Deusto

Universidad de Deusto
Deustuko Unibertsitatea

Ingeniería Web – Proyecto Web Colaborativo

Título: Gestión de la producción

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

Materia: Ingeniería Web

Estudiantes: Aitor López Roncero

Mikel Lerena Tapia

Grupo: IW-02

Profesor: Jon Vadillo Romero

Facultad de Ingeniería
UNIVERSIDAD DE DEUSTO

Vitoria - Gasteiz, mayo de 2020

RESUMEN

En esta memoria se va a tratar de explicar con la mayor precisión posible el funcionamiento de la aplicación web que se ha desarrollado a lo largo de las diversas entregas que ha tenido dicho proyecto.

Primeramente, se hará una breve introducción del trabajo y a lo largo de la memoria se irá comentando o explicando de forma que se recojan adecuadamente todos los puntos a tener en cuenta sobre el mismo.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

CAPÍTULO 1: INTRODUCCIÓN

Como ya se ha comentado anteriormente este proyecto se trata de crear una aplicación web. En este caso se va a orientar dicho proyecto a la gestión de la producción de una empresa, Deustubular S.L.

La empresa Deustubular S.L. quiere digitalizar parte de los procesos de su producción. Esta, cuenta con un amplio departamento de IT y se ha decidido que nuestro equipo se encargará de su desarrollo. La empresa quiere dejar de utilizar papel y poner un PC junto a cada puesto de trabajo, donde estará siempre abierta la aplicación web a desarrollar.

La idea es que el responsable introduzca los procesos a realizar desde la oficina y los operarios puedan ver un listado con los procesos cuando están junto a las máquinas (e incluso actualizar la información, por ejemplo, inicio y fin del proceso). Así se evitará tener que imprimir toda la información y llevarla a los puestos de trabajo.

Los procesos de la producción que se van a digitalizar serán:

Procesos: Creación, eliminación, edición, listar y detalles.
Empleados: Añadir, dar de baja, edición, listar y detalles.
Equipos: Creación, eliminación, edición, listar y detalles.

Y una vez conseguidos dichos objetivos, presentar la aplicación web para su valoración y su posterior utilización en la empresa.

2.1.-TAREAS PRINCIPALES

El OBJETIVO PRINCIPAL de este proyecto no es ni más ni menos que el de proporcionar a dicha empresa un medio mediante el cual poder gestionar procesos de su producción de una forma menos arcaica que la actual, refiriéndose con arcaica al empleo de informes en papel y de esta forma agilizar lo máximo posible los procesos productivos.

Aparte del objetivo principal con esta medida se consiguen objetivos de forma casi indirecta:

-LA NORMALIZACIÓN DE ARCHIVOS.

El archivo digital de documentos convierte toda la heterogeneidad de papeles en una base de datos estructurada y homogénea. Cada uno de esos papeles, en principio diferentes e incluso difíciles de manejar y archivar, se convierte en un registro digital único.

Puesto que todos los registros digitales son homogéneos, se establecerá un sistema de archivo y clasificación normalizado, en el que se podrá acceder a toda la información con diferentes criterios de búsqueda y filtrado, todo ello sin recorrer estanterías y sin abrir archivadores.

-AHORRO DE ESPACIO, MATERIAL Y RECURSOS.

El archivo digital reemplazará por completo al archivo físico. Esto significa que, incluso aquellos documentos que se tenga la obligación de conservar, se harán en formato digital, destruyendo el papel.

La ausencia de papel desencadena a su vez un importante ahorro en costes de papel, impresión, mobiliario, archivadores y diverso material de oficina, del que se puede prescindir casi en su totalidad.

-AHORRO DE TIEMPO, INCREMENTO DE EFICIENCIA.

Las búsquedas de papeles pueden llevar mucho tiempo, incluso en un archivo bien organizado. Cuando un papel no está exactamente en su sitio, ¿cómo podemos saber si realmente no lo tenemos, si alguien se lo ha llevado o si hay que seguir buscando en otras posibles ubicaciones? Después de invertir nuestro tiempo, puede que ni siquiera lleguemos a encontrar lo que buscamos.

