

UNIVERSIDAD DE EL SALVADOR  
FACULTAD MULTIDISCIPLINARIA ORIENTAL  
DEPARTAMENTO DE INGENIERIA Y ARQ.



## TECNICAS DE SIMULACION

---

“Sistema de Inventario de la Farmacia UES | FMO”

DOCENTE: Ing. Luis Jovanni Aguilar

### ALUMNOS:

Luis José Iraheta Medrano	IM15005.
Yenifer Zuleyma García Meléndez	GM15002.
Claudia Patricia Salamanca Martínez	SM13038.
Ana Ruth Sánchez Henríquez	SH13012.

CIUDAD UNIVERSITARIA ORIENTAL, JUEVES 21 DE JUNIO DE 2018

## Indice

INTRODUCCIÓN .....	3
OBJETIVOS.....	4
Objetivo general: .....	4
Objetivos específicos:.....	4
DEFINICIONES .....	5
¿Qué es un framework? .....	5
¿Qué es Django?.....	5
¿Qué hace Django?.....	5
ARBOL DE ARCHIVOS DEL PROYECTO .....	6
DESCRIPCION DEL PROYECTO.....	8
__init__.py .....	8
settings.py .....	8
urls.py .....	8
wsgi.py.....	8
EXPLICACION DE PROCESOS CREADOS EN EL CODIGO .....	15
CONCLUSION .....	21

## INTRODUCCIÓN

Con los grandes avances en la tecnología farmacéutica, los frecuentes cambios y el impacto de estos en las empresas farmacéuticas, estas buscan mejorar la calidad y la competitividad para mantenerse en el mercado laboral, es por ello que se ha decidido la implementación mediante un proyecto de sistema de inventario de la FARMACIA UES | FMO el cual permitirá administrar de manera más eficiente, para llevar un mejor control de las ventas y existencias de los productos.

En el cual los presentes trabajos se siguieron pasó a paso las actividades realizadas en clases en la metodología de desarrollo específica, La implementación fue ejecutada mediante un proyecto de desarrollo en Django framework de desarrollo web escrito en Python que consiste en una serie de librerías creadas con el fin de hacer el desarrollo de aplicaciones web más rápido y cómodo para los desarrolladores. Entre las facilidades que proporciona Django está la implementación de un sistema de plantillas, un ORM bastante potente, servidor para entorno de desarrollo, base de datos; en donde se implementó el servidor de base de datos en Mysql.

El sistema de inventario realizara el control de registro de medicamentos para una buena administración de la farmacia, para que no existan errores al momento de realizar los ajustes o reportes de todos los medicamentos que existe en la institución, el programa brindara beneficios de:

- Registro de entrada y salida de medicamentos.
- Búsqueda de medicamentos.
- Reportes de stock de medicamentos.
- Con dicho programa el propietario u encargado estará ofreciendo a los clientes una rápida y eficaz atención al momento de hacer sus compras y así también se tendrá un crecimiento en el ámbito comercial.

## OBJETIVOS

### **Objetivo general:**

Diseñar y desarrollar un sistema de inventario para la Farmacia UES/FMO.

### **Objetivos específicos:**

- Diseñar un modelo de datos basado en los lineamientos y planes de mejoramiento que permita administrar y almacenar toda la documentación.
- Facilitar el ingreso de la información para que los datos se mantengan actualizados y contar con información real de la operación.
- Determinar la política de inventarios para medicamentos y dispositivos médicos.

## DEFINICIONES

### **¿Qué es un framework?**

Un framework es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

Los framework están basados en patrones de desarrollo, normalmente MVC (Modelo-Vista-Controlador) que ayudan a separar los datos y la lógica de negocio de la interfaz con el usuario

