

## **Ingeniería Web – Proyecto Web Colaborativo**

**Título:** Gestión de pedidos - DEUSTRONIC

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

**Materia:** Ingeniería Web

---

**Estudiantes:**

Ibai Carreño Manso

Arkaitz Brea Samaniego

Juan Carlos Colón García

Asier Aguirre Alonso de Mezquia

**Grupo:** IW-04

**Profesor:** Jon Vadillo Romero

**Facultad de Ingeniería**

**Universidad de Deusto**

**Vitoria - Gasteiz, mayo de 2020**



# Introducción

---

## Resumen del proyecto

El proyecto de gestión de pedidos se trata de una aplicación web, la cual está orientada a la gestión y manipulación de datos que controla una empresa llamada Deustronic. Sus distintas características hacen que se parezca a un ERP de gestión, permitiendo y siendo de gran ayuda para los trabajadores ya que se realiza un seguimiento de los distintos pedidos, facturas, clientes, productos y sus componentes.

La aplicación ha sido desarrollada en cuatro distintos lenguajes (HTML para el modelaje de la aplicación web, CSS para aplicar estilos a este modelaje, PYTHON para la funcionalidad de la propia aplicación y gestión de la lógica de datos, y JAVASCRIPT para complementar distintas funcionalidades a python del lado del cliente); se ha desarrollado bajo un framework web llamado DJANGO que permite el desarrollo de aplicaciones y páginas web seguras y mantenibles, y con el motor programable de Visual Studio Code para la programación y generación de código.

## Descriptores del proyecto

Informática

Aplicacion Web

ERP

Django



# Índice

<b>Introducción</b>	<b>iii</b>
Resumen del proyecto	iii
Descriptores del proyecto	iii
<b>Objetivos del proyecto</b>	<b>1</b>
Resumen para la dirección	1
Tareas principales	2
Planificación temporal	3
<b>Especificación de requisitos del sistema</b>	<b>4</b>
Catálogo de requisitos	4
Descripción de requisitos del nuevo sistema	6
Modelo funcional	6
Modelo lógico de datos	7
Descripción de la interfaz del sistema	10
Descripción general	10
Botones de la aplicación	11
Modelo de comportamiento	12
Interacción entre interfaces	12
<b>Especificación del diseño</b>	<b>14</b>
Introducción	14
Principales funciones del software	14
Descripción del entorno de desarrollo	15
Arquitectura física y entorno tecnológico	15
Descripción general	15
Descripción del diseño	16
ENTREGA:	16
<b>Manual de usuario</b>	<b>21</b>
Funciones de la aplicación	21
Vistas de la aplicación	21
<b>Incidencias del proyecto y conclusiones</b>	<b>25</b>
<b>Bibliografía</b>	<b>26</b>

# Objetivos del proyecto

---

## Resumen para la dirección

La aplicación que se va a desarrollar se trata de un sistema de información, en este caso, un ERP para la empresa Deustronic Components S.L.

Un sistema de información es un conjunto de datos que interactúan entre sí con un fin común. En informática, los sistemas de información ayudan a administrar, recolectar, recuperar, procesar, almacenar y distribuir información relevante para los procesos fundamentales y las particularidades de cada organización, en este caso, de Deustronic Components S.L.

Se ha decidido realizar un ERP debido a que la empresa lo que desea es solucionar sus problemas derivados del uso de hojas Excel y comenzar a utilizar una aplicación para la gestión de pedidos, productos, clientes, etc.

Es por ello, por lo que la solución empleada no ha sido realizar una página Web donde los clientes pudieran realizar compras o visualizar los productos que se desarrollan en la empresa, sino que se ha creído conveniente realizar una aplicación de “escritorio” (como puede ser cualquier sistema de información), pero enfocando la solución a una aplicación Web.

Una ventaja y un salto de la Industria 4.0 es poder tener la información sobre la empresa en la nube y no en local como se viene trabajando hasta la fecha, donde la información únicamente se puede acceder desde el lugar en el que se encuentra instalado el software.

De esta manera, la empresa, puede acceder a los datos desde cualquier parte mediante una conexión a Internet. Otra de las ventajas a destacar serían las actualizaciones, que, al tratarse de una página Web, no hace falta actualizar el programa cada vez que la empresa desarrolladora realiza sobre el software.

Por otro lado, realizar un sistema de información en la nube tiene un gran inconveniente, que es la accesibilidad, ya que cualquier persona con acceso a Internet podría tener “acceso” a la página.

Es por ello por lo que, el ERP, al estar alojado en un servidor externo puede ser accesible por los atacantes, los cuales, en caso de no tener una correcta seguridad, podrían acceder a información sensible de la empresa.

Después de la toma de requisitos mediante reuniones con la empresa, se llegó a la conclusión de que lo que necesitaba y buscaba para la correcta gestión de esta, era una aplicación que pudiera gestionar, de manera dinámica, sencilla e intuitiva, la siguiente información:

- Gestión de productos: creación, visualización (listado y detalle) y baja.
- Gestión de pedidos: creación, visualización (listado y detalle) y actualización.
- Gestión de clientes
- Gestión de componentes que componen cada producto

Al tratarse de un ERP, donde lo que se gestiona es información que se guarda en bases de datos, lo más importante es generar una base de datos normalizada para evitar redundancias. A su vez, uno de los puntos más importantes para poder gestionar dicha información son las relaciones entre las tablas, de tal manera que, si un cliente realiza un pedido de un producto, se pueda mostrar correctamente la información más importante, como, por ejemplo, quien ha realizado la compra, cual es el número de pedido, qué productos componen dicho pedido, cual es el precio del producto y del pedido, etc.

La aplicación se va a desarrollar en Django. Django es un framework de desarrollo web de código abierto, escrito en Python. A su vez, la información se guardará en una base de datos relacional, que, en posteriores apartados se hará hincapié y se describe brevemente sobre el tipo de base de datos por emplear en el proyecto.

En definitiva, la aplicación que se va a desarrollar va a tratar de ser sencilla de utilizar para el empleado de la empresa, dinámica y visual (empleando tablas con los datos a mostrar).

## Tareas principales

Para la realización de la siguiente aplicación se han identificado las siguientes tareas a realizar:

- Creación del modelo de datos
- Creación de la pantalla principal de la aplicación
- Creación de las vistas de crear, editar y eliminar de los diferentes objetos
- Creación de las vistas para la visualización de la lista y detalles de objetos de cada modelo
- Creación de los formularios
- Creación de los html-s para crear, editar, eliminar, listar, ver detalles y formularios
- Creación del html base de la aplicación
- Implementar css a cada uno de los html-s
- Envío de correo electrónico a un nuevo cliente
- Descargar y visualizar los archivos almacenados en cada producto
- Cambio de tamaño de letra y fondo de la aplicación
- Validación de los formularios
- Ordenar ascendente y descendente los productos según el atributo seleccionado
- Obtención de datos y rellenar una tabla mediante un JSON

## Planificación temporal

	Semana 1							Semana 2							Semana 3							Semana 4							Semana 5								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Creación del modelo de datos	x																																				
Creación de la pantalla principal de la aplicación													x																								
Creación de las vistas de crear, editar y eliminar de los diferentes objetos												x																									
Creación de las vistas para la visualización de la lista y detalles de objetos de cada modelo												x																									
Creación de los formularios													x																								
Creación de los html-s para crear, editar, eliminar ,listar, ver detalles y formularios																																					
Creación del html base de la aplicación													x																								
Implementar css a cada uno de los html-s																																					
Envío de correo electrónica a un nuevo cliente																																					
Descargar y visualizar los archivos almacenados en cada producto																																					
Cambio de tamaño de letra y fondo de la aplicación																																					
Validación de los formularios																																					
Ordenar ascendente y descendente los productos según el atributo seleccionado																																					
Obtención de datos y rellenar una tabla mediante un JSON																																					

Como se puede observar en la imagen, este es el proceso que se ha seguido para hacer el proyecto.

