo cliente.

- Envío de emails: Hemos implementado la funcionalidad de envío de emails desde la aplicación, permitiendo enviar notificaciones o mensajes a nuestro correo para que los clientes puedan contactarnos con cualquier duda, sugerencia, problema….

- Paginación en tablas/listados: Hemos proporcionado la opción de paginar los resultados del listado de todos los modelos, dividiendo el contenido en páginas para una mejor visualización y navegación, en este caso hemos optado por mostrar 10 por cada página.

- Autenticación de usuarios: Hemos permitido a los usuarios realizar el inicio de sesión en la aplicación mediante autenticación, brindando seguridad y control de acceso a funcionalidades específicas.

4.2. Descripción del entorno de desarrollo

Como entorno de desarrollo o IDE para escribir y gestionar el código de la aplicación hemos utilizado Visual Studio Code. La aplicación ha sido desarrollada utilizando Django, un framework de desarrollo web en Python, también hemos utilizado JavaScript (JS), CSS y HTML para la parte visual y la interacción con el usuario.

También hemos utilizado SQLite como base de datos, ya que es la que viene por defecto con Django además de ser una base de datos ligera y fácil de usar que se almacena en un archivo local .sqlite3. Django también nos

incluye su propio servidor web de desarrollo llamado "Django Development Server" o "manage.py runserver", en el cual podemos hacer pruebas locales del código de la aplicación Django, este servidor es muy fácil de usar y se ejecuta directamente desde la línea de comandos utilizando el comando "python manage.py runserver". El servidor se ejecuta en el puerto 8000 y podemos ver nuestra aplicación en funcionamiento en la dirección "http://localhost:8000".

Para el control de versiones hemos utilizado Git que nos permite realizar un seguimiento de los cambios, colaborar entre nosotros en el mismo repositorio y revertir modificaciones si es necesario.