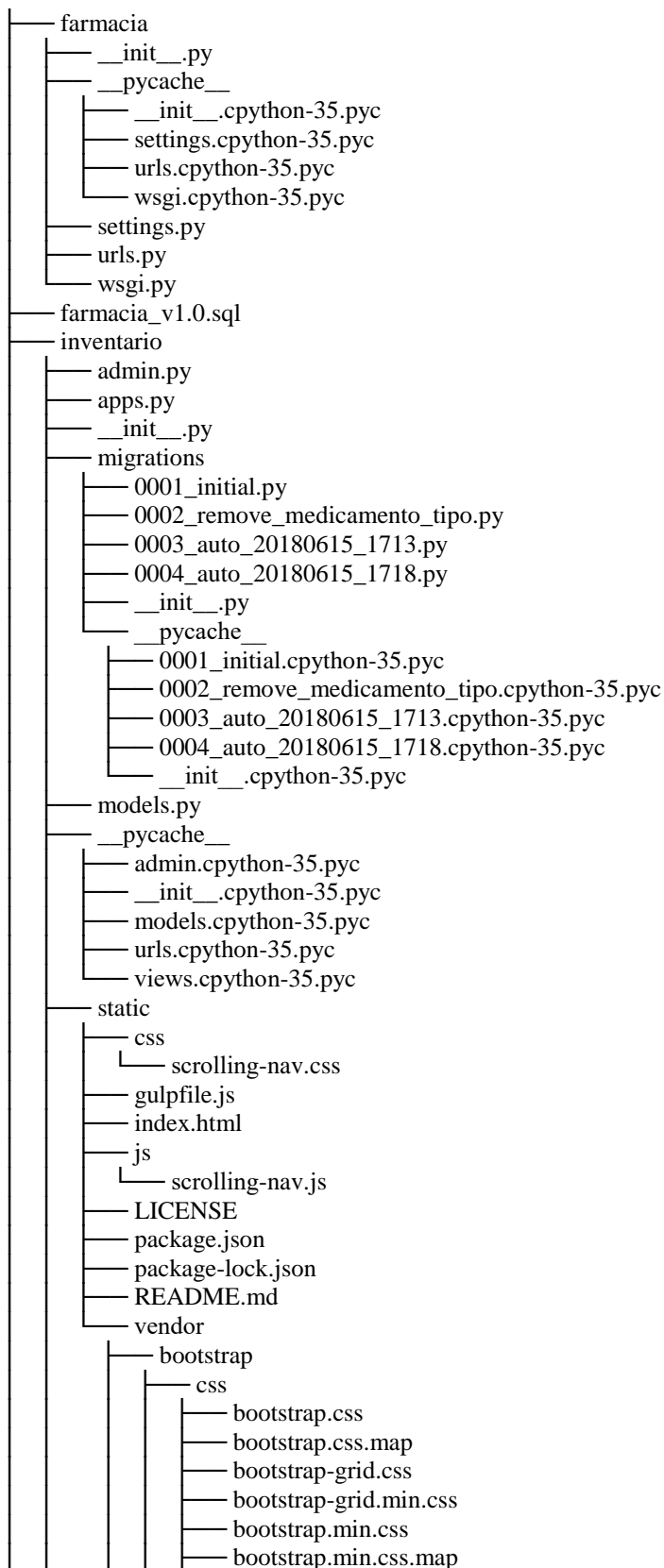
### **¿Qué es Django?**

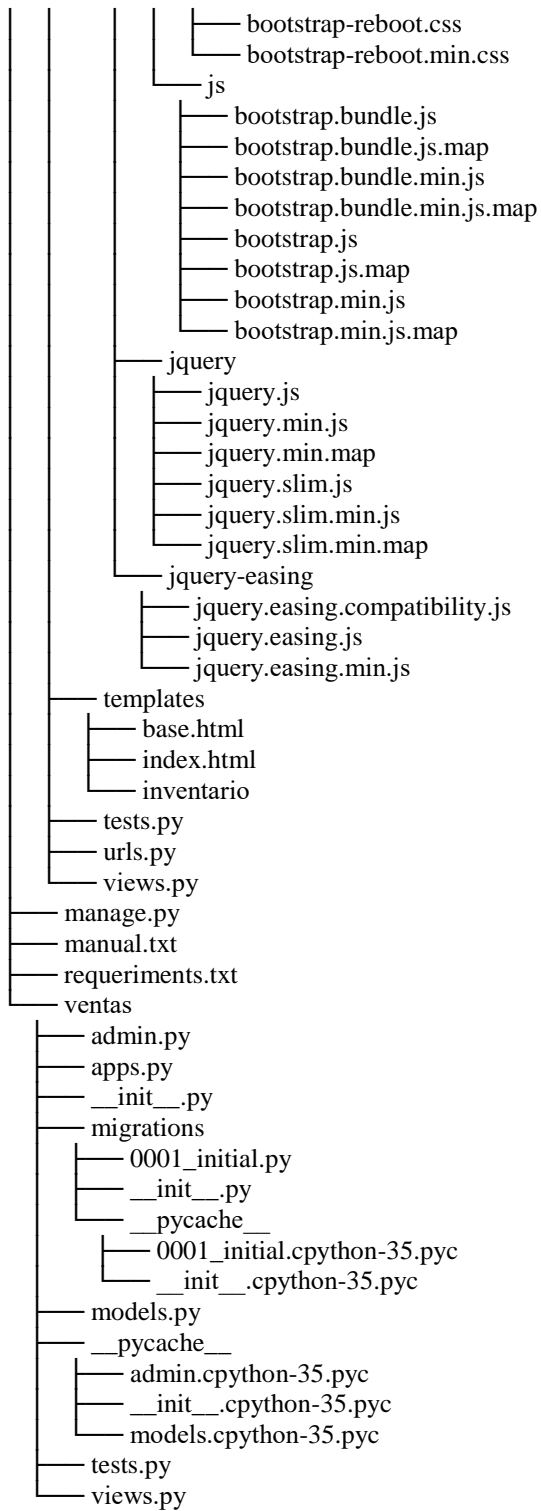
Django es un framework web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático. Desarrollado por desarrolladores experimentados, se encarga de gran parte de las complicaciones del desarrollo web, por lo que puede centrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto.

### **¿Qué hace Django?**

1. Provee una estructura de trabajo bajo el patrón Model Template View.
2. Mapea objetos Python con la base de datos(ORM)
3. Permite diseñar Urls amigables para buscadores (útil para SEO)
4. Sistema de plantillas sencillo para diseñadores.
5. Genera una interfaz de administración automática
6. Puede gestionar formularios, sesiones de usuario, autenticación, cache, almacenamiento, etc.

## ARBOL DE ARCHIVOS DEL PROYECTO





## DESCRIPCION DEL PROYECTO

Se elaboró un sistema de inventario de una farmacia, utilizando un framework para aplicaciones web Django con llenado de configuraciones específicas del sitio, urls, modelos, vistas y plantillas.

Django-admin para crear la carpeta del proyecto, los ficheros de plantillas básicos y el script de gestión del proyecto (**manage.py**). que es un script para ejecutar las herramientas de Django para este proyecto (creadas usando django-admin).

Una carpeta migraciones que se utiliza para guardar las "migraciones".

Ficheros que permiten actualizar las bases de datos a medida que modificas tus modelos.

**\_\_init\_\_.py** — Un fichero vacío creado para que Django/Python reconozca la carpeta como un Paquete Python permita usar sus objetos dentro de otras partes del proyecto.

**settings.py** contiene todos los ajustes del sitio. Es donde registramos todas las aplicaciones que creamos, la localización de nuestros ficheros estáticos, los detalles de configuración de la base de datos, etc.