**SEMANA 1** → En la primera semana, primero se pensó cómo iba a ser la aplicación y su modelo de datos. Además se creó la aplicación. También, se empezó a picar código. Y se hizo: los modelos de datos (en models.py) las primeras vistas, y los formularios.

**SEMANA 2** → En la segunda semana y acercándose la fecha de entrega. Se hicieron los Html-s principales del proyecto e implementación de los css correspondientes a cada html.

**SEMANA 3** → Después de hacer la entrega, estábamos esperando a dar más teoría para hacer las nuevas aportaciones al proyecto.

**SEMANA 4 y 5** → En un corto periodo de tiempo se consiguió implementar las nuevas funcionalidades hechas mediante javascript. Y se implementaron dos nuevas funcionalidades de python.



# Especificación de requisitos del sistema

---

## Catálogo de requisitos

Las principales funciones por cumplir son las que la empresa ha demandado para la correcta gestión de la información especificada.

En cada ventana se mostrarán acciones diferentes, es decir, la ventana de PRODUCTO será independiente a la de CLIENTE, pero en cada una de ellas se realizarán unas funciones en común las cuales son:

- CREAR
- VISUALIZAR
- ACTUALIZAR
- BORRAR

Para realizar la aplicación de manera correcta y desarrollar las funciones necesarias, se realizará un orden en dichas funciones para optimizar el tiempo de trabajo y poder realizar un resultado más profesional.

Debido a que estas funciones son las básicas para poder realizar acciones sobre la información que se encuentra en las tablas de la base de datos, en primer lugar, se ha de realizar el modelo de datos.

A continuación, una vez realizados los correctos formularios en la parte del servidor se han de hacer aquellas funciones que implementan los mismos campos para la manipulación de la información del modelo de datos (visualizar, crear, actualizar y/o borrar dicha información) de manera visual. Estas funciones, se tratan del correcto empleo de la estructura y el diseño de la aplicación Web, como son HTML5 y CSS3, respectivamente.

A su vez, en la gestión de la información se incluirán dos funciones extra para poder crear un cierto tipo de acciones que requieren de un lenguaje de programación.

Estas funciones se dividen en dos tipos de funciones: las empleadas en Django (Python) y las empleadas en JavaScript.

Aquellas que se desarrollan en Django son las siguientes:

- **Envíos de correos electrónicos:** esta función permite enviar un determinado correo electrónico. La implementación de esta funcionalidad en el proyecto desarrollado es la de enviar un email al cliente cuando en el ERP se registra un nuevo cliente.

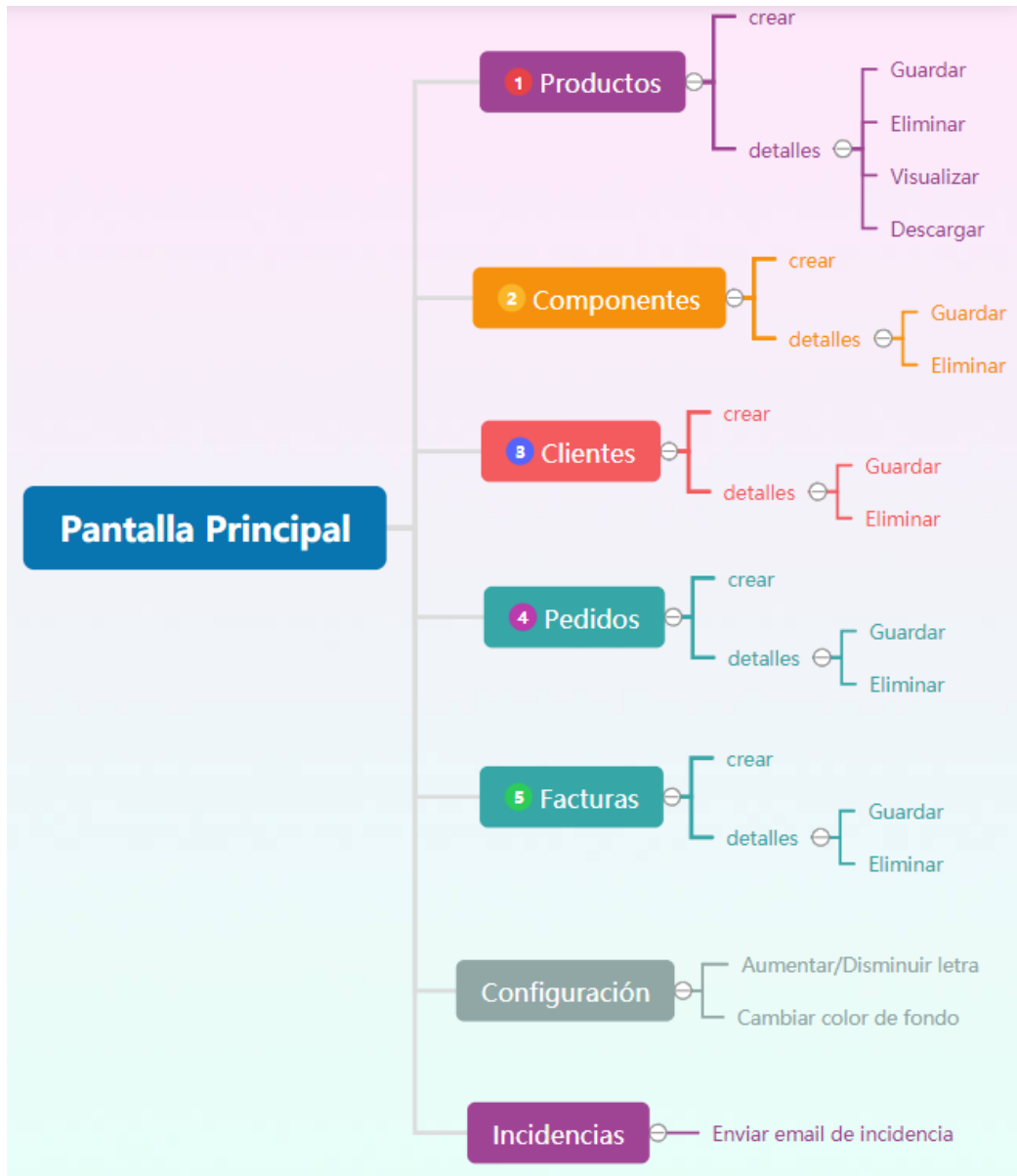
- **Guardar/Descargar/Visualizar archivos:** la función llevada a la práctica es la de poder Guardar, descargar o visualizar archivos de los productos que se encuentran en la base de datos. Por tanto, cada producto puede tener un archivo asociado al mismo, como, por ejemplo, el datasheet de dicho producto.