Las búsquedas de documentos digitales son instantáneas y se realizan con total certeza. Si el documento existe, siempre lo vamos a encontrar porque los registros no pueden traspapelarse. Con una base de datos digital se ahorrarán muchas horas de trabajo relacionadas con la gestión del archivo físico y eliminaremos cualquier duda en una búsqueda.

2.2.-PLANIFICACIÓN TEMPORAL

En el caso de este nuestro proyecto, se han asignado dos puntos de control en los que la aplicación web ha tenido y va a tener distintas metas a cumplir, respetando siempre con el horario máximo de presentación de este.

En la primera presentación del proyecto, se ha mostrado la aplicación prácticamente finalizada salvando algunos detalles relacionados con la última presentación con fecha 24/05/2020.

Para esta última se han añadido funcionalidades relacionadas con JavaScript y demás.

Para la explicación de las funcionalidades se han reservado apartados a lo largo de la memoria en los que se irán explicando en profundidad.

NOTA: “No se ha podido seguir con la planificación designada al 100%, debido a que se han cometido errores y se han encontrado dificultades que han provocado una mayor tardanza en cuanto al cumplimiento de los objetivos. Sin embargo, se han conseguido de una manera eficiente.”

CAPÍTULO 3: ESPECIFICACIÓN DE REQUISITOS DEL SISTEMA

3.1.-INTRODUCCIÓN

3.1.1.- ALCANCE DEL PROYECTO

El alcance de la aplicación web se va a limitar a crear, eliminar, modificar, listar y detallar; los empleados, los procesos y los equipos.

Esto va a ser muy beneficioso para la empresa ya que se consiguen muchos objetivos indirectamente y que son muy importantes. Sin embargo, toda la funcionalidad relacionada con las órdenes de fabricación no se ha introducido en la aplicación, debido a que no se ha tenido el suficiente conocimiento como para poder aplicarlo de una forma eficiente y sin causar problemas a la hora de la ejecución de dicha aplicación web.

3.2.-DESCRIPCIÓN GENERAL

3.2.1.- CATÁLOGO DE REQUISITOS

Una de las necesidades que probablemente se tenga que implementar serán las ordenes de fabricación y su tratamiento.

Para ello se tendrá que buscar la forma óptima y rápida para implementarlo de tal forma que no se tengan posteriores problemas o que de alguna forma sean de rápida solución sin tener que emplear excesivos medios para su corrección.

Y en base a eso, las nuevas necesidades que podrán surgir será una mayor fuente de información de la que poder extraer los conocimientos necesarios para la implementación de dicha funcionalidad.

3.3.-DESCRIPCIÓN DE LA INTERFAZ DEL SISTEMA

3.3.1.- PERFIL DE LOS USUARIOS

En este caso solamente van a existir dos tipos de usuarios a la hora de la utilización de la aplicación web.

Estos van a ser los administradores y los invitados.

Los administradores tienen un libre control sobre la aplicación de tal forma que están a su disposición las funcionalidades de crear, eliminar, listar y detallar de los empleados, procesos y equipos.

Para que un usuario sea administrador tiene que introducir a la hora de iniciar sesión los siguientes datos:

Usuario: root
Contraseña: root

En caso de que se quiera entrar a la aplicación como invitado, esto conllevará una serie de restricciones que solo le permitirán leer la información.

NOTA: “Los usuarios que son invitados, se sobreentiende, que son miembros de la empresa y que tienen permiso para ver los datos, y respetar la confidencialidad de estos. Esta es una aplicación web exclusiva para esta empresa.”

4.1.-INTRODUCCIÓN

4.1.1.- PRINCIPALES FUNCIONES DEL SOFTWARE

Las funciones de la aplicación son las siguientes:

-En cuanto a procesos de producción y personal:

Procesos: Creación, eliminación, edición, listar y detalles.
Empleados: Añadir, dar de baja, edición, listar y detalles.
Equipos: Creación, eliminación, edición, listar y detalles.