**urls.py** define los mapeos urls-vistas. A pesar de que éste podría contener todo el código del mapeo urls, es más común delegar algo del mapeo a las propias aplicaciones, como verás más tarde.

**wsgi.py** se usa para ayudar a la aplicación Django a comunicarse con el servidor web. Puedes tratarlo como código base que puedes utilizar de plantilla.

El script **manage.py** se usa para crear aplicaciones, trabajar con bases de datos y empezar el desarrollo del servidor web.

La herramienta crea una nueva carpeta y la rellena con ficheros para las diferentes partes de la aplicación.

La mayoría de los ficheros se nombran de acuerdo a su propósito, para que sea más útil

(ej. las vistas se guardan en **views.py**, los Modelos en **models.py**, las pruebas en **tests.py**, la configuración del sitio de administración en **admin.py**, el registro de aplicaciones en **apps.py**) y contienen algo de código base mínimo para trabajar con los objetos asociados.



Primeramente, consta de un login, (imagen 1.1), mediante el cual se controla el acceso individual al sistema, como un inicio de sesión, lo cual permite acceder a todas las funciones, donde muestra los siguientes módulos, los cuales son: (imagen 1.2).

❖ Inicio

❖ Ver sitios: en este se encuentra los enlaces rápidos y las acciones recientes.

En los enlaces rápidos se puede: volver al sitio, cambiar contraseña y cerrar sesión. Y las acciones recientes como un historial en donde hemos accedido dentro del sistema

❖ El área de aplicaciones se encuentra el inventario y las ventas.

En inventario están todas las categorías y medicamentos y en ventas se generan las facturas.

Y también está la parte de noticias sobre Django

❖ En el área de administración, está la autenticación y autorización de grupos y usuarios.

Y el soporte, sobre la documentación de Django.

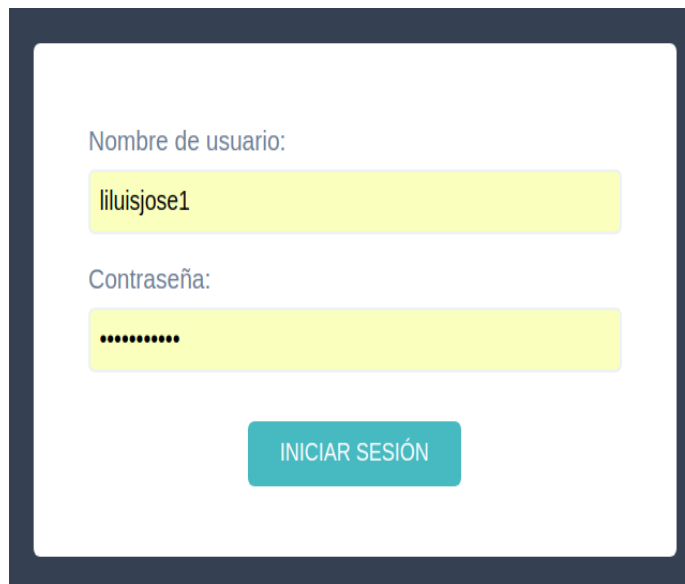
A login form with a dark blue border. It contains two input fields with light yellow backgrounds. The first field is labeled 'Nombre de usuario:' and contains the text 'liluisjose1'. The second field is labeled 'Contraseña:' and contains a series of dots. Below the fields is a teal button with the text 'INICIAR SESIÓN'.

Imagen 1.1

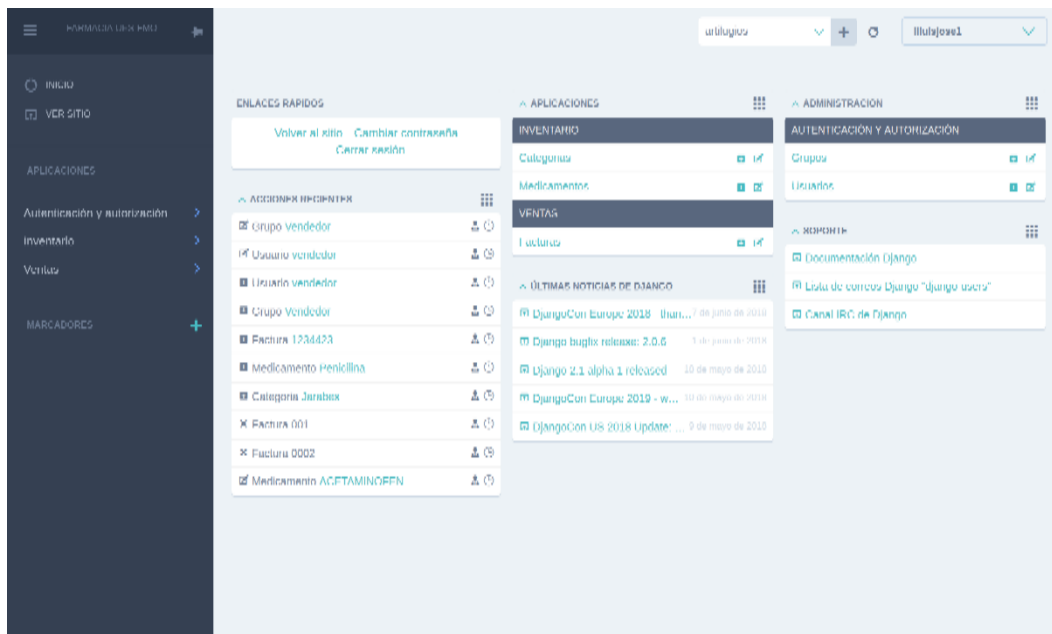


Imagen 1.2

Si accedemos a categoría, nos da la opción de:

- Añadir categorías, en donde podemos agregar el nombre de la categoría y por ende guardarla. (imagen 1.3)

En el área de inventarios se encuentran todos los medicamentos registrados, en donde podemos buscar por código.