Aquellas que se desarrollan en JavaScript son las siguientes:

- **Reloj:** la función del reloj es una función básica y sencilla gracias al Objeto “Date” y sus métodos para obtener el día, mes, año, horas, minutos, etc.
- **Métodos de ordenación:** se ha realizado la funcionalidad de ordenar una tabla al visualizarla, por ejemplo, en el listado de productos se puede poder ordenar de manera ascendente/descendente por referencia, precio, nombre o categoría.
- **Validación de formularios:** gracias a la funcionalidad introducida mediante JavaScript, la validación resulta más sencilla. No obstante, la validación mediante JavaScript no ejerce como seguridad, sino que sólo advierte al usuario si ha introducido correctamente el tipo de dato o longitud en un campo. Es por ello, por lo que los formularios se han de validar siempre tanto en el FrontEnd como en el BackEnd.
- **Cambiar letra/fondo:** en caso de que el usuario desee cambiar el tamaño de la letra o el color de fondo, puede hacerlo gracias a la funcionalidad implementada en la página de ajustes. Por otro lado, debido a que al recargar la página los ajustes volverían a su estado inicial, se han guardado dichos datos en variables en el local storage, de tal forma que, aunque la página se recargue, el color y el tamaño de letra va a continuar siendo el determinado por el usuario.
- **Generar código de un JSON retornado de la BBDD:** esta funcionalidad está implementada en el listado de productos, el cual, mediante JavaScript, se genera dinámicamente tantas filas como productos existen en la base de datos. Gracias a Django también se puede generar código dinámicamente.

## Descripción de requisitos del nuevo sistema

### Modelo funcional



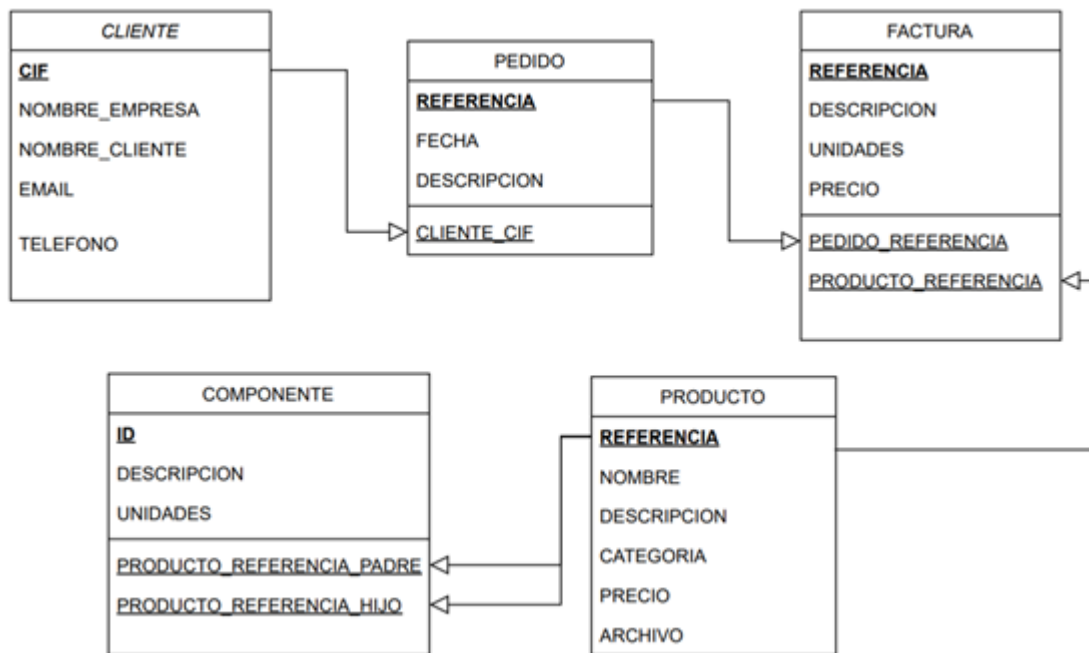
La aplicación empezará desde una pantalla principal en la que aparecerán 7 botones diferentes con sus respectivos nombres; Producto, Componente, Cliente, Pedido, Factura, Configuración e Incidencias. Al pinchar en uno de ellos, se cambiará la página y aparecerá su correspondiente lista, menos en el caso de Configuración e Incidencias que aparecerán sus pantallas específicas.

Desde la lista de objetos, habrá la opción de crear nuevos o acceder a los detalles de los ya creados, también se podrá volver a la pantalla anterior.

La pantalla de detalle es la que más funcionalidades tiene, en caso de estar compuesto por otros objetos también se verá una lista de estos, existirá la opción de eliminar el objeto y si se hacen cambios sobre los campos y se pulsa en guardar estos se editarán.

## Modelo lógico de datos

### ENTIDAD - RELACIÓN



Un CLIENTE puede realizar N PEDIDO's, pero un PEDIDO sólo puede estar asociado a un CLIENTE. A su vez, un PEDIDO puede contener N FACTURA's. En las FACTURA's se guarda también el PRODUCTO, por lo que un PRODUCTO puede estar en diferentes(N) FACTURAS.

Para comprender de manera más sencilla la explicación de las relaciones del párrafo anterior, a continuación, se pondrá un ejemplo de un pedido realizado por un cliente a una empresa real.

Imaginándose que la empresa es AMAZON y el CLIENTE es una persona que desea realizar la compra de, por ejemplo, un móvil 5G, un portátil HACER y unos cascos inalámbricos, al seleccionar los artículos y proceder al método de pago la persona únicamente realiza un PEDIDO (con los tres artículos seleccionados), pero cada artículo está asociado a una FACTURA diferente, donde se encuentra la relación del PEDIDO y del PRODUCTO.

Por otro lado, un PRODUCTO puede tener muchos componentes, por ejemplo, si la empresa DEUSTRONIC desea realizar un ordenador, el ordenador tendrá los COMPONENTE's de placa base, procesador, tarjeta gráfica, memoria, etc. Y a su vez, estos COMPONENTE's tendrán otros COMPONENTE's, como puede ser el caso de la placa base, que tendría los COMPONENTE's de diodos, resistencias, etc.

En definitiva, existe una jerarquía donde el PRODUCTO padre sería el ORDENADOR y éste tendría diferentes COMPONENTE's, pero que a su vez no dejan de ser un PRODUCTO.

Es por ello por lo que COMPOENTE tiene dos claves foráneas de la tabla producto, ya que un COMPONENTE está compuesto por dos PRODUCTO's, ordenador y procesador, padre e hijo, respectivamente.

Aquellos campos los cuales se encuentran subrayados y en negrita en la figura XX se refieren a las claves primarias de las tablas; por el contrario, aquellos campos los cuales únicamente se encuentran subrayados hacen referencia a las claves foráneas.