-En cuanto usuarios:

Posibilidad de iniciar sesión por parte del usuario con Usuario: root y Contraseña: root para ser administrador y tener permiso para la completa utilización y la aplicación, y en caso de no iniciar sesión poder entrar como invitado con restricciones.

-En cuanto a emails:

Posibilidad del envió de emails a los usuarios que quieran con las novedades de la aplicación. Para ello habrá que pinchar en el enlace de “Contáctanos”.

4.1.2.- DESCRIPCIÓN DEL ENTORNO DE DESARROLLO

Para esta aplicación se ha empleado GitHub para poder subir los avances que se vayan haciendo en la aplicación. Además, se ha utilizado GitKraken mediante el cual se puede clonar el archivo que se tenga en GitHub y poder tenerlo en el ordenador

Con esto se consigue que un grupo de trabajo pueda tener el trabajo actualizado. Como cada persona del equipo va a ir avanzando con un tema probablemente distinto a los demás, de esta forma sube los cambios a GitKraken y automáticamente se suben a GitHub para que el resto del grupo lo pueda tener casi al instante.

Además, se ha utilizado el programa PyCharm para la programación de la aplicación y para poder iniciar el servidor de la aplicación. Esto último es fundamental, ya que sin iniciar el servidor la aplicación no puede funcionar, y además es necesaria para que los programadores puedan visualizar los cambios que han realizado en esta, aparte de usar Visual Studio Code.

4.2.-ARQUITECTURA FISICA Y ENTORNO TECNOLÓGICO

4.2.1.- DESCRIPCIÓN GENERAL

Para la realización de esta aplicación web, se han empleado todos los recursos proporcionados para la creación de esta. Estos recursos son PyCharm (para poder programar la aplicación), Visual Studio Code (para poder hacer pruebas antes de implementarlas en la aplicación), navegadores de internet (para comprobar que lo que se está programando funciona correctamente), nuestro portátil personal (para poder llevar todo esto a cabo) y todos los sistemas necesarios para la programación (Python, Django, JavaScript, HTML, CSS, AJAX, JSON y demás).

4.3.-DESCRIPCIÓN DEL DISEÑO

4.3.1.- ESPECIFICACIÓN DE LAS INTERACCIONES

En todas las pestañas de la aplicación se va a respetar el mismo header, nav y footer, de esta forma se consigue una limpieza y un orden mayor que si tuviesen cada uno los suyos propios.

En el header se encuentran el logo que permite volver a la página principal directamente, el botón de atrás que permite volver a la anterior pestaña en la que se estaba y el botón de LogOut el cual permite cerrar sesión a los usuarios y llevarlos a la pestaña de iniciar sesión.

En el nav hay un menú a través del cual se puede navegar por la sección que uno quiera de la aplicación web.

En el footer se encuentran los datos de contacto y un enlace en la palabra “Contáctanos” que permitirá al usuario introducir su correo para así poder recibir emails con las novedades sobre la aplicación. Además, en este se encuentran también unos iconos que funcionan como enlace para llevar a los usuarios a nuestras cuentas de Facebook, Instagram y Twitter, solo con pinchar en ellos.

El usuario podrá crear, eliminar, modificar, listar y detallar, los empleados, procesos y equipos de la empresa. En estas pestañas se han tenido en cuenta todas las posibilidades, como son:

-Poder confirmar lo que uno está haciendo.
-Poder cancelar en caso de no estar conforme con lo que uno está haciendo.

Así se consigue una limpia navegación por la aplicación.

4.3.2.-DISEÑO DETALLADO

4.3.2.1.- DESCRIPCION DE CADA PROCEDIMIENTO

En la aplicación se han añadido algunas funcionalidades que hacen más fácil la navegación en ella, pero si estas funcionalidades no estuvieran la aplicación seguiría siendo igual de útil, solo cambiaria la navegación.

