

**Ingeniería Web – Proyecto Web Colaborativo**

**Titulo:** APLICACIÓN WEB PARA LA GESTIÓN DE PEDIDOS

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

**Materia:** Ingeniería Web

**Estudiantes:** Maialen Aguilar Ayuso

Ignacio Chapero Garnica

Álvaro Husillos Echazarreta

**Grupo:** IW-07

**Profesor:** Jon Vadillo Romero

**Facultad de Ingeniería**

UNIVERSIDAD DE DEUSTO

**Vitoria - Gasteiz, mayo de 2020**

# INDICE DE LA MEMORIA

Capítulo 1: INTRODUCCION

Capítulo 2: OBJETIVOS DEL PROYECTO

2.1 TAREAS PRINCIPALES

2.2 PLANIFICACIÓN TEMPORAL

Capítulo 3: ESPECIFICACION DE REQUISITOS DEL SISTEMA

3.1 INTRODUCCION

3.1.1 ALCANCE DEL PROYECTO

3.1.2 CATALOGO DE REQUISITOS

3.2 DESCRIPCION DE REQUISITOS DEL NUEVO SISTEMA

3.2.1 MODELO FUNCIONAL

3.3 DESCRIPCION DE LA INTERFAZ DEL SISTEMA

3.3.1 PERFIL DE LOS USUARIOS

**Capítulo 4: ESPECIFICACION DEL DISEÑO**

4.1 INTRODUCCION

4.1.1 PRINCIPALES FUNCIONES DEL SOFTWARE

4.1.2 DESCRIPCION DEL ENTORNO DE DESARROLLO

4.2 ARQUITECTURA FISICA Y ENTORNO TECNOLOGICO

4.2.1 DESCRIPCION GENERAL

4.3 DESCRIPCION DEL DISEÑO (Para Programación por Eventos)

4.3.1 DISEÑO DE LA ESTRUCTURA FISICA DE LOS DATOS

4.3.1.1DEFINICION DE VISTAS

**Capítulo 7: MANUAL DE USUARIO**

**Capítulo 8: INCIDENCIAS DEL PROYECTO y CONCLUSIONES**

**BIBLIOGRAFIA**

Capítulo 1: INTRODUCCION

Capítulo 2: OBJETIVOS DEL PROYECTO

2.1 TAREAS PRINCIPALES

En cuanto a planificación:

* Visualizar los aspectos a llevar a cabo
* Organizar el equipo
* Priorizar los elementos más importantes
* Planificar los pasos a seguir
* Marcar hitos
* Instalar softwares necesarios
* Ir informando de nuestros logros
* Reuniones una vez acabado el objetivo principal para ponernos al día

En cuanto a programación:

* HTML de la página web
* Dar estilo mediante CSS
* Elección del proyecto a realizar
* BrainStorm
* Definir modelos a desarrollar
* Tipo de vistas y acciones que necesitaremos
* Desarrollo de ideas

Capítulo 3: ESPECIFICACION DE REQUISITOS DEL SISTEMA

3.1 INTRODUCCION

3.1.1 ALCANCE DEL PROYECTO (qué se va a incluir dentro de los límites del sistema y qué no)

* Funcionalidades en Python, implementando nuevas funcionalidades.
* Funcionalidades JavaScript para cargar y almacenando los datos utilizando Fetch:
* Cargar datos y modificar DOM mediante JavaScript:

Llamada a API utilizando Fetch para obtener datos y mostrar los valores modificando el DOM.

* Envío de datos de un formulario mediante AJAX para su almacenamiento en BBDD.