## RELACIONAL

CLIENTE(**CIF**, NOMBRE\_EMPRESA, NOMBRE\_CLIENTE\_EMAIL)

PRODUCTO(**REFERENCIA**, NOMBRE\_DESCRIPCION\_CATEGORIA, PRECIO, ARCHIVO)

COMPONENTE(**ID**, DESCRIPCION, UNIDADES, PRODUCTO\_REFERENCIA\_PADRE,  
PRODUCTO\_REFERENCIA\_HIJO)

PEDIDO(**REFERENCIA**, FECHA, DESCRIPCIÓN, CLIENTE\_CIF)

FACTURA(**REFERENCIA**, UNIDADES, PRECIO, PEDIDO\_REFERENCIA,  
PRODUCTO\_REFERENCIA)

En el modelo relacional se pueden ver los campos por los cuales está formada cada tabla. En este modelo, siguiendo la misma lógica que en el modelo Entidad-Relación, los campos subrayados y en negrita son las claves primarias; mientras que, los que únicamente están subrayados son las claves foráneas.

## ESPECIFICACIÓN DE ENTIDADES

```
# Tabla producto
class Producto(models.Model):
    producto_referencia = models.CharField(max_length=13, primary_key=True)
    producto_nombre = models.CharField(max_length=50)
    producto_descripcion = models.CharField(max_length=250)
    producto_categoria = models.CharField(max_length=50)
    producto_precio = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    #Atributo fichero
    producto_pdf = models.FileField(blank=True, null=True)

    # Funcion que devuelve el producto_nombre cuando se visualiza en el /admin
    def __str__(self):
        return self.producto_nombre
```

Imagen del código de la clase Producto del models.py

La creación de las tablas de la base de datos se realiza en el archivo de `models.py` de Django. Un modelo en Django es un tipo especial de objeto que se guarda en la base de datos.

Una base de datos es una colección de datos, es decir, es el lugar en el cual se almacenará la información sobre las tablas de la figura XX.

En este proyecto se utilizará una base de datos SQLite para almacenar los correspondientes datos. SQLite es un sistema de gestión de bases de datos relacional.

Como se puede observar en la figura ZZ la clase `producto` contiene los campos que la componen. De esta manera se pueden crear, actualizar, leer o borrar objetos de la clase `producto`, los cuales se encontrarán en la base de datos.

En cada campo se indica el tipo de dato del que se trata, es decir, el campo `producto_referencia` de la clase `producto` es de tipo `CharField` debido a que se refiere a un campo alfanumérico.

Por otro lado, algunas propiedades importantes a tener en cuenta podrían ser:

- **Primary\_key:** indica que el campo se trata de la clave primaria de la tabla.
- **Max\_length:** indica el tamaño máximo del campo.
- **Max\_digits:** al igual que `Max_length`, pero para campos decimales.
- **Default:** establece un valor por defecto al campo.
- **Unique:** restricción que se utiliza para garantizar que no se inserten valores duplicados en una columna.
- **On\_delete:** atributo que se encuentra en los campos de claves foráneas y que indica la acción por realizar en caso de que se realice un borrado de la tabla a la que hace referencia.
  - **Cascade:** borra en cascada, por ejemplo, si se borra un `PRODUCTO`, se borrarán los `COMPONENTE's` de dicho `PRODUCTO`.
  - Etc.

A su vez, en caso de que un campo se tratase de la clave foránea de otra tabla, no hace falta introducir el tipo de dato del que se trata debido a que apunta al tipo de dato de la clave primaria de la tabla a la cual se está haciendo la relación.

```
# Tabla factura
class Factura(models.Model):
    referencia = models.CharField(max_length=12, primary_key=True)
    pedido_referencia = models.ForeignKey(Orden_Pedido, on_delete=models.CASCADE)
    producto_referencia = models.ForeignKey(Producto, on_delete=models.CASCADE)
```

Imagen del código de la clase `Factura` del `models.py`

La tabla FACTURA tiene de claves foráneas pedido\_referencia (PEDIDO) y producto\_referencia (PRODUCTO), por lo que se indica la tabla con la cual mantienen la relación y el tipo de borrado por realizar.

Por otro lado, la función `__str__` que se encuentra dentro de una clase en el modelo de datos sirve para mostrar objetos de la clase que devuelve una cadena de caracteres con lo que se desea mostrar. Este método se invoca cada vez que se llama a la función `str`.

A su vez, `__init__` es, al igual que `__str__`, un método especial y que equivale, en cierta medida, a lo que se conoce como constructor en otros lenguajes. Su principal función es establecer un estado inicial en el objeto nada más instanciarlo, es decir, inicializar los atributos.

## Descripción de la interfaz del sistema

### Descripción general





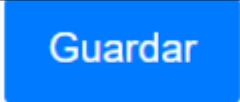
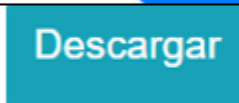
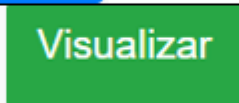
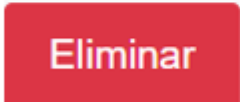

La aplicación está compuesta por una pantalla principal, pantallas de visualización de listas, detalles, detalles y lista; pantallas de creación y eliminación; una de configuración y otra de incidencias. Desde las pantallas de lista se podrá acceder a la de creación y a la de detalle y desde las de detalle se podrá editar y eliminar. En el caso de los productos desde la pantalla de detalles también se podrá visualizar y descargar el archivo correspondiente al producto. En el siguiente esquema se ven las funcionalidades principales de la aplicación:

La aplicación dispondrá de diferentes interfaces, pero todas dispondrán de un header común y la mayoría de estas también contarán con un footer que tendrá unos botones u otros dependiendo de la pantalla en la que nos encontremos. En la aplicación existen diferentes tipos de pantallas:

- Pantalla de inicio
  - 5 botones azules
    - Productos
    - Componentes
    - Clientes
    - Pedidos
    - Facturas
  - 2 botones grises
    - Configuración
    - Incidencias
- Pantalla de listado → Aparece la lista de objetos.
  - Botón crear
  - Botón volver
  - Doble clic sobre objeto → Detalles del objeto
- Pantalla de detalle → Detalles del objeto (más lista si está compuesto por otros)
  - Botón de guardar
  - Botón eliminar
  - Botón visualizar y descargar en producto
  - Botón volver
- Pantalla de crear
  - Botón guardar
  - Botón volver
- Pantalla de eliminar
  - Botón eliminar
  - Botón cancelar

## Botones de la aplicación

Los botones de las pantallas tienen esta funcionalidad:

Imagen del botón	Funcionalidad
	Ir a la pantalla principal
    	Entrar a la pantalla de lista
 	Acceder a la pantalla de configuración e incidencias
	Acceder a pantalla de creación
	Guardar nueva creación o edición de un objeto
 	Descargar o visualizar el archivo de un producto
	Eliminar un objeto
	Botón para volver a la pantalla anterior