4.3. Representación gráfica cliente-servidor

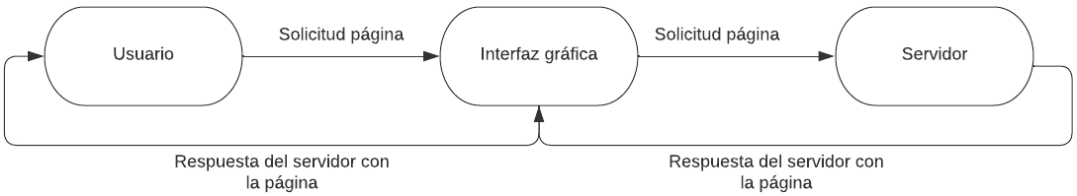

Figura 3: Interacción cliente-servidor

4.4. Representación gráfica ER

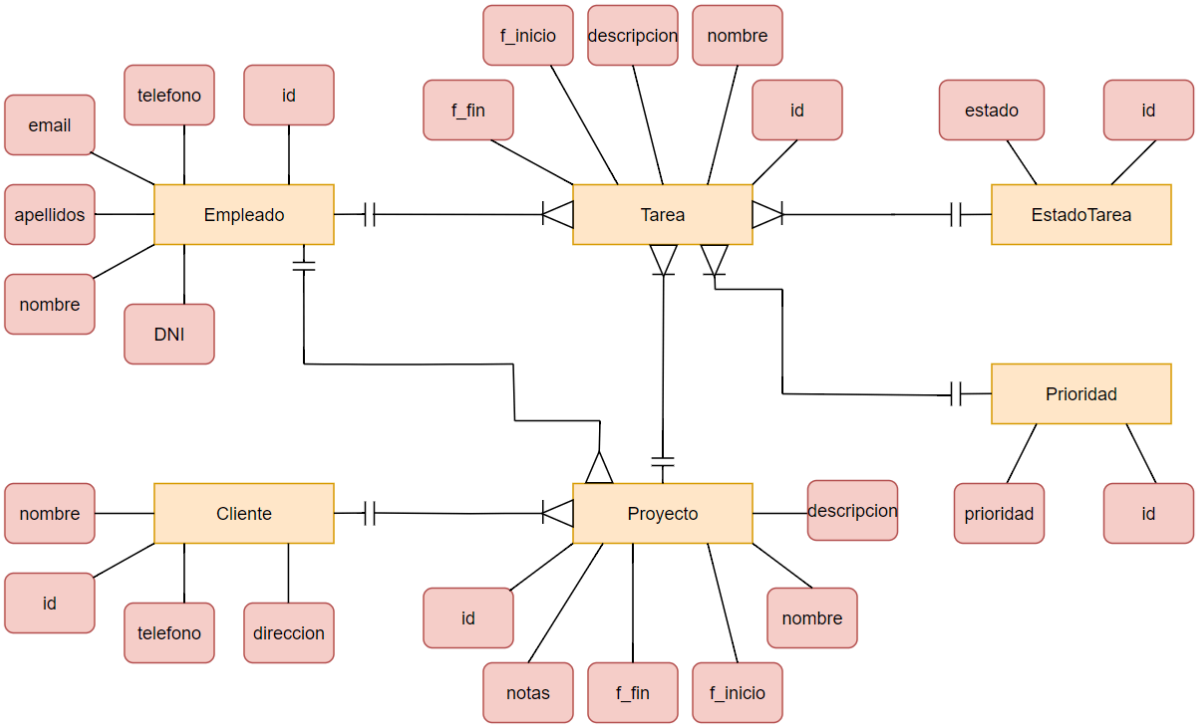

Figura 4: Diagrama ER BBDD

- Empleado → Representa a los empleados de la empresa. Tiene una relación de uno a muchos con la entidad Tarea, ya que un empleado puede participar en múltiples tareas de la empresa, de igual manera ocurre con la entidad Proyecto.
- Tarea → Representan las tareas que componen los proyectos de la empresa. Tiene 4 relaciones mucho a uno con las entidades Prioridad, EstadoTarea, Proyecto y Empleado. Esto se debe a que cada una de las entidades forman parte de muchas tareas.
- Prioridad → Representa el tipo de prioridad requerida para una tarea. Tiene una relación de uno a muchos con la entidad Tarea, ya que un tipo de prioridad se puede asignar a varias tareas.
- EstadoTarea → Representa el estado de la tarea respecto a la entidad Tarea a la que está asociada. Tiene una relación de uno a muchos con la entidad Tarea, ya que un estado de la tarea se puede asignar a varias tareas.
- Cliente→ Representa al cliente que ha solicitado el proyecto. Tiene una relación de uno a muchos con la entidad Proyecto, ya que un cliente se puede asignar a varios proyectos.

- Proyecto → Representa el proyecto que nos ha solicitado el cliente y en el que trabajarán nuestros empleados realizando tareas. Tiene dos relaciones mucho a uno con las Entidades Cliente y Empleado y una relación de uno a muchos con la Entidad Tarea.

5. Incidencias del proyecto y conclusiones

5.1. Incidencias del proyecto

- Problemas con el fetch al obtener los datos de la API.
- Problemas con la paginación en la función de la API.
- Problemas de conflictos cuando alguien bajaba los cambios que había subido otro integrante del grupo.
- Problemas para enviar un email.

5.2. Mejoras detectadas para futuro

- Creación de diferentes roles de usuario que tengan limitaciones de funcionalidad.
- Funcionalidad de búsqueda avanzada, para que los usuarios puedan filtrar, ordenar…
- Integrar servicios de 3 como iniciar sesión con cuenta de Google, Google Maps, APIs de otras aplicaciones…
- Chat en tiempo real, disponible para los usuarios.Personalización de los perfiles.
- Implementar la posibilidad de arrastrar y soltar archivos para facilitar la carga de archivos de la página.

5.3. Conclusiones

Para concluir, hemos cumplido con los objetivos planteados, hemos aprendido que el trabajo tiene varios aspectos clave, como la importancia del diseño y arquitectura de la página web; cómo interactúan los elementos entre sí, la importancia de tener un buen flujo en nuestra página, la presentación de la información al usuario… Otro aspecto importante es la importancia de desarrollar un front-end que sea interactivo, fácil de usar y agradable a la vista para el usuario.

Por supuesto, la importancia de tener un buen diseño de base de datos y una buena lógica de negocio para poder consultar y presentar la información de la mejor manera posible.

6. Bibliografía

- Django. (s.f.). Recuperado de https://docs.djangoproject.com/en/4.2/ref/templates/language/
- Django pagination. (s.f.). Recuperado de https://docs.djangoproject.com/en/4.2/topics/pagination/
- Pagination in Django using JavaScript. (s.f.). Recuperado de https://www.keysolutionsin.com/post/how-to-make-pagination-in-django-using-javascript
- Jon Vadillo. (13 de mayo de 2020). Cómo crear un API REST con Django [Video]. Recuperado de https://www.youtube.com/watch?v=XqRBb_4CLS4
- Vadillo, J. (16 de mayo de 2023). JavaScript fetch API example. Recuperado de https://github.com/jvadillo/js-fetch-api-example