Hay un botón para ir hacia atrás en todas las ventanas salvo en la ventana principal, ya que es la primera. Este botón aparece en la cabecera de la página, a la derecha, al lado del botón de cerrar sesión (logout). En las ventanas de crear y en la de suscribirse este botón mandara a la pagina principal, sin embargo, en las páginas de detalles, editar y borrar llevan a las páginas de lista respectivamente. Por ejemplo, viendo los detalles de un empleado pulsando el botón de atrás lleva a la pagina en la que se ven todos los empleados como lista.

Tras filtrar por nombre de empleado, marca de equipo o nombre de proceso, los enlaces siguen estando disponibles. Esto se hace mediante JavaScript, de esta forma la funcionalidad de la pagina es la misma que si no se filtrara y facilita la usabilidad. De esta forma no hace falta ir al estado anterior para poder usar los enlaces.

4.3.2.2.- DEFINICION DE LA INFORMACION DE E/S

La información esta almacenada en base de datos, para introducirla hay dos opciones, para ambas hay que iniciar sesión como administrador. La primera opción es usar los formularios que se han dispuesto en la aplicación web, la segunda opción es usar la aplicación de administrador que dispone Django.

Esta información se puede editar y borrar, también hay que iniciar sesión como administrador, todo ello desde la aplicación web o la de Django. Con la información de base de datos, se crean las APIs con las que se obtiene la información para poder filtrar mediante lo que introduce el usuario en el cuadro de texto. Esta información se usa para mostrar al usuario como lista los empleados, equipos y procesos existentes. También se usa para mostrar los detalles de cada empleado, equipo y proceso.

En cuanto a los datos que hay en la tabla de novedades en base de datos, estos datos solo se introducen mediante un formulario. No se muestran de ningún modo ya que solo serán usados para mandar correos electrónicos a los usuarios que se registran.

4.3.3.- DISEÑO DE LA ESTRUCTURA FÍSICA DE LOS DATOS

```python
class Equipo(models.Model):
    modelo = models.CharField(max_length=100)
    marca = models.CharField(max_length=100)
    categoria = models.CharField(max_length=100)
    fecha_adquisicion = models.DateField(default="01/01/1990")
    fecha_instalacion = models.DateField(default="01/01/1990")

    def __str__(self):
        return f"ID: {self.id} | Modelo: {self.modelo} | Marca: {self.marca} | Categoria: {self.categoria} | F. Adquisicion: {self.fecha_adquisicion} | F. Insta
```

Ilustración 1 Estructura de los datos de los equipos

En la imagen anterior se muestra como es la estructura de los datos de los equipos, con sus atributos y sus valores preasignados.

```python
class Empleado(models.Model):
    """Para introducir DNIs extranjeros"""
    dni = models.CharField(primary_key=True, max_length=20)
    """Por nombres compuestos"""
    nombre = models.CharField(max_length=100)
    """Dos apellidos"""
    apellidos = models.CharField(max_length=150)
    email = models.EmailField(max_length=100)
    """Valor defecto ejemplo"""
    telefono = models.IntegerField(default=666666666)
    direccion = models.CharField(max_length=150)
    fecha_nacimiento = models.DateField(default="01/01/1990")

    def __str__(self):
        return f"DNI: {self.dni} | Nombre: {self.nombre} | Apellidos: {self.apellidos} | Email: {self.email} | Telefono: {self.telefono} | Direccion: {self.dire
```

Ilustración 2 Estructura de los datos de los empleados

En la imagen anterior se muestra como es la estructura correspondiente a los datos de los empleados, con sus atributos y sus valores preasignados.

```python
class Proceso(models.Model):
    """codigo_orden_fabricacion numeros y letras"""
    codigo_orden_fabricacion = models.CharField(max_length=75)
    """codigo_proceso Mezcla numeros y letras"""
    codigo_proceso = models.CharField(primary_key=True, max_length=75)
    nombre_proceso = models.CharField(max_length=100)
    """ref Mezcla numeros y letras"""
    referencia = models.CharField(max_length=75)
    fecha_inicio = models.DateField(default="01/01/1990")
    fecha_fin = models.DateField(default="01/01/1990")
    equipo = models.ForeignKey(Equipo, on_delete=models.CASCADE)
    empleados = models.ManyToManyField(Empleado)
    activo = models.BooleanField(default=False)

    def __str__(self):
        return f"C. Preceso: {self.codigo_proceso} | Nombre: {self.nombre_proceso} | Referencia: {self.referencia} | F. Inicio: {self.fecha_inicio} | F. Fin: {s
```