## Modelo de comportamiento

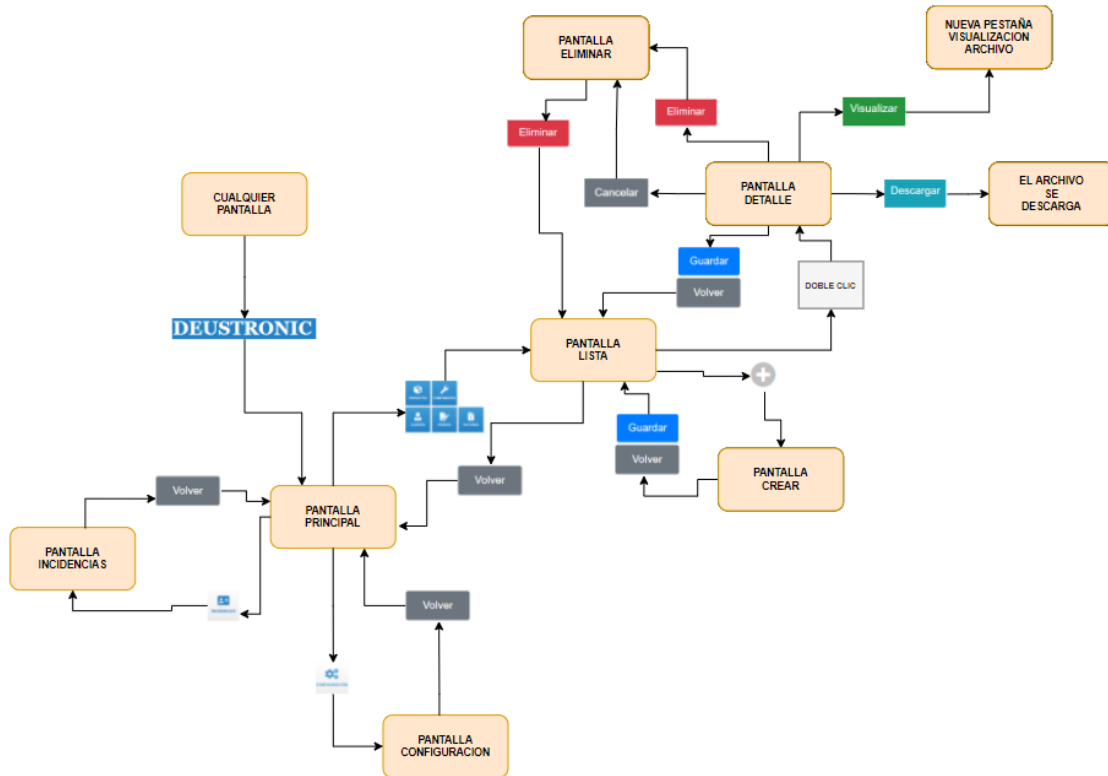


Imagen del modelo de comportamiento de la aplicación

## Interacción entre interfaces



1 Pantalla principal

La aplicación empezará mostrando la pantalla principal. En esta parecerán 5 botones azules y 2 botones grises. Los botones azules corresponden a Productos, Componentes, Pedidos, Clientes y Facturas y al pulsar encima de ellos te llevan a la lista de objetos del botón pulsado. Los grises, Configuración e Incidencia; el primero sirve para cambiar el color de fondo y el tamaño de la letra, mientras que el segundo sirve para enviar un email al correo indicado avisando de una incidencia.



2 Pantalla Lista

En la vista de la lista de un objeto, se pueden hacer dos cosas. Por un lado, pulsando el botón en la parte derecha de Productos se accederá a la pantalla de creación de un objeto de ese tipo. Por el otro lado, si se hace doble clic sobre uno de los objetos de la lista se accede a los detalles de ese objeto. Dentro de esta pantalla de detalle se podrá editar y eliminar el objeto. A la hora de editar un objeto basta con cambiar el atributo que se quiera y pulsar en guardar. Si se quiere eliminarlo nos saldrá un aviso que se deberá aceptar o no. Si dispone de un archivo se podrá descargar y visualizar mediante sus correspondientes botones.



3 Pantalla Detalle



4 Pantalla Crear

Todas las pantallas disponen de un botón volver que te lleva a la pantalla anterior y si se pulsa sobre el logo de DEUSTRONIC se vuelve a la pantalla de inicio.

Para ver mejor la transición entre las diferentes interfaces, se explicara el Diagrama de interfaces . Arriba del todo se puede ver la pantalla principal, después de estas las pantallas de listar objetos, la pantalla de configuración y la pantalla de incidencias. Debajo de las pantallas de listado, están las pantallas de crear o ver detalles del objeto seleccionado anteriormente. Y por último, después de la pantalla de detalle se encuentra la pantalla de

eliminación de un objeto o de visualización de un archivo en caso de que el objeto contenga uno. Como se ha dicho anteriormente, existe un botón para volver a la pantalla anterior y si se pulsa sobre

el logo DEUSTRONIC se vuelve a la pantalla principal. (Ver diagrama en la imagen anexada Diagrama de interfaces.)



5 Diagrama de interfaces

# Especificación del diseño

---

## Introducción

### Principales funciones del software

Nuestra aplicación tiene como objetivo ser una ERP. Para ello, le hemos añadido varias funcionalidades, con las que podemos llegar a controlar toda una empresa, dichas funcionalidades son:

- Controlar la cantidad de **productos** y **componentes** que tiene la empresa, para ello almacenan una información de cada producto y se muestra en unas listas (código, nombre, descripción...).
- Añadir nuevos **clientes**, cada cliente tiene una información la cual se almacena en listas al igual que productos y componentes.
- Muestra en una lista los **pedidos** que se han hecho, además todos los pedidos tienen una **factura**, en la cual se añade el precio de el pedido que han hecho. Y ese precio, aparece también en unas listas, en las que a cada precio le corresponde un código de factura, un nombre, un código de producto...
- Todas los objetos pueden ser **creados**, **editados** y **borrados** desde la propia app.
- Para facilitar la navegación del usuario dentro de la app, se ha añadido una serie de **configuraciones** para poder editar la página, pudiendo cambiar el tamaño de la letra y cambiar el color de fondo. Además, en el encabezado de la página se encuentra un **reloj**, que indica la hora y fecha.

- El precio de la factura está por defecto en euros, pero puede visualizarse en dólares pulsando un botón con el símbolo \$.
- Posibilidad de enviar un **correo** a cualquier dirección de correo, consiguiendo de esta forma alertar al remitente que se ha producido una incidencia, o que el cliente se ha dado de alta en el sistema

## Descripción del entorno de desarrollo

El entorno de desarrollo que se ha utilizado ha sido Django. Este entorno, es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Las principales razones por las que utilizar Django son:

- Se adapta a cualquier tipo de aplicación web
- Permite la construcción de aplicaciones de forma segura
- Gratuito y de código abierto
- Hay mucha información en internet, y nos puede ayudar mucho.
- Incluye al instalarlo un conjunto de componentes para gestionar la mayoría de las tareas de desarrollo web. Y de esta forma nos facilita el trabajo.

Al instalar este entorno, genera una serie de carpetas, en las que hay código ya escrito. Esto facilita al programador para generar la base de datos. Aún así, hay partes de ese código que hay que cambiar y añadir para poder hacer bien la base de datos.

Este entorno está dividido en 5 archivos distintos → asig.py, wsig.py, settings.py, \_\_init\_\_.py y urls.py. (en descripción del diseño explicaremos cual es la funcionalidad de cada archivo dentro del programa)

## Arquitectura física y entorno tecnológico

### Descripción general

En primer lugar se crea el esquema de la base de datos. Una vez haber tenido la idea y el esquema se instala el framework Django. Después, se crea el proyecto y la aplicación, la cual se introduce en Visual Studio Code. En ella hay dos partes separadas y creadas automáticamente, Proyecto y Aplicación.

En cuanto a la Aplicación, se crean los modelos en models.py y se referencian en el admin.py. Después, se generan todas las vistas en views.py. También, se introducen las direcciones en urls.py. Finalmente, se crean los formularios en forms.py.