3.1.2 CATALOGO DE REQUISITOS(Funciones a cumplir, inconvenientes actuales, nuevas necesidades)

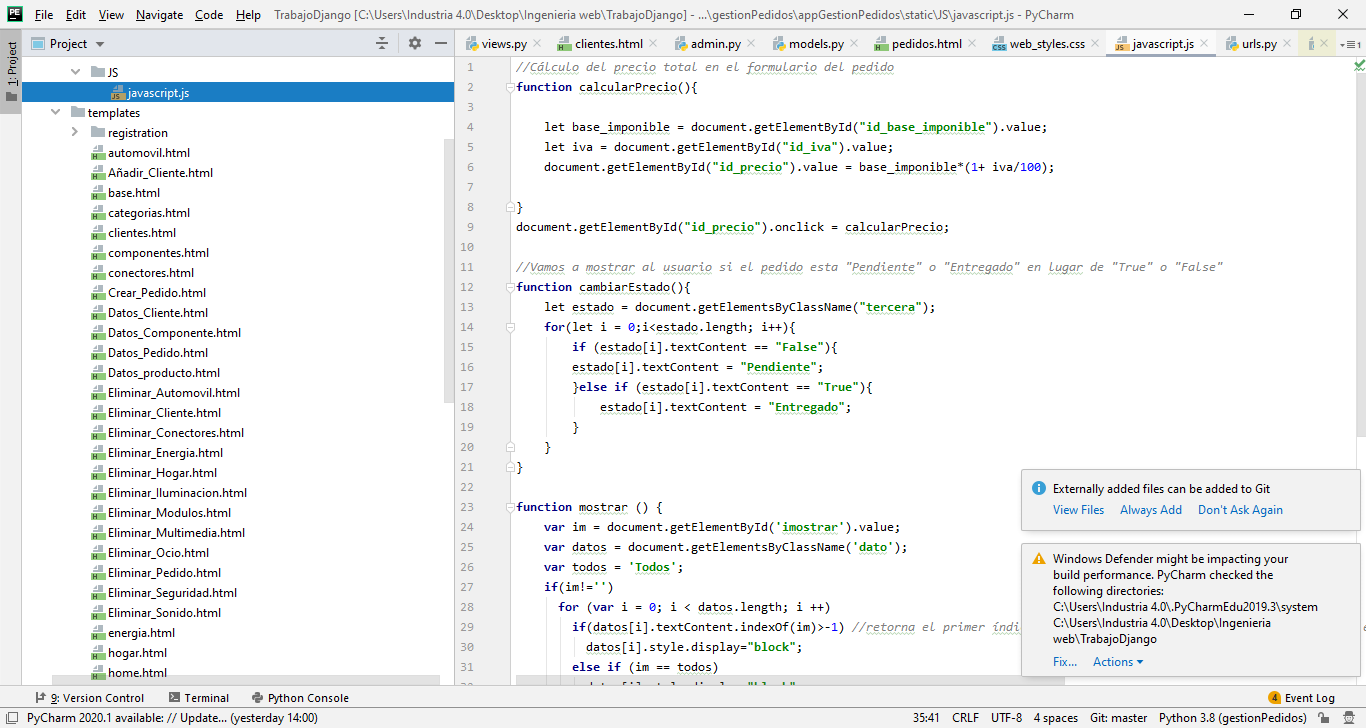
**Descripción de las funcionalidades implementadas (Funciones a cumplir) (JavaScript y Django):**

Django:

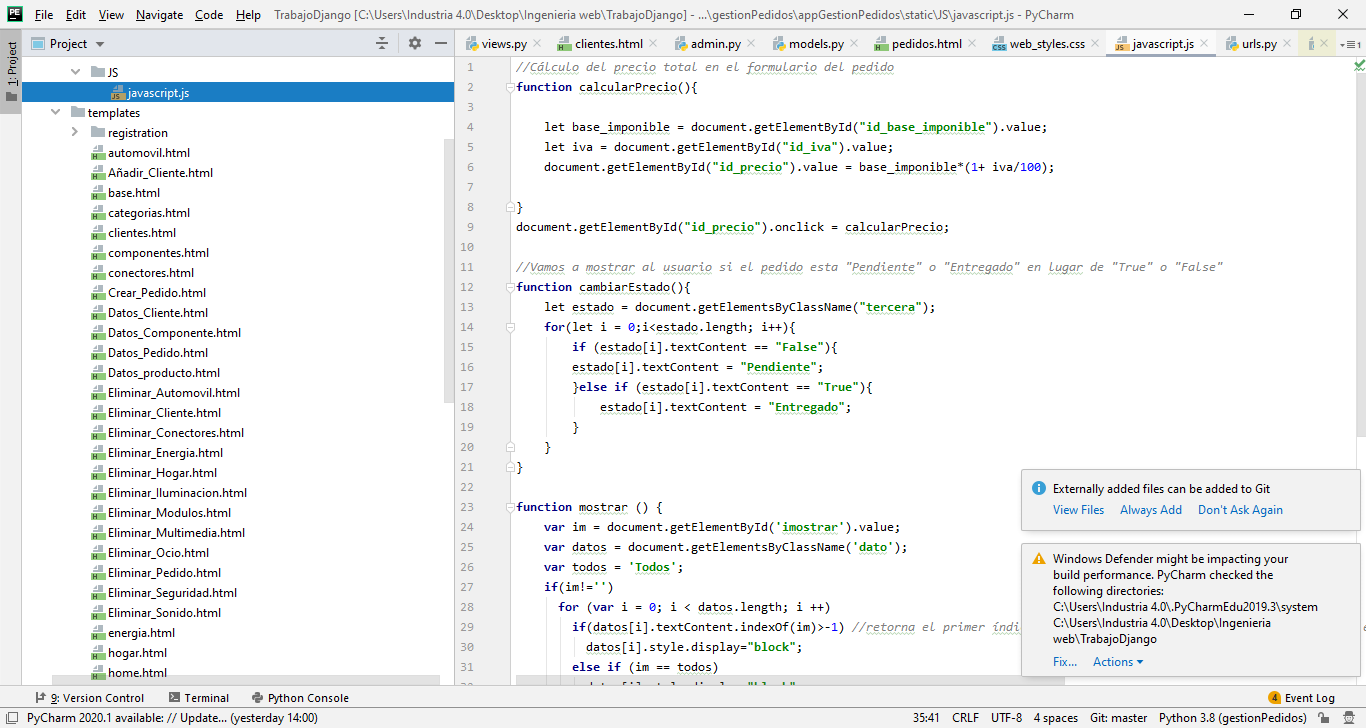
* Links en el menú a sus respectivas páginas
* Formularios para crear/añadir Pedidos, clientes y productos (implementar botones)
* Mostrar listas de datos (ListView)
* Mostrar los datos de un apartado en concreto (DetailView)
* Eliminar Pedidos, clientes o productos (DeleteView)
* Realizar las templates necesarias, para mostrar datos generales de cada categoría, pedidos, productos, componentes, los datos específicos de cada uno menos los componentes y los templates de eliminar.