Y se encuentra los siguientes datos de cada medicamento (imagen 1.4)

- La categoría
- El nombre
- Descripción
- Fecha de creación
- Precio de compra
- Precio de venta
- Stock

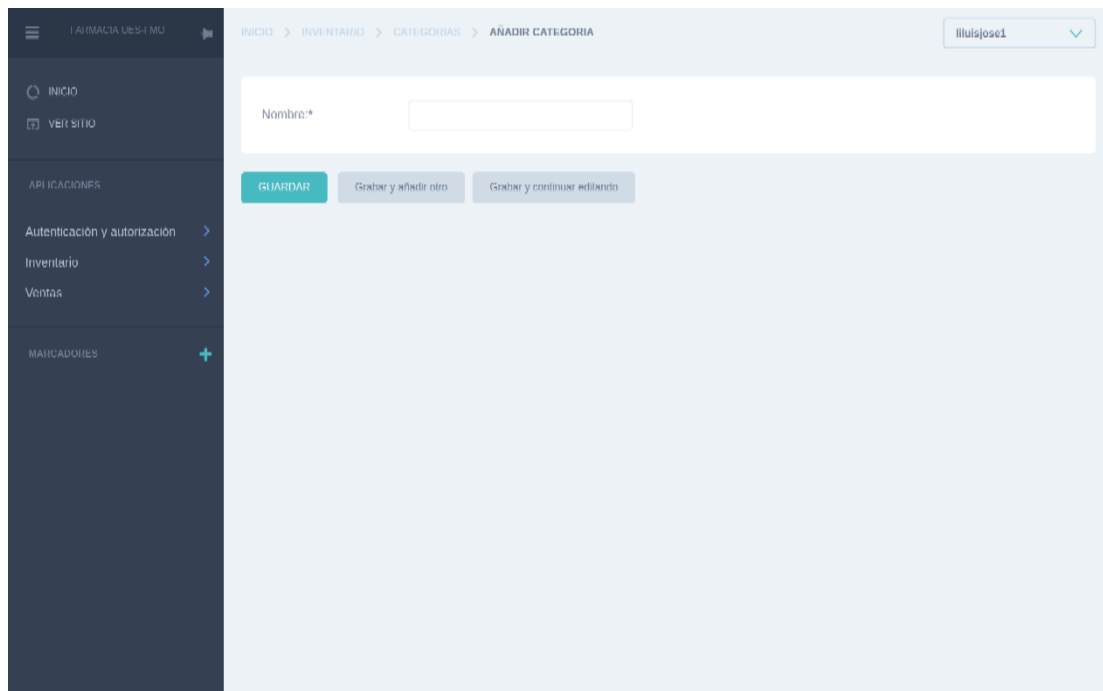


Imagen 1.3

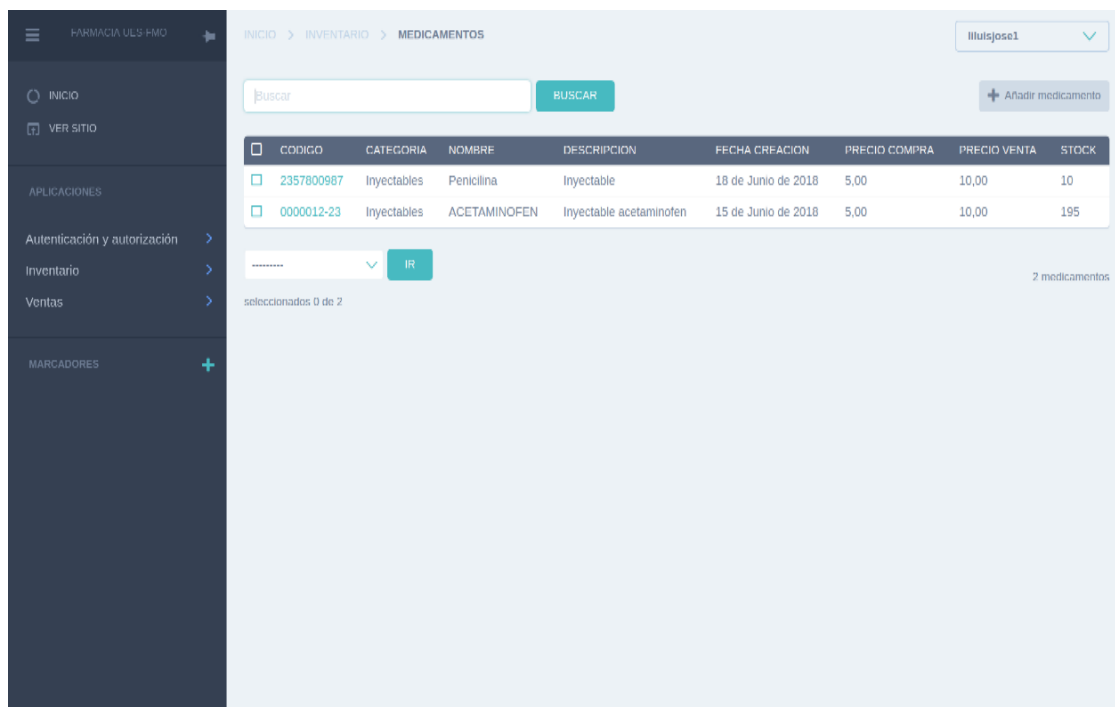


Imagen 1.4

Como se pudo observar en la imagen anterior en el área de medicamentos, se pueden añadir más.

Agregándoles los siguientes datos: (imagen 1.5)

Código

Categoría

Nombre

Fecha de creación

Descripción

Precio de compra

Precio de venta

Y el stock de cuantos productos se van a añadir

Después de todo esto se procede a guardar y ya queda registrado en el inventario.