En la carpeta de templates, se crean todas las plantillas de HTML.. El más importante es base.html, ya que se extenderá en las demás plantillas. Hay una plantilla para cada vista de la aplicación.

A la par de la generación de códigos HTML, se crean los archivos css. Finalmente, se generan archivos de javascript. Cada uno de los archivos de javascript tiene una funcionalidad dentro de la aplicación distinta.

## Descripción del diseño

El programa se divide en dos partes:

### ENTREGA:

-Esta parte es en la que está la base de datos, en este caso se ha utilizado django.

Dentro de entrega hay 4 archivos python distinto, cada uno cumple una función distinta:

- **asig.py y wsgi.py** → Son dos servidores de Django, los cuales permiten que la aplicación tenga una comunicación en tiempo real. La diferencia de estos dos servidores es que el primero de ellos se encarga en controlar las comunicaciones asíncronas y el segundo las comunicaciones síncronas. El código introducido dentro de estos dos archivos es casi idéntico, en él simplemente importamos el servidor dentro de nuestra app.
- **settings.py** → Toda la configuración que usa Django en nuestro proyecto está dentro de este archivo. El código que está introducido dentro del archivo lo genera django por defecto. Dentro de ese código hay varias variables que tienen distintas funciones dentro del programa. Por ejemplo, la variable `Installed_apps` es donde hay que agregar las aplicaciones de terceros que hemos utilizado en el programa o la variable `databases` donde podemos configurar cualquier motor de base de datos que Django utilice(MySQL, Postgres, Oracle, etc)
- **urls.py** → La url que necesita Django para hacer la aplicación, está dentro de este archivo. El código que está introducido dentro del archivo lo genera django por defecto. En la captura de la imagen se puede observar aparece un código con el cual se consigue redirigir todo lo que entre a 'http://127.0.0.1:8000/' hacia `appEntrega.urls` y buscará más instrucciones allí.

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('appEntrega/', include('appEntrega.urls')),  
]
```

Imagen del código del `urls.py` del proyecto

### APPENTREGA:

- Esta parte es la programación de toda la aplicación. Está dividida en una serie de carpetas, y en cada una de ellas se utiliza una programación distinta (Python, Javascript, HTML y CSS) estos 4 softwares forman la app.

### PYTHON:

Los archivos de python están en la propia carpeta de `appEntrega`, y hay 6 archivos distintos:

- **models.py** → La función de este archivo es introducir todas las tablas que tiene nuestra aplicación. Para hacerlo de forma correcta, primero tienes que diseñar el esquema de tu base de datos. Y una vez habérsela hecho, introducir todas las tablas de tu base de datos en este archivo. Toda tabla necesita tener distintos campos, en los cuales hay que añadir que tipo de dato es, y una serie de argumentos. Por ejemplo:

Dentro de la tabla clientes uno de sus campos es cif. En el código aparece:

- **models.CharField** → esto hace referencia a que tipo de dato es, en este caso es una cadena de longitud en la que hay que especificar el tamaño máximo que puede tener.
- **primary\_key** → clave primaria de la entidad.

```
# Tabla cliente
class Cliente(models.Model):
    cif = models.CharField(max_length=12, primary_key=True)
```

Imagen del código del parámetro CIF de Cliente

- **views.py** → La función de este archivo es tomar una solicitud web y devolver una respuesta web. Esta respuesta puede ser el contenido HTML de una página web, una redirección, un error 404... Para rellenar views.py hemos introducido la información que tiene nuestro modelo. Más tarde, esta información pasará a la plantilla.

Se han creado todas las vistas necesarias para que nuestro código funcione. Cada una de las vistas se conectan con un archivo HTML. Estas vistas, se pueden dividir en 6 clases distintas: Listas de campos, Añadir campo, Eliminar campo, Editar campo, Envío de emails y settings\_view)

```
# Vista de FacturaListView
class FacturaListView(ListView):
    model = Factura
    template_name = 'listaFacturas.html'
    context_object_name = 'lista_facturas'

# Borrar Factura
class DeleteFacturaView(DeleteView):
    model = Factura
    template_name = 'borrarFactura.html'
    success_url = reverse_lazy('listaFactura')

# Editar Factura
class UpdateFacturaView(UpdateView):
    model = Factura
    context_object_name = 'factura'
    template_name = 'updateFactura.html'
    fields = ['pedido_referencia', 'producto_referencia', 'unidades', 'descripcion']
    success_url = reverse_lazy('listaFactura')
```

Imagen del código de las vistas de Factura

Estas 3 vistas hacen la función de mostrar lista, borrar y editar las facturas. La estructura del código es muy similar. En cada una de ellas hay que poner a qué campo corresponde, a qué archivo html corresponde. Y al ser vistas basadas en clases hay que añadir la línea de `success_url`. Para los demás campos el código de mostrar lista, borrar y editar un campo son prácticamente iguales, pero cambiando los parámetros.

```
# Vista de formulario de crear una nueva factura
class CreateFacturaView(View):
    def get(self, request, *args, **kwargs):
        form = FacturaForm()
        context = {
            'form': form,
            'titulo_pagina': 'Nueva factura'
        }
        return render(request, 'nuevoFactura.html', context)

    def post(self, request, *args, **kwargs):
        form = FacturaForm(request.POST)
        if form.is_valid():
            form.save()

            # Volvemos a la pagina que queramos despues de crear un nuevo producto
            return redirect("listaFactura")

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

Imagen del código de las vistas de creación

Este es el código que utilizamos para añadir una nueva factura. Lo que más se puede destacar, es que tenemos que hacer dos funciones distintas, `get` y `post`. La primera de ellas para crear la nueva

factura y la segunda para guardarla y volver a `listaFactura`. Para los demás campos el código de añadir un campo es prácticamente igual, pero cambiando los parámetros.

Para finalizar, está el código encargado de enviar a un email el texto:

```
'¡Hola! Al parecer, su empresa ha provocado una incidencia
en nuestro sistema. Pronto, Deustronic S.A. se pondrá en contacto con
usted mediante el n° de teléfono que indicó al realizarse el alta.
Disculpe las molestias.',
```

y el `settingsView`, el cual permite el cambio de estilo de la página, es decir; permite cambiar el tamaño de las letras, y cambiar el color del fondo.

**urls.py** → Este archivo es el encargado de hacer todos los enlaces que va a tener nuestra aplicación. Cada archivo HTML creado corresponde a una URL distinta, la cual se añade en este archivo.

```
# URL lista pedidos

path('pedidos/', PedidoListView.as_view(), name='listaPedido'),
```

Todas las url siguen esta estructura:

- **Path:** Dirección url
- **Pedidos:** Campo correspondiente
- **PedidoListView.as\_view():** La clase de la vista del views.py
- **Name:** Nombre dinámico de la dirección

**foms.py** → Este archivo es el encargado de generar todos los formularios. En nuestra aplicación, tenemos 5 formularios distintos, cada uno de ellos para generar un nuevo dato de uno de los campos. Para ello creamos una clase, en la que añadimos forms.ModelForm, debido a que es un formulario y en la que se añade el campo que hace referencia. Además añadimos fields = “\_\_all\_\_” ya que especificamos todos los campos del modelo de datos lógico indicado.

```
# Formulario nuevo producto

class ProductoForm(forms.ModelForm):

    class Meta:

        model = Producto

        fields = '__all__'
```

**apps.py** → Su función es ayudar al usuario a incluir cualquier configuración de aplicación para la aplicación.

**admin.py** → Su función es registrar todos los modelos de nuestra aplicación. Este código está importando los modelos y después llama a `admin.site.register` para registrar a cada uno de ellos.

```
# Aquí se registran los modelos/tablas
admin.site.register(Cliente)
admin.site.register(Producto)
admin.site.register(Componente)
admin.site.register(Orden_Pedido)
admin.site.register(Factura)
```

Imagen de los modelos registrados de la aplicación de administrador



TEMPLATES:

- En esta carpeta están todos los archivos html. En ellos se hace la estructura de las páginas de la aplicación. El archivo base.html, es el que se utiliza en todos los htmls como plantilla. En él, se ha hecho el header y el body. Y el footer, dependiendo de que archivo html fuera era distinto. Aunque al ser un ERP la mayoría de los archivos html no tienen footer.