JavaScript:

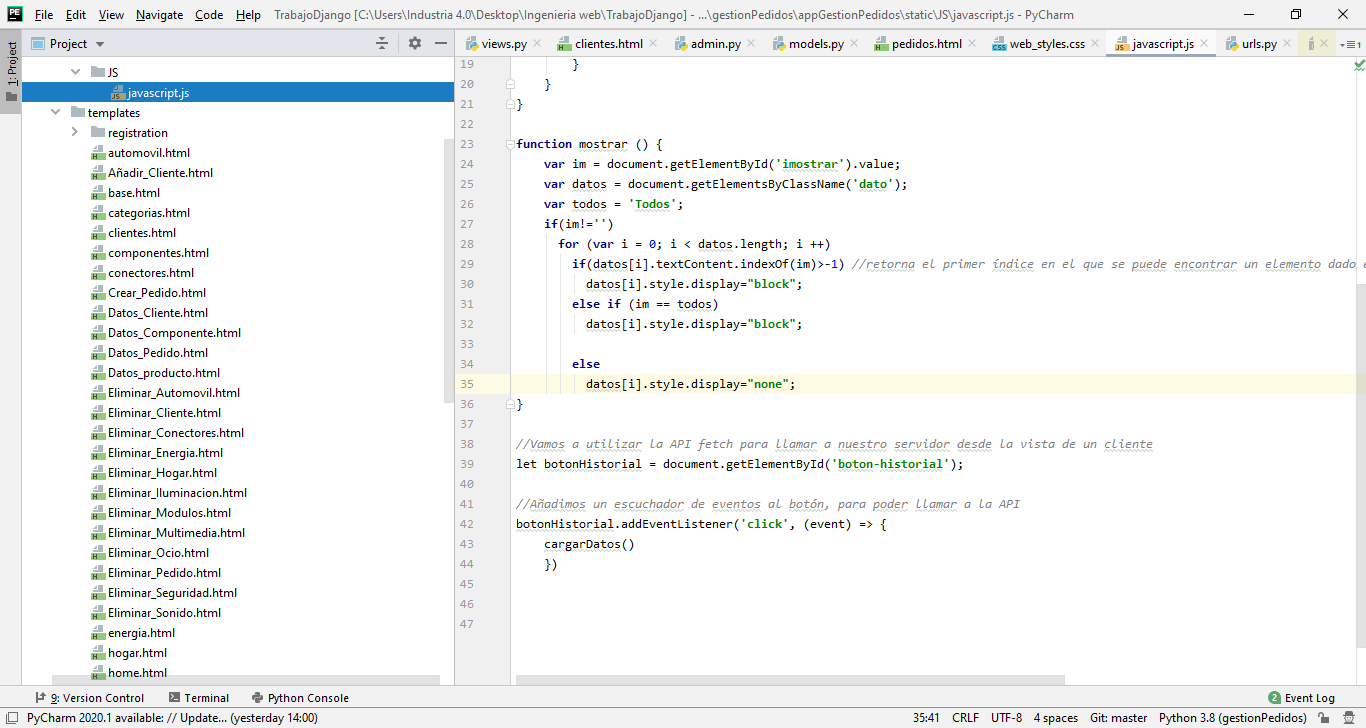
* Función para calcular el precio total del pedido según la base imponible, el iva y el precio señalado.