Ilustración 3 Estructura de los datos de los procesos

En la anterior imagen se muestra como es la estructura de los datos correspondiente a los procesos, con sus atributos y sus valores preasignados.

```python
class Novedades(models.Model):
    usuario = models.CharField(max_length=50)
    genero = models.CharField(max_length=9, choices=(('mas', 'Masculino'), ('fem', 'Femenino'), ('otro', 'Otro')), default='otro')
    email = models.EmailField(max_length=100)

    def __str__(self):
        return f"Usuario: {self.usuario} | Genero: {self.genero} | Email: {self.email}"
```

Ilustración 4 Estructura de los datos para el envío de emails

En la anterior imagen se muestra como es la estructura correspondiente a los datos para los envíos por mail (novedades), con sus atributos y sus valores preasignados.

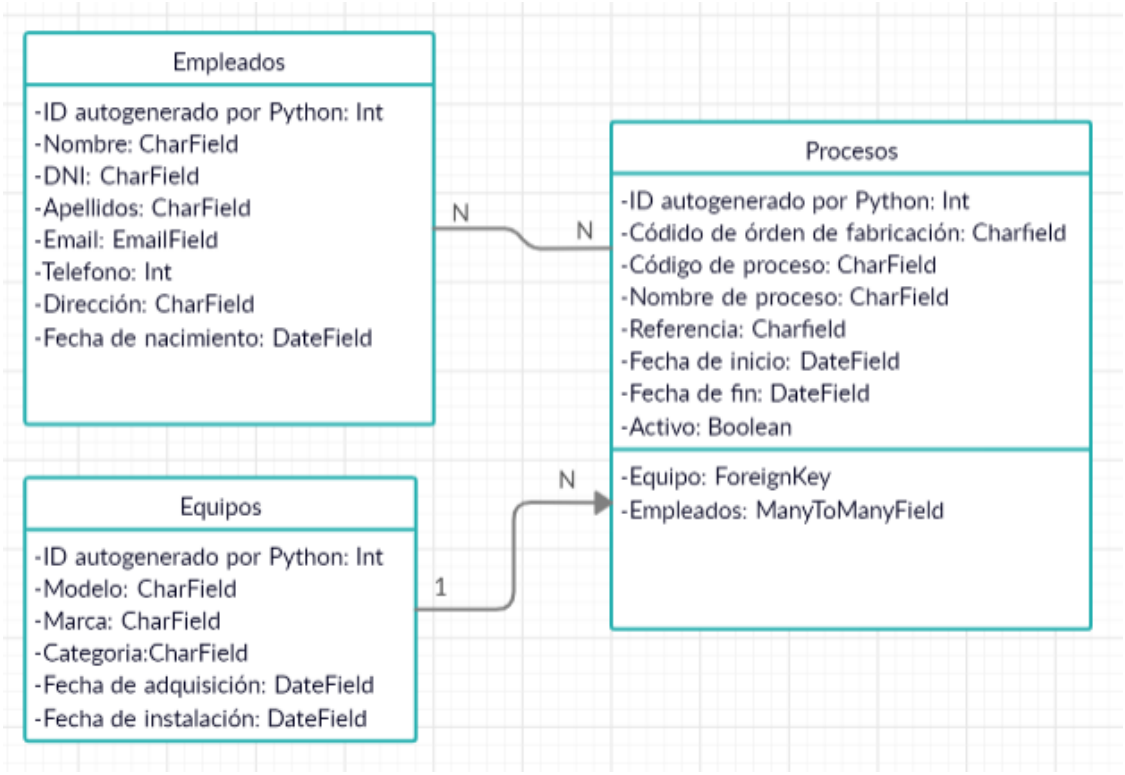

Ilustración 5 Diagrama UML sobre la estructura de los datos y de la BD

4.3.3.1.- DEFINICIÓN DE VISTAS

En cuanto a la creación, eliminación, listado, detallado y modificación; de los procesos, equipos y empleados, se han realizado las siguientes vistas.

```python
"""Vista para ver el listado de empleado"""
class EmpleadoListView(ListView):
    model = Empleado
    template_name = 'empleado_list.html'
    queryset = Empleado.objects.order_by('nombre')

    def get_context_data(self, **kwargs):
        context = super(EmpleadoListView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Empleados existentes'
        return context
```