STYLE:

En esta carpeta están todos los archivos css. CSS son las hojas de archivo que utilizamos para dar forma y estilo a los archivos html. Para ello en los html hacemos extends de los css correspondientes. En esta aplicación tenemos 11 archivos css distintos.

- **bootstrap.css** → el cual es un kit de herramientas de código.
- **correo.css** → Este es el css encargado de editar la página correo.html
- **forms.css** → Este es el encargado de editar los formularios.
- **header.css** → Este es el css encargado de editar el header, el cual está en todos los htmls
- **index.css** → Este es el encargado de editar la página principal (127.0.0.1:8000/appEntrega/)
- **modal.css** → Este es el encargado de crear una ventana modal al pulsar en los botones indicados.
- **productos.css** → Este es el encargado de editar el icono de añadir y el de ordenar la lista de productos
- **settings.css** → Este es el encargado de editar la página de configuración
- **styles.css** → Este es el encargado de editar base.html
- **table.css** → Este es el encargado de editar las tablas de cada lista

JS:

En esta carpeta se almacenan los scripts de javascript. Cada archivo tiene una funcionalidad distinta dentro del programa.

- **Conversormoneda** → Se encarga de pasar el precio de las facturas de euros a dólares.
- **Formularios.js** → Se encarga de validar los campos introducidos correctamente de los formularios.
- **reloj** → Se encarga de colocar un reloj y un calendario en el header.
- **settings.js** → Se encarga de subir y bajar el tamaño de la letra, y de cambiar el color del fondo de la pantalla.
- **Bundle.js** → Agrupa todos los ficheros JS en uno solo. Así el navegador no necesita hacer varias peticiones HTTP.
- **generarCodigo** → Se encarga de ordenar las listas en función del parámetro seleccionado.

IMG:

En esta carpeta se almacenan todas las fotos que se han utilizado para hacer la aplicación.

# Manual de usuario

---

La aplicación de gestión de proyectos trae distintas funcionalidades que ayudan al usuario a gestionar los productos, pedidos, facturas, clientes, y componentes de los propios productos (que a su vez, los componentes son catalogados como productos). Se trata de un ERP que la empresa DEUSTRONIC podrá implantar para que sus empleados cumplan los objetivos marcados en la empresa, mejorando tiempos y disminuyendo la opción de dar cabida a errores al introducir los datos.

## Funciones de la aplicación

La aplicación consta con varias funciones, tal y como se ha comentado previamente. Da lugar a crear y gestionar distintos “objetos”, así como clientes, productos y sus componentes, pedidos, y facturas de los pedidos.

Con todos esos tipos de objetos, las mayores funcionalidades son la creación, edición / actualización de su información, eliminación, y mostrar consultas acerca de los distintos objetos.

Para ello, la aplicación consta de una base de datos, donde se almacenan los datos introducidos por los empleados.

Dependiendo de la función que se vaya a llevar a cabo, y del tipo de objeto que se vaya a gestionar, la aplicación al ser web, da la opción de mostrar distintas vistas (ej.: mostrar clientes, actualizar producto, etc.).

## Vistas de la aplicación

Todas las vistas de la aplicación tendrán una cabecera común, donde se encontrarán el logo de la empresa Deustronic S.A., un reloj con la fecha y hora actual (UTC+1), y unos iconos donde, si el usuario lo desea, se podrían añadir funcionalidades como abrir un desplegable de las preguntas de uso más frecuentes (FAQ), una bandeja de entrada de correo electrónico, un apartado de alarma y recordatorio de eventos, y la opción de registrar qué trabajador está operando en ese momento.

Al iniciar la aplicación, esta mostrará la página principal llamada “index”. En ella se mostrarán distintos botones donde el usuario seleccionará el tipo de objeto con el que se querrá trabajar. Además de los tipos de objetos, también habrá dos botones adicionales donde se podrá configurar el estilo visual de la página, y una opción de reportar una incidencia a un cliente por mail.



Imagen de la página principal de la aplicación

En caso de seleccionar cualquier opción de tipo de objeto, se mostrará una vista de listado de todos los objetos que se hayan dado de alta, sean clientes o sean componentes de productos. También se dará la opción de eliminar un objeto de la base de datos pulsando en el icono de eliminar del final de cada fila. Véase en el ejemplo:

DEUSTRONIC					
29 / 05 / 2021 - 11:43:05					
Clientes +					
NOMBRE DE LA EMPRESA	CIF	EMAIL	Nº TELEFONO	RESPONSABLE dpto. COMPRAS	
Talgo	1	talgo@gmail.com	945001903	Ángel	
Asier S.A.	2	aguirre.asier@opendeusto.es	945172827	Asier	
					<a href="#">Volver</a>

Imagen de la vista de listado de clientes

En el caso de este ejemplo de la imagen, suponiendo que se acaba de dar de alta la empresa “Asier S.A.”, se enviará un mail al correo electrónico indicando que el alta en el sistema se ha realizado correctamente.

DEUSTRONIC S.A.



iempresadeusto@gmail.com <iempresadeusto@gmail.com>

11:26

Para: aguirre.asier@opendeusto.es

Hola! Gracias por dar de alta su empresa en Deustronic S.A. y por confiar en nosotros.  
Será un placer trabajar con usted!

Imagen del correo electrónico al dar de alta un cliente

En el caso de que el tipo de objeto seleccionado sea Producto, da la opción a descargar un archivo adicional (PDF, jpg, etc.), e incluso ordenar el listado de productos por orden de código de referencia, alfabético, precio, descripción, y categoría. Véase en la imagen siguiente:

29 / 05 / 2021 - 11:43:38

**Productos**

Ordenar: por: Precio

CÓDIGO	PRODUCTO	DESCRIPCIÓN	CATEGORIA	PRECIO	ARCHIVO
3	Transistor	Transistor electrónico	Electrónica	0.12	
2	Condensador	Condensadores de 4uF	Electrónica	0.67	
1	Arduino Uno	Placa programable	Electrónica	70.85	

nota: doble click para acceder a detalle

Volver

Imagen de la vista de listado de productos

En el caso de querer editar / actualizar los datos de algún objeto, estando en la vista del listado, no habrá más que hacer click sobre la fila del propio objeto, para que se cargue una vista donde se da la opción de editar los datos del objeto. Para ello, habrá que pulsar sobre la casilla de texto, e introducir el valor deseado.