También en el área de venta se generan lo que son facturas por código de medicamentos.

(imagen 1.6)

Y por consiguiente se puede añadir una nueva factura.

The screenshot shows the 'AÑADIR MEDICAMENTO' (Add Medication) form within the 'FARMACIA DESI MO' application. The interface includes a dark sidebar with navigation options like 'INICIO', 'VER SITIO', 'APLICACIONES', 'Autenticación y autorización', 'Inventario', 'Ventas', and 'MARCADORES'. The main content area has a breadcrumb trail: 'INICIO > INVENTARIO > MEDICAMENTOS > AÑADIR MEDICAMENTO'. The form fields are as follows:

- Código:\***: A text input field.
- Categoría:\***: A dropdown menu with a search icon and a close icon.
- Nombre:\***: A text input field.
- Fecha creacion:\***: A date picker.
- Descripción:\***: A large text area.
- Precio Compra:\***: A text input field with the value '0.0'.
- Precio venta:\***: A text input field with the value '0.0'.
- Stock:\***: A text input field.

At the bottom of the form, there are three buttons: 'GUARDAR' (Save), 'Grabar y añadir otro' (Save and add another), and 'Grabar y continuar editando' (Save and continue editing).

Imagen 1.5

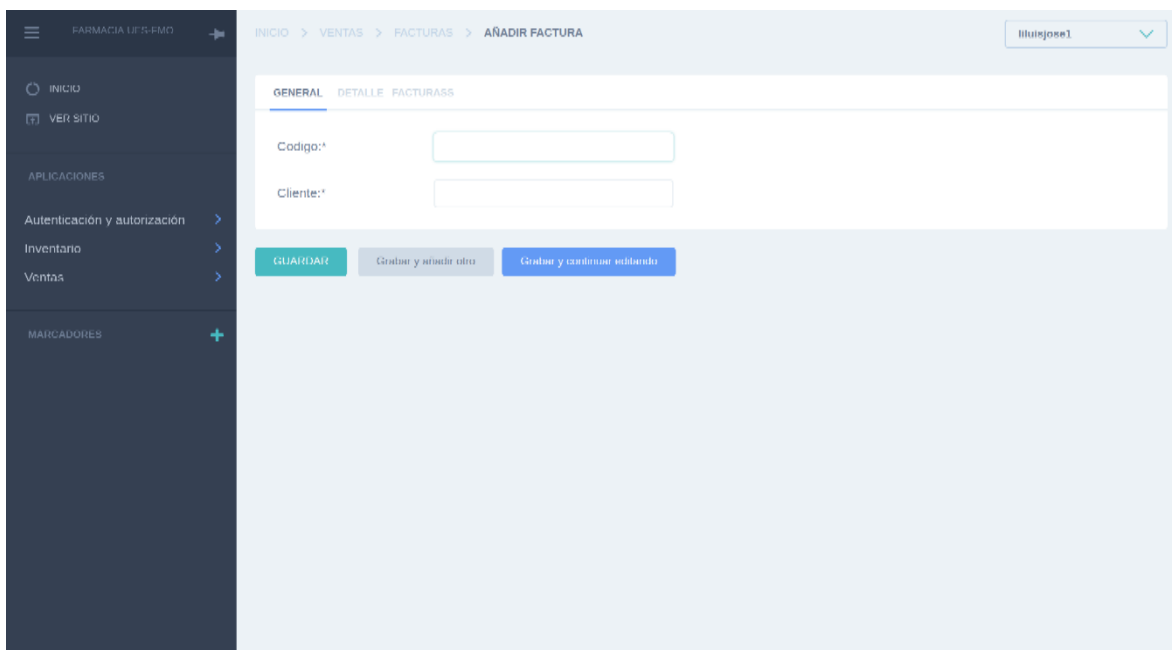


Imagen 1.6

Cuando se añade una factura se escribe

- El código de la factura y el nombre del cliente. Y la opción de guardar (imagen 1.7).

En esta sección del sistema consta de lo general de la factura y la factura a detalle. En la factura a detalle va agregado: (imagen 1.8)

- el medicamento, y la cantidad que se compró.

También podemos eliminar o agregar otro detalle de factura y por ende guardar dichos cambios.

INICIO > VENTAS > FACTURAS > AÑADIR FACTURA
 lluisjose1

GENERAL DETALLE\_FACTURASS

Codigo:\*

Cliente:\*

GUARDAR Grabar y añadir otro Grabar y continuar editando

Imagen 1.7

INICIO > VENTAS > FACTURAS > AÑADIR FACTURA
 lluisjose1

GENERAL DETALLE\_FACTURASS

MEDICAMENTO	CANTIDAD	¿ELIMINAR?
<input type="text"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="text"/>	
<input type="text"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="text"/>	
<input type="text"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="text"/>	

+ Añadir otro Detalle facturas

GUARDAR Grabar y añadir otro Grabar y continuar editando

Imagen 1.8

## EXPLICACION DE PROCESOS CREADOS EN EL CODIGO

A continuación, se ejecuta el siguiente comando para crear la aplicación *FarmaciaUES* que vivirá dentro de nuestro proyecto *Farmacia* (éste se ejecuta en la misma del **manage.py** del proyecto):

El directorio actualizado del proyecto tiene el aspecto siguiente:

```
1  #!/usr/bin/env python
2  import os
3  import sys
4
5  if __name__ == "__main__":
6      os.environ.setdefault("DJANGO_SETTINGS_MODULE", "farmacia.settings")
7      try:
8          from django.core.management import execute_from_command_line
9      except ImportError as exc:
10         raise ImportError(
11             "Couldn't import Django. Are you sure it's installed and "
12             "available on your PYTHONPATH environment variable? Did you "
13             "forget to activate a virtual environment?"
14         ) from exc
15     execute_from_command_line(sys.argv)
16
```