Ilustración 6 Vista para el listado de los empleados existentes

```python
"""Vista para el formulario de creacion de empleado"""
class EmpleadoCreateView(StaffRequiredMixin, View):
    def get(self, request, *args, **kwargs):
        form = EmpleadoForm()
        context = {
            'form': form,
            'titulo_pagina': 'Apartado para la creación de empleados'
        }
        return render(request, 'create_empleado_form.html', context)

    def post(self, request, *args, **kwargs):
        form = EmpleadoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('empleado_list')

        return render(request, 'create_empleado_form.html', {'form': form})
```

Ilustración 7 Vista para añadir nuevos empleados

```python
"""Vista para ver el detalle de los empleados"""
class EmpleadoDetailView(DetailView):
    model = Empleado
    template_name = 'empleado_detail.html'

    def get_context_data(self, **kwargs):
        context = super(EmpleadoDetailView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Detalles del empleado'
        return context
```

Ilustración 8 Vista para mostrar los detalles del empleado seleccionado

```python
"""Vista para modificar los datos de los empleados"""
class EmpleadoUpdateView(StaffRequiredMixin, UpdateView):
    model = Empleado
    form_class = EmpleadoForm
    template_name = 'empleado_update.html'
    success_url = reverse_lazy('empleado_list')

    def get_context_data(self, **kwargs):
        context = super(EmpleadoUpdateView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Modificar empleados'
        return context
```

Ilustración 9 Vista para la modificación de los datos de los empleados

```python
"""Vista para eliminar empleados"""
class EmpleadoDeleteView(StaffRequiredMixin, DeleteView):
    model = Empleado
    template_name = 'empleado_delete.html'
    success_url = reverse_lazy('empleado_list')
```

Ilustración 10 Vista para la eliminación de los empleados de la BD

```python
"""Vista para el formulario de creacion de equipo"""
class EquipoCreateView(StaffRequiredMixin, View):
    def get(self, request, *args, **kwargs):
        form = EquipoForm()
        context = {
            'form': form,
            'titulo_pagina': 'Apartado para la creación de empleados'
        }
        return render(request, 'create_equipo_form.html', context)

    def post(self, request, *args, **kwargs):
        form = EquipoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('equipo_list')

        return render(request, 'create_equipo_form.html', {'form': form})
```

Ilustración 11 Vista para la creación de nuevos equipos

```python
"""Vista para ver el listado de equipos"""
class EquipoListView(ListView):
    model = Equipo
    template_name = 'equipo_list.html'
    queryset = Equipo.objects.order_by('id')

    def get_context_data(self, **kwargs):
        context = super(EquipoListView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Equipos existentes'
        return context
```

Ilustración 12 Vista para el listado de los equipos existentes

```python
"""Vista para ver el detalle de los equipos"""
class EquipoDetailView(DetailView):
    model = Equipo
    template_name = 'equipo_detail.html'

    def get_context_data(self, **kwargs):
        context = super(EquipoDetailView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Detalles del equipo'
        return context
```

Ilustración 13 Vista para mostrar los detalles del equipo seleccionado

```python
"""Vista para modificar los datos de los equipos"""
class EquipoUpdateView(StaffRequiredMixin, UpdateView):
    model = Equipo
    form_class = EquipoForm
    template_name = 'equipo_update.html'
    success_url = reverse_lazy('equipo_list')

    def get_context_data(self, **kwargs):
        context = super(EquipoUpdateView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Modificar equipos'
        return context
```