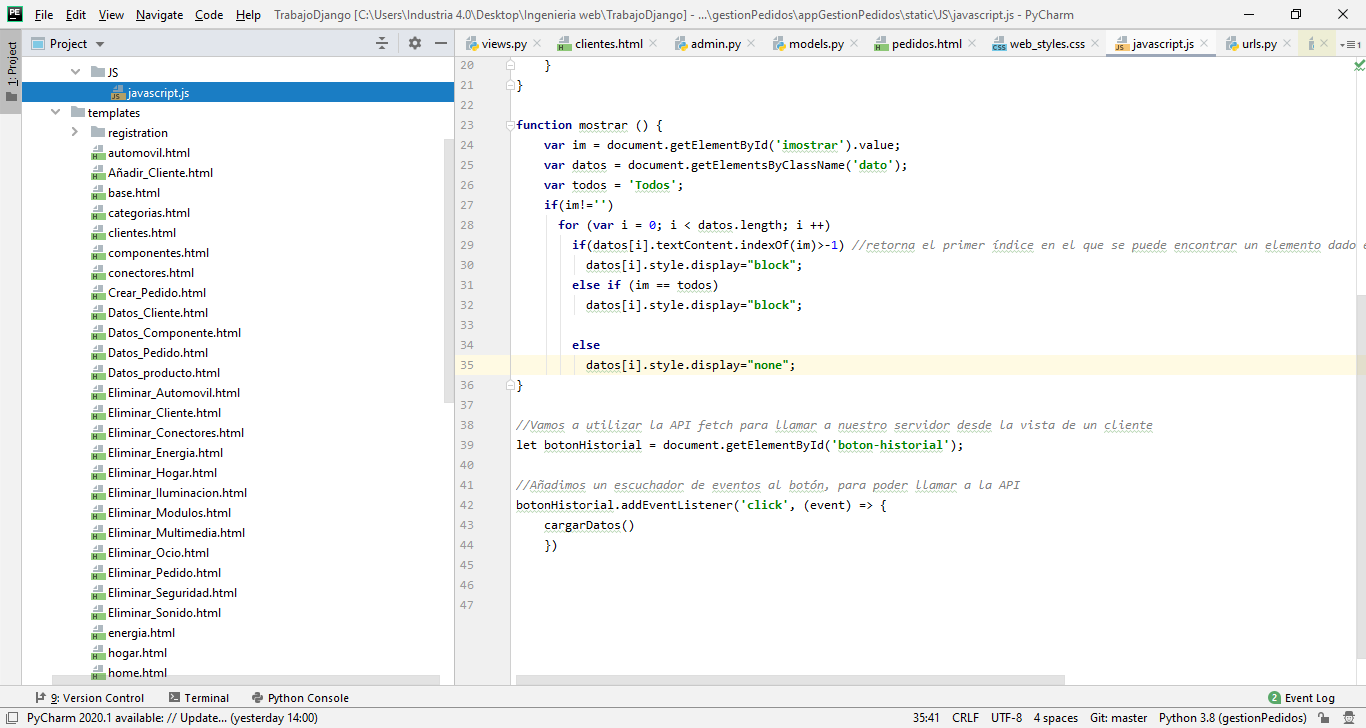
* Función para nombrar como Pendiente o Entregado el atributo Entregado del modelo de pedidos según si su valor es True (entregado) o False (pendiente)

