

UNIVERSIDAD SAN CARLOS DE GUATEMALA

CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA CIENCIAS Y SISTEMAS



LABORATORIO ANÁLISIS Y DISEÑO 1

ING: JOSÉ MOISÉS GRANADOS GUEVARA

ESTUDIANTE:

201313852 - Jonathan Ubaldo Mate Jacinto

201730705 -Sergio Daniel Cifuentes Argueta

201730919 - Bryan René Gómez Gómez

TEMA: “Sistema de Inventario - DASHTORY”

FECHA: 28 de octubre de 2,021

ÍNDICE

ÍNDICE	2
REQUERIMIENTOS	4
Requisitos Mínimos:	4
Requerimientos de hardware para programar Django (Python)	4
Requerimientos de Software	4
Lenguaje de Programación Python 3	4
Django:	4
PostgreSQL:	4
Editor de Texto Visual Studio Code:	4
Librerías Varias	5
INSTALACIÓN	6
Descarga del proyecto	6
Creación de entorno virtual:	6
Instalación de librerías necesarias para el proyecto:	6
Archivo env de configuración:	6
Configuración de la BD en PostgreSQL:	7
Inserción de datos:	7
Ejecución del proyecto:	7
MARCO TEÓRICO	8
ESTRUCTURA DEL PROYECTO	9
Aplicaciones:	9
Directorio migrations	10
Directorio test	10
__init__.py	10
Admin.py	10
Apps.py	10
Forms.py	10
Test.py	10
Modelo-Vista-Template:	11
Model.py	11
View.py	11
Template.html	11
ESTRUCTURA DE UN TEMPLATE:	12
base.html:	12
inventory_base.html:	12
dashboard.html:	12
OTROS ARCHIVOS IMPORTANTES:	13
middleware.py:	13
urls.py:	13

settings.py:	13
DEFINICIÓN DE LAS APLICACIONES:	14
clients:	14
Common:	14
dashboard:	14
index:	14
products:	14
users:	15

REQUERIMIENTOS

Requisitos Mínimos:

Requerimientos de hardware para programar Django (Python)

Para las aplicaciones generadas se debe tener un mínimo de 1 GB de RAM, se recomienda que se tengan 2 GB o más. El procesador en principio no es tan crítico como la memoria RAM, pero se recomienda utilizar al menos un Core 2 Duo a 2.4 Ghz como mínimo .

Requerimientos de Software

Para trabajar con El framework django se necesita:

Lenguaje de Programación Python 3

Para la programación de nuestra página web se utilizó Python 3 específicamente la versión 3.9.5.

Django:

Se necesita instalar la versión de django 3.2.4, con esta versión de django se creó la estructura principal del proyecto.

PostgreSQL:

Como motor de base de datos para la aplicación se usó PostgreSQL en su versión 12.5 .

Editor de Texto Visual Studio Code:

Al desarrollar para un entorno web no se vio en la necesidad de un ide muy potente para el desarrollo de la aplicación, así que cualquier editor de código podría usarse, aunque el equipo de trabajo se decantó por Visual studio code por su simplicidad y las múltiples opciones de autocompletado para diversos lenguajes de programación.

Librerías Varias

Para la implementación de la aplicación se utilizaron diferentes librerías de python las cuales pueden encontrarse documentadas en el archivo **requirements1.txt** en la ruta principal del proyecto, en el proceso de instalación se detalla como instalar estas librerías de forma automática.

Sistemas operativos admitidos

- Windows 10
- Windows 8.1
- Windows 8
- Windows 7 SP1
- Linux/GNU y sus Derivados

INSTALACIÓN

Descarga del proyecto

Una vez teniendo instalado Python y Django en las versiones antes mencionadas debemos descargarnos el repositorio del proyecto:

<https://github.com/jmateo95/inventory>

Creación de entorno virtual:

Con el repositorio descargado procedemos a crear un entorno virtual de trabajo esto para que las librerías que son requeridas para el proyecto no interfieran con otros proyectos; esto lo hacemos ejecutando el comando

py -m venv nombre_del_entorno

Seguidamente activamos nuestro entorno virtual con el comando:

python .\ruta_del_entorno_virtual\manage.py runserver

Una vez activado podremos ver en nuestra consola el nombre de nuestro entorno virtual con esto sabremos que nuestro entorno virtual está activado.

Instalación de librerías necesarias para el proyecto:

Como anteriormente se mencionó todas las librerías necesarias para utilizar la aplicación están detalladas en el archivo **requirements1.txt** y para instalarlas debemos ejecutar el siguiente comando:

pip install -r requirements1.txt

Una vez ejecutado este comando (Cabe recalcar que todos estos comandos deben ejecutarse con el entorno virtual activado) se empezará con el proceso de instalación de nuestras librerías.

Archivo env de configuración:

Uno de los archivos que no se incluyen dentro del repositorio por motivos de seguridad pero que es esencial y primordial en el funcionamiento de la aplicación es un

fichero “.env” en el cual se guardan la información de conexión a la base de datos, información del correo de soporte, entre otras variables, se debe solicitar un ejemplo de este archivo para rellenarlo con la información adecuada para cada caso.