Ilustración 14 Vista para la modificación de los datos de los equipos

```python
"""Vista para eliminar equipos"""
class EquipoDeleteView(StaffRequiredMixin, DeleteView):
    model = Equipo
    template_name = 'equipo_delete.html'
    success_url = reverse_lazy('equipo_list')
```

Ilustración 15 Vista para la eliminación de los equipos de la BD

```python
"""Vista para el formulario de creación de procesos"""
class ProcesoCreateView(StaffRequiredMixin, View):
    def get(self, request, *args, **kwargs):
        form = ProcesoForm()
        context = {
            'form': form,
            'titulo_pagina': 'Apartado para la creación de procesos'
        }
        return render(request, 'create_proceso_form.html', context)

    def post(self, request, *args, **kwargs):
        form = ProcesoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('proceso_list')

        return render(request, 'create_proceso_form.html', {'form': form})
```

Ilustración 16 Vista para la creación de nuevos procesos

```python
"""Vista para ver el listado de procesos"""
class ProcesoListView(ListView):
    model = Proceso
    template_name = 'proceso_list.html'
    queryset = Proceso.objects.order_by('codigo_proceso')

    def get_context_data(self, **kwargs):
        context = super(ProcesoListView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Procesos existentes'
        return context
```

Ilustración 17 Vista para listar los procesos existentes

```python
"""Vista para ver el detalle de los procesos"""
class ProcesoDetailView(DetailView):
    model = Proceso
    template_name = 'proceso_detail.html'

    def get_context_data(self, **kwargs):
        context = super(ProcesoDetailView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Detalles del proceso'
        return context
```

Ilustración 18 Vista para mostrar los detalles del proceso seleccionado

```python
"""Vista para modificar los datos de los procesos"""
class ProcesoUpdateView(StaffRequiredMixin, UpdateView):
    model = Proceso
    form_class = ProcesoForm
    template_name = 'proceso_update.html'
    success_url = reverse_lazy('proceso_list')

    def get_context_data(self, **kwargs):
        context = super(ProcesoUpdateView, self).get_context_data(**kwargs)
        context['titulo_pagina'] = 'Modificar procesos'
        return context
```

Ilustración 19 Vista para la modificación de los datos de los procesos

```python
"""Vista para eliminar procesos"""
class ProcesoDeleteView(StaffRequiredMixin, DeleteView):
    model = Proceso
    template_name = 'proceso_delete.html'
    success_url = reverse_lazy('proceso_list')
```

Ilustración 20 Vista para la eliminación de los procesos de la BD

También se han implementado vistas JSON para el listado y detallado de los empleados, procesos y equipos existentes. Estas últimas se han implementado para la creación de la API. Se pueden ver a continuación.

```python
"""Vista para ver el listado de equipos en formato JSON"""
class EquipoJsonListView(View):
    def get(self, request):
        if('marca' in request.GET):
            eList = Equipo.objects.filter(marca__contains=request.GET['marca'])
        elif ('modelo' in request.GET):
            eList = Equipo.objects.filter(modelo__contains=request.GET['modelo'])
        else:
            eList = Equipo.objects.all()
        return JsonResponse(list(eList.values()), safe=False)


"""Vista para ver el detalle de los equipos en formato JSON"""
class EquipoJsonDetailView(View):
    def get(self, request, pk):
        equipo = Equipo.objects.get(pk=pk)
        return JsonResponse(model_to_dict(equipo))
```

Ilustración 21 Vistas para listar y detallar los equipos en formato JSON

```python
"""Vista para ver el listado de empleados en formato JSON"""
class EmpleadoJsonListView(View):
    def get(self, request):
        if('nombre' in request.GET):
            emList = Empleado.objects.filter(nombre__contins=request.GET['nombre'])
        else:
            emList = Empleado.objects.all()
        return JsonResponse(list(emList.values()), safe=False)


"""Vista para ver el detalle de los empleados en formato JSON"""
class EmpleadoJsonDetailView(View):
    def get(self, request, pk):
        empleado = Empleado.objects.get(pk=pk)
        return JsonResponse(model_to_dict(empleado))
```