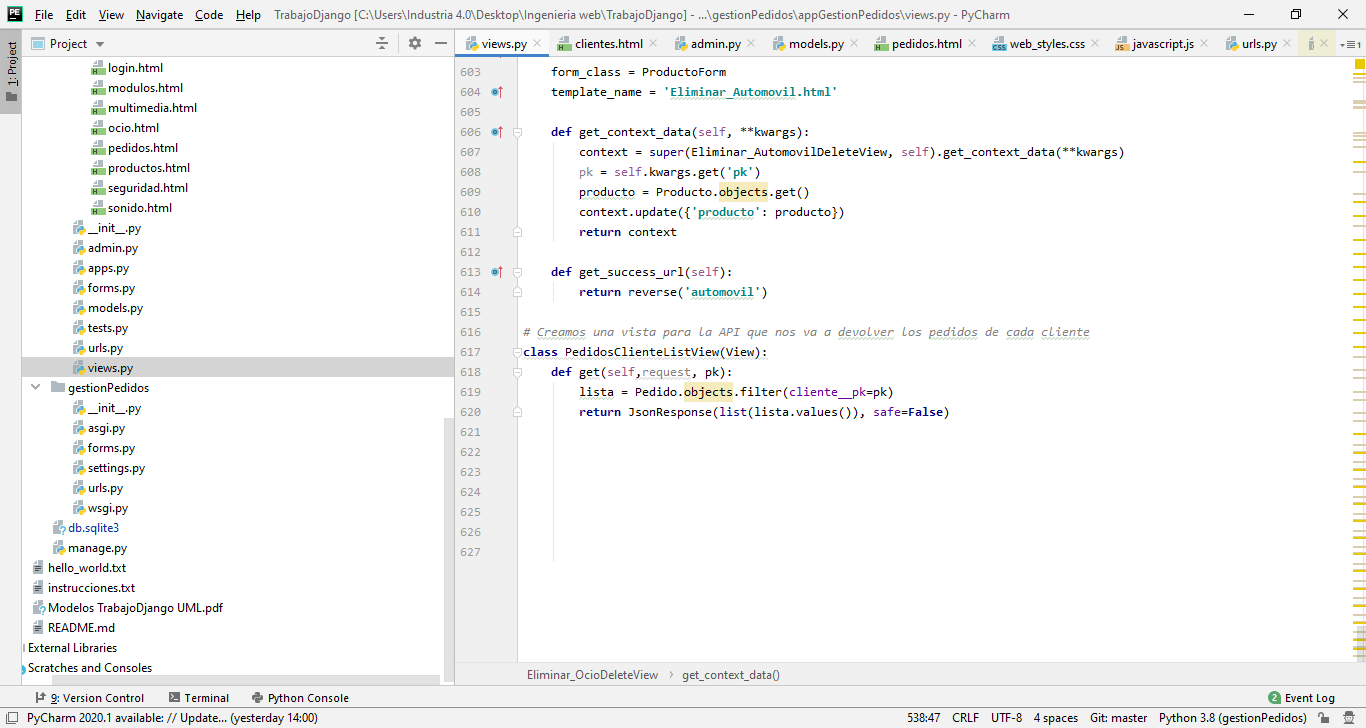


* Función para realizar una búsqueda con la cual podamos filtrar y mostrar todos los pedidos, solo los entregado o aquellos que estén pendientes de entrega.



- Funcionalidades JavaScript para cargar y almacenar datos utilizando Fetch:





**Inconvenientes actuales:**

* Uno de los compañeros dio la idea de hacer un registro para poder acceder al proyecto, esta acción requería de conocimientos poco más grandes que los que contenemos, pero finalmente se consiguió.
* Después de la primera entrega una vez revisando las funcionalidades de nuestro proyecto, nos dimos cuenta de que nuestras clases DeleteView habían dejado de hacer su función. Nuestra reacción fue inmediata, nos pusimos a revisar el código y vimos el cambio, lo volvimos a poner como debía y seguimos avanzando.
* Las funciones implementadas con JavaScrip dejaron de funcionar al estar todas juntas.

**Nuevas necesidades:**

* Con JavaScript :

Aplicar varias funciones con las cuales tu tuviéramos que recargar la página. Con lo cual pensamos varias para aplicarlas:

* Principalmente en los pedidos, cambiar el estado true o false y mostrar pendiente o entregado.
* Por otra parte, poder mostrar según las preferencias que tengamos, todos, solo los entregados o aquellos que estén pendientes.
* En la realización de un pedido realizar una función en la cual con varios aspectos del formulario nos calcule el precio total.

3.2 DESCRIPCION DE REQUISITOS DEL NUEVO SISTEMA

3.2.1 MODELO FUNCIONAL

**Descripción de los escenarios y funcionalidades planteadas:**

1º HOME

Esta pantalla contemplara la función de menú, es decir únicamente nos mostrara una pantalla en la cual visualizaremos enlaces a las distintas pantallas de nuestro proyecto (Home, Clientes, Catalogo, Componentes, Pedidos).

2º CLIENTES

En este apartado, se mostrará una tabla con los datos de todos nuestros clientes, conteniendo en cada uno de ellos la opción de eliminarlo o mostrarnos todos sus detalles al completo. A la parte inferior de esta tabla, contenemos un botón, con el cual podremos crear un cliente nuevo (pulsaremos y nos llevara a otra pantalla para rellenar un formulario con los datos del nuevo cliente).

3º CATALOGO

Apartado catálogo, en esta pantalla mostraremos todas las categorías de productos disponibles (clicando sobre ellos accederemos a una lista con todos sus productos) y al igual que en el anterior, en la parte inferior disponemos de un botón para crear un nuevo producto (accedemos a un formulario para crear el producto).