Configuración de la BD en PostgreSQL:

Una vez terminada la configuración de la aplicación debemos crear una base de datos con el nombre que hayamos especificado en nuestro archivo de configuración .env una vez creada la base de datos ejecutamos en nuestro proyecto el comando:

```
python .\src\manage.py migrate
```

Este comando conectará nuestra aplicación con nuestra base de datos recién creada y creará todas las tablas y esquemas que se definieron para el funcionamiento de la aplicación.

Inserción de datos:

En los archivos del proyecto podemos encontrar el fichero **inserts.sql** aquí existen inserciones iniciales que el proyecto necesita para funcionar por lo cual debemos copiar el script de inserción y ejecutarlo en nuestra base de datos para el correcto funcionamiento de nuestra aplicación.

Ejecución del proyecto:

Una vez realizadas todas las configuraciones anteriores podemos ejecutar nuestra aplicación con el comando:

```
python .\src\manage.py runserver
```

Este comando nos permitirá ejecutar una instancia de nuestra aplicación en nuestro local host específicamente en el puerto 8000.

MARCO TEÓRICO

Un sistema de control de inventarios te permite controlar los bienes y el stock, registrar los movimientos y hacer un seguimiento de las compras, manteniendo organizada la cadena de suministro sin ningún contratiempo. La buena organización y el funcionamiento de estos factores es determinante en el resultado que obtenga tu empresa, por lo que no es algo que se pueda ni deba manejar simplemente con la ayuda de lápiz y papel.

Estos sistemas se sirven de códigos de barras, imágenes o listas importadas para conocer las existencias en un almacén, con cálculos hechos a partir de las compras y ventas.

De cualquier manera, para que la gestión de inventarios sea efectiva, es necesario mantener una visión total de todo el proceso y mantener los niveles de stock más adecuados durante todo el año para no tener pérdidas.

En este sentido, los sistemas de control de inventarios suponen una gran ventaja, porque recogen información sobre los bienes durante todo el proceso, la producción, el embalaje, el almacenamiento y envío, y en definitiva, todos los movimientos que se hacen entre almacenes y sucursales.

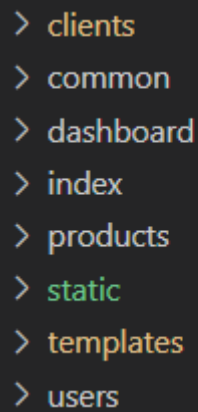
Aunque la gestión de inventarios varía en función del sector, las necesidades de la empresa y la inversión que esta quiera hacer, los sistemas de gestión de inventarios, en general, incluyen herramientas para generar informes, crear listas de chequeos, firmar y validar reportes y enviar emails al personal indicado, entre otras funcionalidades.

ESTRUCTURA DEL PROYECTO

Aplicaciones:

Django usa una estructura que divide el proyecto en aplicaciones, esto para tener una mejor escalabilidad con el tiempo.

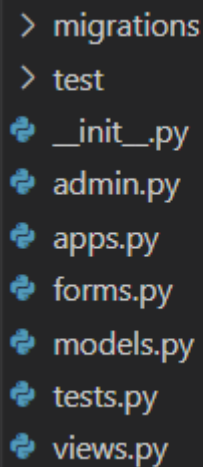
En el proyecto tenemos 6 aplicaciones



```
> clients
> common
> dashboard
> index
> products
> static
> templates
> users
```

Clientes, common, dashboard, index, products y users, además de existir 2 directorios más static (para archivos estáticos como imágenes, iconos, etc) y templates (almacena todos nuestros archivos html).

Cada una de estas aplicaciones tiene dentro de su estructura los siguientes archivos:



```
> migrations
> test
+ __init__.py
+ admin.py
+ apps.py
+ forms.py
+ models.py
+ tests.py
+ views.py
```

Directorio migrations

Alberga todos los cambios que se le deben realizar a la base de datos, este directorio se crea automáticamente a partir del directorio **models.py** cuando ejecutamos el comando **py makemigration**, por lo cual son configuraciones necesarias para nuestra base de datos, pero se podría decir que son archivos que se generan de forma automática.

Directorio test

Alberga archivos con formato `test_algo.py` los cuales son ficheros en los que se definen las pruebas unitarias realizadas a los métodos definidos de nuestra aplicación.

`__init__.py`

Es un fichero de generación automática el cual se crea al ejecutar nuestro proyecto.

Admin.py

Es un fichero en el cual podremos publicar los modelos que definimos en el fichero `models.py`, esto para que podamos administrarlos desde el panel de administración de django (el registro de los modelos es opcional).

Apps.py

Es el fichero en el cual se registra el nombre de nuestra aplicación, este nombre nos servirá posteriormente para hacer uso de nuestra aplicación en nuestro proyecto.