En el pie de página, se encuentran una serie de botones donde se da la opción de guardar los datos introducidos (excepto la clave primaria), eliminar el objeto de la base de datos, y volver a la página anterior. En el caso de que el tipo de objeto sea Producto, habrá dos botones añadidos para visualizar el documento adjunto al producto, y para descargar dicho documento. Además, aparecerán a la derecha de los datos del producto los componentes por los que está compuesto el propio producto. Véase en el ejemplo:

29 / 05 / 2021 - 11:44:20

**Arduino Uno**

Producto nombre:

Producto descripción:

Producto categoría:

Producto precio:

Producto pdf:  

Currently: arduino-uno\_gFokhio.jpg

Clear

Change:

**Componentes de Arduino Uno**

CÓDIGO	COMPONENTE	UNIDADES	DESCRIPCION	PRECIO	
2	Condensador	8	Condensadores de la placa Arduino Uno	0.67	
3	Transistor	400	Transistores de la placa Arduino Uno	0.12	

nota: doble click en componente para acceder a su producto

Volver

Imagen de la vista de actualización de de datos de objetos

En el caso de seleccionar en la página principal la opción de configuración, se mostrará una sencilla vista donde dará la opción de configurar el color de fondo de la página, y el tamaño de la fuente empleada del texto. El tamaño de la fuente se podrá aumentar y reducir, y el color del fondo de la aplicación se podrá poner de color blanco, o con un color naranja más cálido para evitar sobrecarga en los ojos.

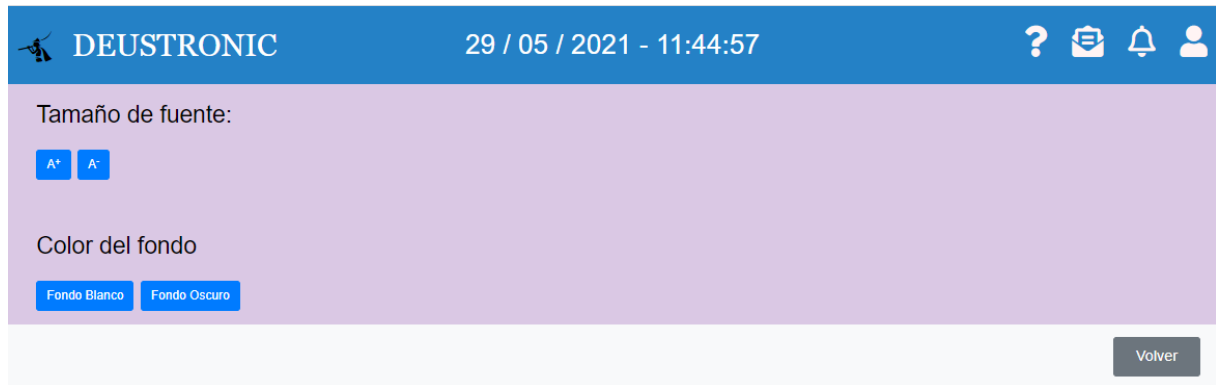


Imagen de la vista de configuración de apariencia

Si se selecciona la opción de Incidencia, se abrirá una nueva vista donde se introducirá el correo electrónico del cliente, y se le notificará que se ha producido una incidencia en el sistema, y que pronto se contactará con él.

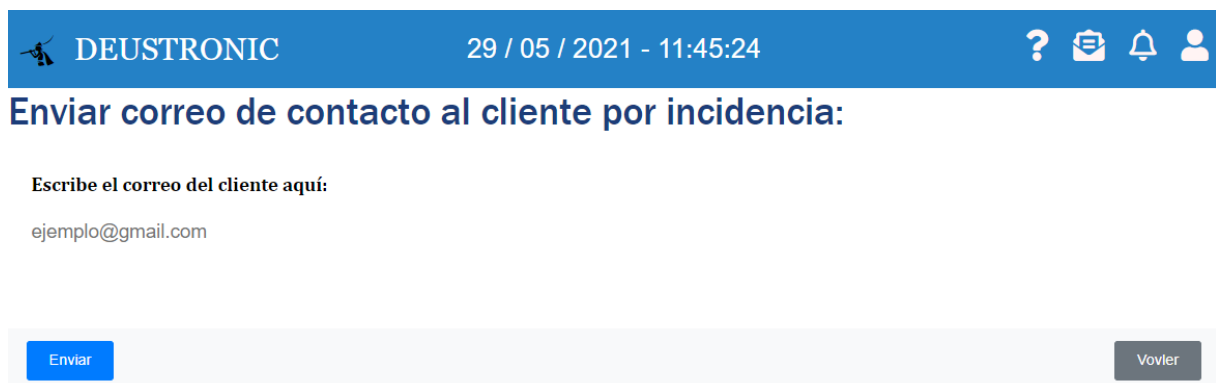


Imagen de la vista de Incidencia

Al pulsar el boton de enviar, se enviará el siguiente mail a la dirección introducida:

DEUSTRONIC S.A.



impresadeusto@gmail.com <impresadeusto@gmail.com>   
12:14

Para: aguirre.asier@opendeusto.es

Hola! Al parecer, su empresa ha provocado una incidencia en nuestro sistema. Pronto, Deustronic S.A. se pondrá en contacto con usted mediante el nº de teléfono que indicó al realizarse el alta. Disculpe las molestias.

Imagen del mail enviado al cliente por incidencia

# Incidencias del proyecto y conclusiones

---

Durante el transcurso y fases del proyecto, se han tenido una serie de incidencias, y una serie de éxitos.

Una incidencia que surgió fue la organización entre compañeros del grupo, ya que lo mismo alguien realiza algún cambio en el código, y después no se avisaba, no se comentaba el código nuevo, se dejaba incompleto, etc. En base a eso, la conclusión sacada ha sido que lo primordial en un trabajo en grupo, como este Proyecto Web Colaborativo, es una organización de todas las partes del grupo.

Otro problema que apareció fue que en ocasiones, no siempre se conocía a la perfección lo que se tenía que hacer, o incluso que alguien desvaría de su función y acaba haciendo la parte de otro integrante o una funcionalidad añadida que no se había mencionado previamente. Al hilo de lo comentado en el párrafo anterior, una buena organización del grupo es la base para el éxito del proyecto.

Algunos aspectos a mejorar sobre el proyecto serían los siguientes; implementación de métodos de ordenación en todas las tablas, multiplicación del precio del producto y la cantidad comprada para generar el precio de la factura. También creemos que se podría dividir la aplicación según departamentos, es decir, que en ventas se puedan solo gestionar clientes, pedidos y facturas. Por otro lado, el departamento de almacén se encargará de los productos y sus componentes. Otra bonita implementación sería hacer funcionalidades para los cuatro botones a la derecha del header.

Como conclusiones, el grupo ha aprendido bastante en el aspecto de la Ingeniería del Software, de cómo organizar un proyecto colaborativo, qué aspectos y tareas son las más costosas, etc.

La sensación final ha sido positiva en ámbitos generales.

## Bibliografía

- <https://docs.djangoproject.com/en/3.2/>
- <https://www.djangoproject.com/>
- <http://blog.enriqueoriol.com/2014/07/upload-de-imagenes-con-django.html>
- <https://docs.djangoproject.com/en/3.2/ref/class-based-views/generic-editing/>