4º COMPONENTES

En componentes únicamente mostraremos una tabla con todos aquellos disponibles.

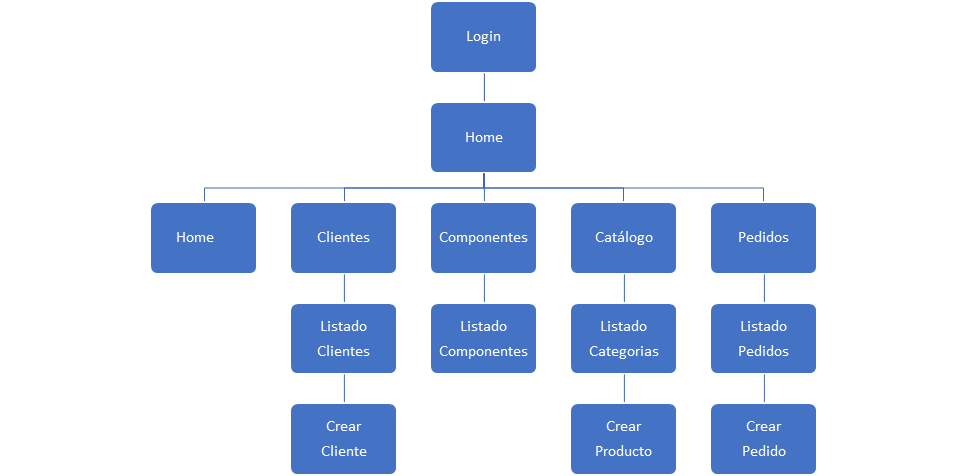
5º PEDIDOS

Pantalla pedidos, mostramos en una tabla todos los pedidos llevado a cabo, tanto los entregados como pendientes, junto a ello también estarán las funciones de eliminado y mostrado de cada uno. Para hacer de esta pantalla algo más claro, tenemos un buscador el cual podremos seleccionar si queremos ver todos, únicamente los entregados o los que están pendientes. En la parte inferior, repetimos el botón para crear un nuevo pedido.

**¿Que tienen en común todos nuestros escenarios?**

Todos ellos contienen el menú con acceso a las diferentes pantallas y en la parte superior, en la zona derecha de nuestro encabezado mostrara el usuario logeado y una opción para cerrar sesión.

**Descripción del modelo de datos diseñado:**



3.3 DESCRIPCION DE LA INTERFAZ DEL SISTEMA

3.3.1 PERFIL DE LOS USUARIOS

(novatos, entrenados, nivel educativo, experiencia en productos similares, necesidades de Formación…)

Si nos ponemos a analizar nuestra aplicación, podremos observar que hemos intentado hacerla lo más comprensible posible, es decir, desde nuestro punto de vista, una aplicación debe ser clara y funcional, con lo cual la hemos desarrollado de manera que incluso un novato pueda utilizarla. Dicho lo cual, no creemos que sea necesaria una formación, pero si tal vez una pequeña explicación de lo que se pueden encontrar.

**Capítulo 4: ESPECIFICACION DEL DISEÑO**

4.1 INTRODUCCION

4.1.1 PRINCIPALES FUNCIONES DEL SOFTWARE

4.1.2 DESCRIPCION DEL ENTORNO DE DESARROLLO

 Descripción de las plantillas utilizadas para las interfaces web (descripción de la estructura HTML y estilos CSS).