Forms.py

Es un fichero el cual no se genera automáticamente al crear la aplicación dentro de nuestro proyecto, este fichero puede crearse manualmente y es usado para separar lógica de programación. Su función principal es la definición de formularios web, los cuales son usados en los templates para solicitar información a un usuario.

Test.py

Es el fichero que crea Django automáticamente al crear la aplicación para la implementación de pruebas, es recomendable no utilizarlo y crear un directorio test el cual mantendrá nuestra lógica ordenada de mejor forma.

Modelo-Vista-Template:

Django facilita la construcción de aplicación debido a su filosofía de implementación de código.

Model.py

Este fichero es el encargado de manipular la información de nuestra aplicación, aquí definimos nuestras entidades, atributos y sus tipos, es aquí donde Django utiliza su ORM para facilitarnos el acceso a la información (como si de un objeto en memoria se tratara.).

View.py

En este fichero es en donde definimos toda la lógica de nuestra aplicación, es un enlace entre el modelo y el template. Decide qué información será mostrada y por cual template.

Template.html

Este fichero no se aloja en la estructura de nuestra aplicación, además de que no tiene porque llamarse template.html.

Son ficheros los cuales encontraremos en el directorio templates y siempre con una extensión .html en ellos se decide como será mostrada la información que se envió directamente desde la view.py.

ESTRUCTURA DE UN TEMPLATE:

Para facilitar el desarrollo de templates de una forma más fácil, Django nos permite crear templates generales los cuales pueden albergar secciones parciales de código, creando bloques de código en los cuales puede incrustarse información de otro template, por ejemplo:

base.html:

Ubicado en la ruta `/src/inventory/templates/base.html`, este archivo carga todos los css, js, librerías, etc. Prácticamente todos los archivos necesarios para nuestro tema y que nuestros templates se renderizan de forma correcta, aquí podemos ver que se establecen varios bloques los cuales podrán rellenarse con información de otro template.

inventory_base.html:

Ubicado en la ruta `/src/inventory/templates/inventory_base.html`, este archivo extiende del archivo `base.html`, en este se rellenan de información los bloques que se definieron en el archivo `base.html`, además de que en él podemos ver que se definen nuevos bloques y estos son rellenos con información de otros templates, básicamente es el esqueleto principal de nuestros templates, ya que aquí definimos el lugar de la sidebar, navbar, menus, etc.

dashboard.html:

Ubicado en la ruta `/src/inventory/templates/dashboard/ dashboard.html`, es un template final el cual podrá ser llamado para renderizarse desde un archivo `view.py`, aquí desplegamos nuestro html y definimos como vamos a mostrar la información.

OTROS ARCHIVOS IMPORTANTES:

middleware.py:

Ubicado en la ruta `/src/inventory/common/middleware.py` en este archivo se define qué usuario y qué rol tiene acceso a qué rutas, en pocas palabras es el archivo responsable de implementar la seguridad en las rutas de nuestra aplicación.

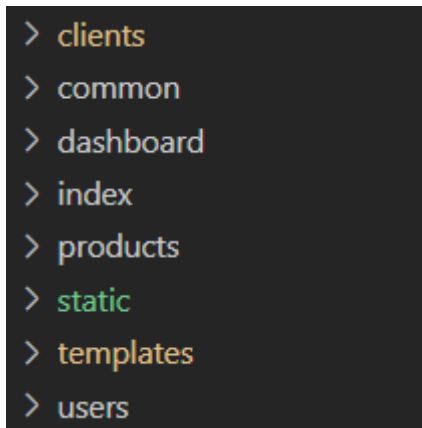
urls.py:

Ubicado en la ruta `/src/inventory/urls.py` en este archivo se definen las rutas de nuestra aplicación, aquí creamos una ruta, le damos un nombre y asociamos un método el cual se disparara al momento de acceder a esa ruta.

settings.py:

Ubicado en la ruta `/src/settings.py` este es un archivo de configuración de nuestra aplicación en el cual debemos registrar todas las aplicaciones que nuestra aplicación pueda alcanzar, además de muchas otras configuraciones, tales como zona horaria, dirección de archivos estáticos, etc.

DEFINICIÓN DE LAS APLICACIONES:



clients:

Esta aplicación alberga toda la lógica referente al CRUD de los clientes y al CRUD de las ventas (transacciones).

Common:

Esta aplicación alberga la lógica de la seguridad por roles a las diferentes rutas del proyecto.

dashboard:

Esta aplicación alberga la lógica que fue necesaria para crear al dashboard principal de nuestro sistema de inventario.

index:

Esta aplicación alberga la lógica de nuestra página principal.

products:

Quizás la aplicación más importante así como el corazón del sistema de inventarios ya que en ella se alberga la lógica referente al CRUD de los proveedores, CRUD de tipos de

productos, CRUD de grupos de productos, CRUD de categorías de productos y el CRUD de las órdenes de productos.

users:

Es la aplicación que alberga la lógica referente al CRUD de los usuarios autorizados para nuestro sistema de inventario.