Ilustración 22 Vistas para listar y detallar los empleados en formato JSON

```python
"""Vista pAra ver el listado de procesos en formato JSON"""
class ProcesoJsonListView(View):
    def get(self, request):
        if('codigo_proceso' in request.GET):
            pList = Proceso.objects.filter(codigo_proceso__contins=request.GET['codigo_proceso'])
        else:
            pList = Proceso.objects.all()
        return JsonResponse(list(pList.values()), safe=False)


"""Vista para ver el detalle de los procesos en formato JSON"""
class ProcesoJsonDetailView(View):
    def get(self, request, pk):
        proceso = Proceso.objects.get(pk=pk)
        return JsonResponse(model_to_dict(proceso))
```

Ilustración 23 Vistas para listar y detallar los procesos en formato JSON

Además, a continuación, se muestra la vista correspondiente al Logout de la aplicación.

```python
"""Vista para que el usuario pueda cerrar sesión"""
class LogoutView(RedirectView):
    permanent = False
    query_string = True
    pattern_name = 'login'

    def get_redirect_url(self, *args, **kwargs):
        if self.request.user.is_authenticated():
            logout(self.request)
        return super(LogoutView, self).get_redirect_url(*args, **kwargs)
```

Ilustración 24 Vista para cerrar sesión

Todas las vistas visualizadas en este apartado, se pueden ver más a detalle en la propia aplicación.

CAPÍTULO 5: MANUAL DE USUARIO

En este capítulo se intentará explicar cómo funciona la aplicación y que hacer para navegar por ella.

Primeramente, hay que hacer, o bien iniciar sesión como administrador o entrar a la aplicación como invitado.

En el primer caso, como ya se ha comentado anteriormente se tiene plena libertad en cuanto a creación, eliminación, modificación, listado y detalles de todos los "elementos" pertenecientes al proceso productivo, los cuales son; los empleados, los equipos y los procesos.

En el segundo caso, no se va a tener ese "libre albedrio" en cuanto a la navegación, simplemente se podrán visualizar los datos, ya que se trata de un invitado.

Sea como sea, todos los usuarios van a poder utilizar las funcionalidades básicas de la aplicación web. Estas son:

-Poder ir a la página principal clicando en el logo de la aplicación, que está situado en la parte izquierda del header.

-Poder ir hacia la pestaña anterior clicando en el botón de atrás.

-Poder clicar en el botón de LogOut para enviar al usuario a la pestaña de inicio de sesión.

-Poder contactar con nosotros clicando en los diferentes iconos de diferentes redes sociales; Facebook, Twitter e Instagram, situados en la parte derecha del footer.

-Poder recibir correos clicando en "Contáctanos" que está situado en la parte izquierda del footer, encima de los iconos de las redes sociales.

CAPÍTULO 6: INCIDENCIAS DEL PROYECTO Y CONCLUSIONES

Como primera incidencia, surgió un problema relacionado con la instalación de Python en uno de los ordenadores portátiles. Sin embargo, a lo largo del proyecto se solvento y se ha conseguido seguir adelante con la programación de la aplicación web.

Como segunda incidencia, se invirtió más tiempo de lo esperado en solucionar un problema relacionado con los estilos CSS, pero al final se podría decir que, en cuanto a estilos, esta aplicación web, no se queda corta.

Y hablando en general, se han realizado muchas fases de prueba para código, HTML, estilos, etc., que se podría haber invertido en otras cosas.

Como conclusión del proyecto, se podría decir que se ha aprendido muchos de los errores cometidos. Errores relacionador a la mala organización en determinados momentos y a la mala planificación. Sin embargo, pese a todo esto se ha sacado un gran provecho de este proyecto.

https://www.youtube.com/channel/UCLQUXOwRimwewRObZx1v2VA

https://www.youtube.com/channel/UCLXRGxAzeaLDGaOphqapzmg

https://alud.deusto.es/course/view.php?id=13310

https://stackoverflow.com/

Jon Vadillo, profesor de la asignatura de Ingeniería Web en la Universidad de Deusto.