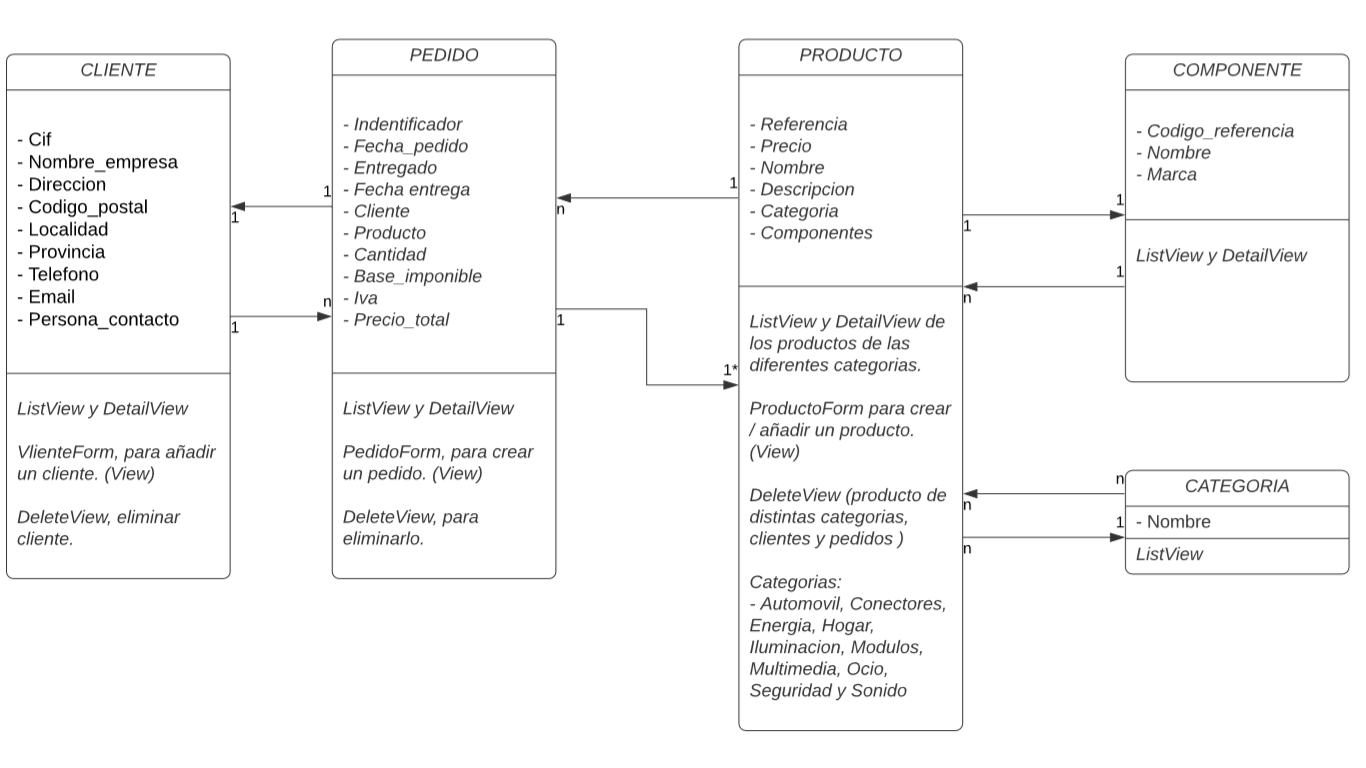
La estructura HTML parte de una base común, “base.html” de la que heredan todas las ventanas menos la de login que tiene su formato propio. En la base tenemos la información sobre la versión de html. También tenemos una cabecera que comparten todas las ventanas con el logotipo de la empresa, el nombre, el usuario activo en el momento y un botón de logout para salir de la aplicación. En la parte izquierda de la pantalla tenemos un menú de navegación para las distintas páginas y en la parte inferior el logotipo y contacto de la empresa desarrolladora. En la parte central tenemos el espacio para el body, el cual será el que se modifique entre pantallas.

4.2 ARQUITECTURA FISICA Y ENTORNO TECNOLOGICO

4.2.1 DESCRIPCION GENERAL

4.3 DESCRIPCION DEL DISEÑO (Para Programación por Eventos)

4.3.1 DISEÑO DE LA ESTRUCTURA FISICA DE LOS DATOS



4.3.1.1 DEFINICION DE VISTAS

Si analizamos generalmente las vistas que contenemos, estas son las que implementamos ListView, DetailView, DeleteView y View (para formularios).

ListView:

Principalmente esta Vista nos mostrara una lista de todos aquellos objetos creados de un modelo.

Como se implementa:

Pondremos un ejemplo:

*# DEVUELVE EL LISTADO DE CLIENTES***class** ClientesListView(ListView):// Definimos el tipo de vista

//definimos el modelo a utilizar  
 model = Cliente

//definimos la plantilla que mostrara dicha lista   
 template\_name = **'clientes.html'**

//nombre de la lista

context\_object\_name = **'lista\_clientes'**//aparte de los datos del cliente le diremos que pase el titulo de la página

**def** get\_context\_data(self, \*\*kwargs):

//cargamos el diccionario context  
 context = super(ClientesListView, self).get\_context\_data(\*\*kwargs)