Los modelos están definidos, normalmente, en el archivo **models.py** de la aplicación. Son implementados como subclases de `django.db.models.Model`, y pueden incluir campos, métodos y metadata. El fragmento de código más abajo muestra un modelo que se utilizó para el proyecto.

```

1  from django.db import models
2
3  # Create your models here.
4  from django.db.models import signals
5
6
7  # Create your models here.
8  class Categoria(models.Model):
9      #codigo = models.CharField(max_length=10, unique=True)
10     nombre = models.CharField(max_length=60)
11
12     def __str__(self):
13         return u'%s' % (self.nombre)
14
15     class Medicamento(models.Model):
16         codigo = models.CharField(max_length=10, unique=True)
17         categoria = models.ForeignKey(Categoria, null=True, on_delete=models.SET_NULL)
18         nombre = models.CharField(max_length=200, unique=True)
19         fecha_creacion = models.DateField()
20         #f_produccion = models.DateField()
21         descripcion = models.TextField(max_length=400)
22         precio_compra = models.DecimalField(max_digits=5, decimal_places=2, default=0.00)
23         precio_venta = models.DecimalField(max_digits=5, decimal_places=2, default=0.00)
24         stock = models.PositiveSmallIntegerField()
25

```

Son la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica va aquí. Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.

Y quedó de la siguiente manera

```

1  from django.shortcuts import render
2  from django.utils import timezone
3  from .models import Categoria, Medicamento
4  from ventas.models import Factura, Detalle_Facturas
5  from django.shortcuts import redirect
6
7
8  # Create your views here.
9  from django.shortcuts import render, get_object_or_404
10 def index(request):
11     """
12     View function for home page of site.
13     """
14     # cantidad de objetos del modelo post
15     categorias=Categoria.objects.all()
16     medicamentos=Medicamento.objects.all()
17     factura=Factura.objects.all()
18     d_factura=Detalle_Facturas.objects.all()
19
20     # Render the HTML template index.html with the data in the context variable
21     return render(
22         request,
23         'index.html',
24         context={'num_cat':categorias, 'num_med':medicamentos, 'fact':factura, 'd_fac':d_factura},
25     )

```



Farmacia UES | FMO

Inicio

Login

Medicamentos

Codigo	Nombre	Categoria	Fecha Creacion	Descripcion	Precio Compra	Precio Venta	Stock
0000012023	ACETAMINOFEN	2	15 de Junio de 2018	Injectable acetaminofen		10,00	195
7406137000	MELOXIDOL TAB.	1	18 de Junio de 2018	--		10,00	200
4104480705	ABRILAR JB	1	18 de Junio de 2018	--		100,00	15
8901790689	ACEITE DE AJO Y PEREJIL SAIME	3	18 de Junio de 2018	--		1,00	200
7453001303	ACERTAR 750MG	1	18 de Junio de 2018	--		40,00	100
8901790683	ACETAMINOFEN TABLETAS SAIMED	1	19 de Junio de 2018	---		100,00	500

Categorias

ID	Nombre
1	No Hay
2	Injectables
3	Jarabes

Facturas

Codigo	Cliente	Fecha
--------	---------	-------

Created by

Luis Jose Iraheta Medrano IM15005

Yennifer Zuleyma Garcia Melendez GM15002

Claudia Patricia Salamanca Martinez SM13038

Ana Ruth Sanchez Henriquez SH13012

Copyright © Tecnicas de Simulacion Django 2018

Sin duda, uno de los archivos más importantes de todo el proyecto, toda la configuración que usará Django en nuestro proyecto estará dentro de este archivo, por lo que se debe tener mucho cuidado al momento de manipularlo. Por defecto Django te trae solo un archivo **settings.py**, veamos cual es el contenido de este archivo.

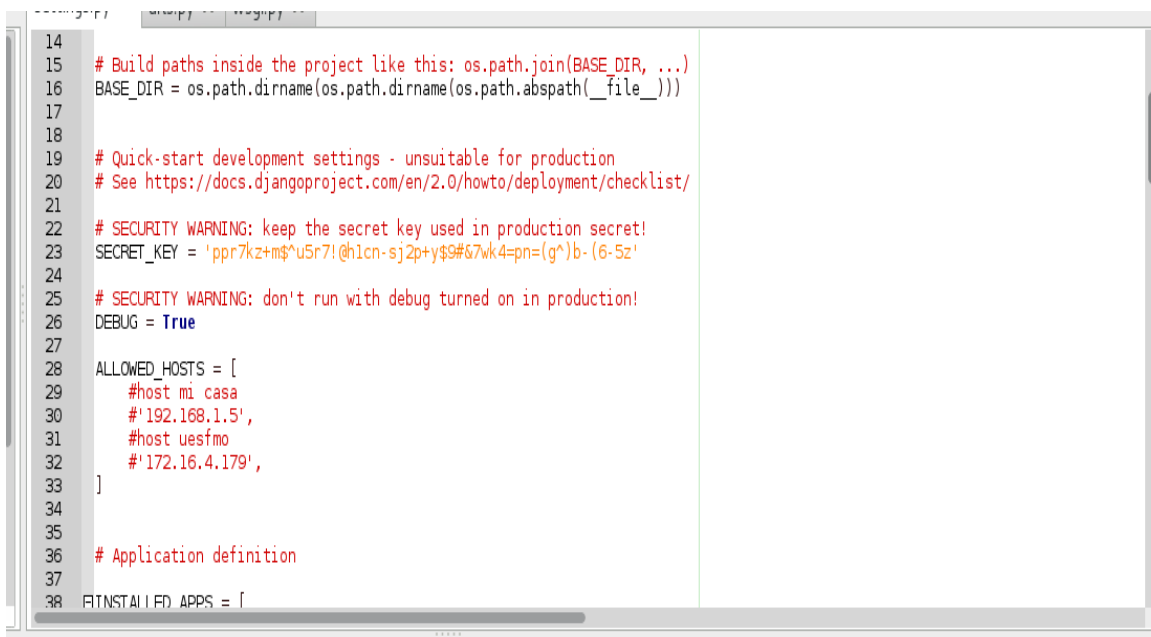
Este es el archivo **settings.py** que Django crea por defecto (según la versión de Django, ya que tiende a tener algunas variaciones), toda configuración siempre está asociado al nombre de una variable, así que al momento de hacer las modificaciones siempre se toma en cuenta eso, las variables no pueden ser modificadas porque Django simplemente ya no las tomará en cuenta. Se procede a explicar las configuraciones más importantes que trae ese archivo.

## BASE\_DIR

Esta es la variable que tiene por contenido la ruta a una determinada zona de tu proyecto, por defecto lleva la ruta al archivo **settings.py**.

## SECRET\_KEY

Todo proyecto en Django tiene esta variable que lleva una clave secreta, el contenido de esta variable siempre debe estar protegida y nunca escrita como texto plano dentro del archivo, por defecto Django la pone así pero ya es trabajo del desarrollador protegerla.

A screenshot of a code editor window showing a portion of a Django settings.py file. The code is color-coded: comments are in red, strings in green, and code in black. Line 23 contains the SECRET\_KEY assignment, which is highlighted with a light blue background. The key is a long, complex string: 'ppr7kz+ms^u5r7!@h1cn-sj2p+y\$9#67wk4=pn=(g^)-{6-5z'.

```
14
15 # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
16 BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'ppr7kz+ms^u5r7!@h1cn-sj2p+y$9#67wk4=pn=(g^)-{6-5z'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = [
29     #host mi casa
30     #'192.168.1.5',
31     #host uesfmo
32     #'172.16.4.179',
33 ]
34
35
36 # Application definition
37
38 INSTALLED_APPS = [
```

## DEBUG

Variable booleana solo acepta True o False. True cuando tu proyecto está en modo de depuración, esto lo notas fácilmente cuando te sale algún error y ves la pantalla amarilla de Django donde te muestra el detalle del error obtenido. False cuando el proyecto ya no se encuentra en modo de depuración, esto se suele utilizar cuando el proyecto ya se encuentra en producción.

## ALLOWED\_HOSTS

Hosts permitidos para el proyecto, mientras el DEBUG este en True esta variable puede permanecer vacía, cuando DEBUG cambie a False debes agregar un host obligatorio que puede ser el dominio de tu sitio.

## INSTALLED\_APPS

Variable donde agregamos las aplicaciones de terceros que vayamos utilizando o las que vayamos creando.

```
27
28 ALLOWED_HOSTS = [
29     #host mi casa
30     #'192.168.1.5',
31     #host uesfmo
32     #'172.16.4.179',
33 ]
34
35
36 # Application definition
37
38 INSTALLED_APPS = [
39     'jet.dashboard',
40     'jet',
41     'django.contrib.admin',
42     'django.contrib.auth',
43     'django.contrib.contenttypes',
44     'django.contrib.sessions',
45     'django.contrib.messages',
46     'django.contrib.staticfiles',
47     'inventario',
48     'ventas',
49 ]
50 IMPORT_EXPORT_USE_TRANSACTIONS = True
51
```

## TEMPLATES

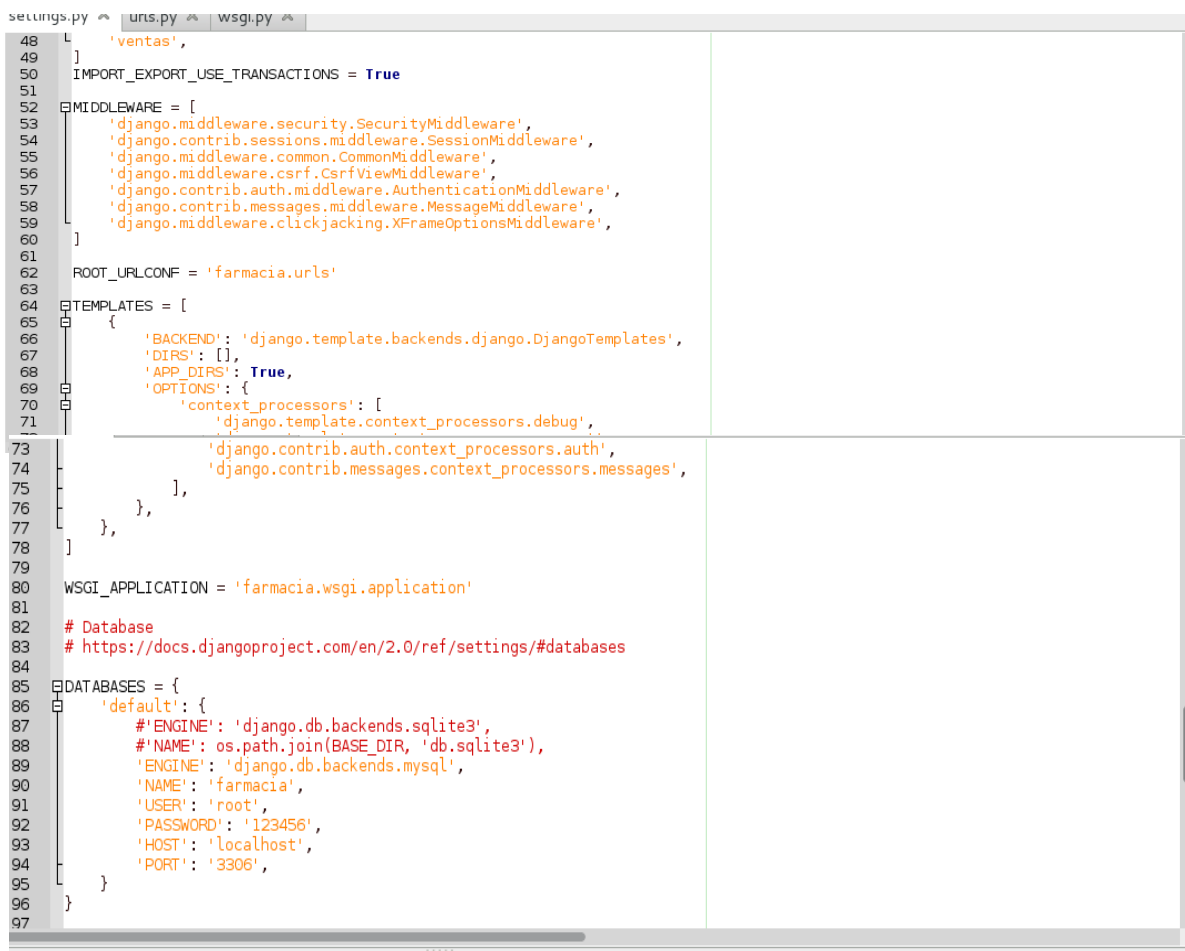
También podemos configurar los templates gracias a esta variable.