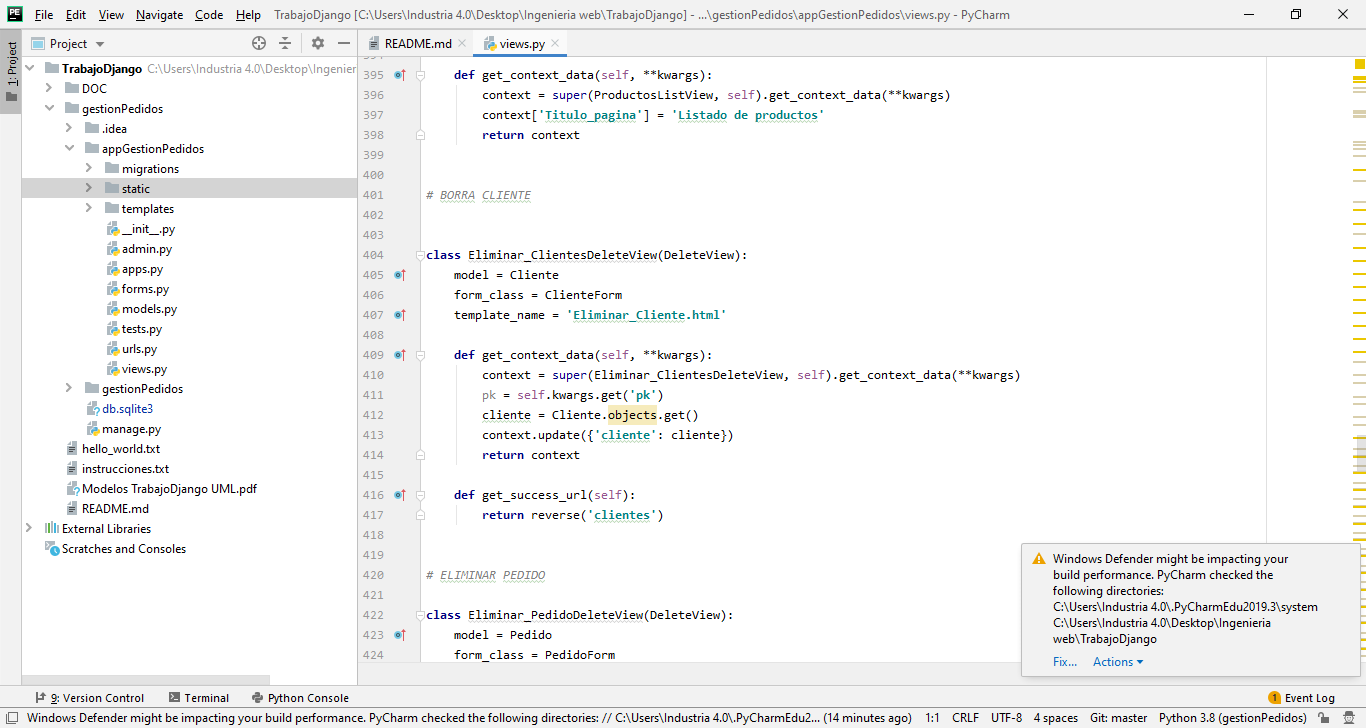
//añadimos la información que queremos, en este caso aparecerá como

Subtitulo.  
 context[**'Titulo\_pagina'**] = **'Listado de clientes'  
 return** context

DetailView:  
En este caso es una vista de tipo genérica que nos mostrara en detalle la información de un objeto.

DeleteView:

La acción de dicha vista será eliminar un objeto. El contexto de esta vista es parecido a las anteriores, el único cambio es que utilizaremos la pk (primarykey) con lo cual añadiremos este apartado.



View (Formularios):

Crearemos formularios para crear/añadir objetos.

Ejemplo cliente:

*# AÑADIR CLIENTE NUEVO***class** CrearClienteView(View):  
 **def** get(self, request, \*args, \*\*kwargs):  
 form = ClienteForm()  
 context = {  
 **'form'**: form,  
 **'titulo\_pagina'**: **'Añadir cliente'** }  
 **return** render(request, **'Añadir\_Cliente.html'**, context)  
  
 **def** post(self, request, \*args, \*\*kwargs):  
 form = ClienteForm(request.POST)  
 **if** form.is\_valid():  
 form.save()  
  
 *# Volvemos a la lista de clientes* **return** redirect(**'clientes'**)  
  
 **return** render(request, **'Añadir\_Cliente.html'**, {**'form'**: form})

**Modelo Cliente:**

Para este modelo disponemos de 4 vistas, ListView, DetailView, DeleteView y View (ClienteForm()).

**Modelo componente:**

En este caso solo tendremos ListView y DetailView.

**Modelo producto:**

Tenemos un ListView y DeleteView para cada categoría y la View para el formulario (ProductoForm()) y crear un nuevo producto.

**Modelo pedido:**

Al igual que los productos y los clientes tendrá ListView, DetailView, View (formulario) y DeleteView.

**Capítulo 7: MANUAL DE USUARIO**

**Capítulo 8: INCIDENCIAS DEL PROYECTO y CONCLUSIONES**

**BIBLIOGRAFIA**