## DATABASES

Variable de configuración de Base de Datos, aquí podemos configurar cualquier motor de base de datos que Django utilice(MySQL, Postgres, Oracle, etc). En nuestro proyecto utilizamos Mysql.

## STATIC\_URL

Variable que utilizamos al momento de enlazar nuestros archivos estáticos en los templates.



```
48 ]
49
50 IMPORT_EXPORT_USE_TRANSACTIONS = True
51
52 MIDDLEWARE = [
53     'django.middleware.security.SecurityMiddleware',
54     'django.contrib.sessions.middleware.SessionMiddleware',
55     'django.middleware.common.CommonMiddleware',
56     'django.middleware.csrf.CsrfViewMiddleware',
57     'django.contrib.auth.middleware.AuthenticationMiddleware',
58     'django.contrib.messages.middleware.MessageMiddleware',
59     'django.middleware.clickjacking.XFrameOptionsMiddleware',
60 ]
61
62 ROOT_URLCONF = 'farmacia.urls'
63
64 TEMPLATES = [
65     {
66         'BACKEND': 'django.template.backends.django.DjangoTemplates',
67         'DIRS': [],
68         'APP_DIRS': True,
69         'OPTIONS': {
70             'context_processors': [
71                 'django.template.context_processors.debug',
72                 'django.contrib.auth.context_processors.auth',
73                 'django.contrib.messages.context_processors.messages',
74             ],
75         },
76     },
77 ]
78
79
80 WSGI_APPLICATION = 'farmacia.wsgi.application'
81
82 # Database
83 # https://docs.djangoproject.com/en/2.0/ref/settings/#databases
84
85 DATABASES = {
86     'default': {
87         # 'ENGINE': 'django.db.backends.sqlite3',
88         # 'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
89         'ENGINE': 'django.db.backends.mysql',
90         'NAME': 'farmacia',
91         'USER': 'root',
92         'PASSWORD': '123456',
93         'HOST': 'localhost',
94         'PORT': '3306',
95     }
96 }
```

Bien estas son algunas de las variables que Django trae por defecto, el hecho de todo esto que necesitamos diferentes archivos de configuración para trabajar dependiendo del ambiente de desarrollo donde nos encontremos.

## CONCLUSION

Podemos indicar que el sistema de inventario de la FARMACIA UES | FMO diseñado podrá registrar datos de los medicamentos ya sea de entrada (compra) o de salida (venta), también verificará el stock de los productos para la respectiva adquisición de medicamentos que hagan falta en la farmacia.

Con estos beneficios el sistema facilitará el trabajo al propietario y/o farmacéutico encargado al momento de realizar consultas o reportes, mejorando su nivel de profesionalidad al momento realizar la atención a los clientes ya que reducirá el tiempo al momento de realizar la búsqueda de medicamentos para ofrecerlos.

Ya que Los inventarios representan bienes destinados a las ventas en el curso normal de los negocios. Para mayor amplitud de las funciones y servicios de los inventarios depende de la naturaleza y el tipo de empresa, la importancia de los gastos de materiales y bienes de equipo y organización de la empresa.

La administración de inventario se centra en cuatro aspectos básicos; como los son: el número de unidades que deberán producirse en un momento determinado, en que momento debe producirse el inventario, que artículo merece atención especial, y podemos protegernos de los cambios en los costos de los artículos en inventario. De esta manera podemos señalar que la administración de inventario consiste en proporcionar los inventarios que se requieren para mantener la operación al costo más bajo posible.

El control de inventario se realiza con la finalidad de desarrollar pronósticos de ventas o presupuesto, para así determinar los costos de inventarios, compras u obtención, recepción, almacenaje, producción, embarque y contabilidad.

Los inventarios se clasifican de acuerdo a las características de la empresa, y una de las formas de clasificarlos es: Inventario de Materia Prima, Producción en Proceso, Productos Terminados, Materiales y Suministros.

De esta forma con este sistema elaborado con este framework llamado Django se le hará más factible al momento de llevar todos los registros requeridos para realizar un inventario en las empresas.

También lleva como finalidad que se pueda entender todos los procesos que se llevan a cabo al realizar dicho sistema, en referencia a su desarrollo y su aplicación.