

PYTHON

Adrián Jiménez Nieto

PROYECTO FINAL



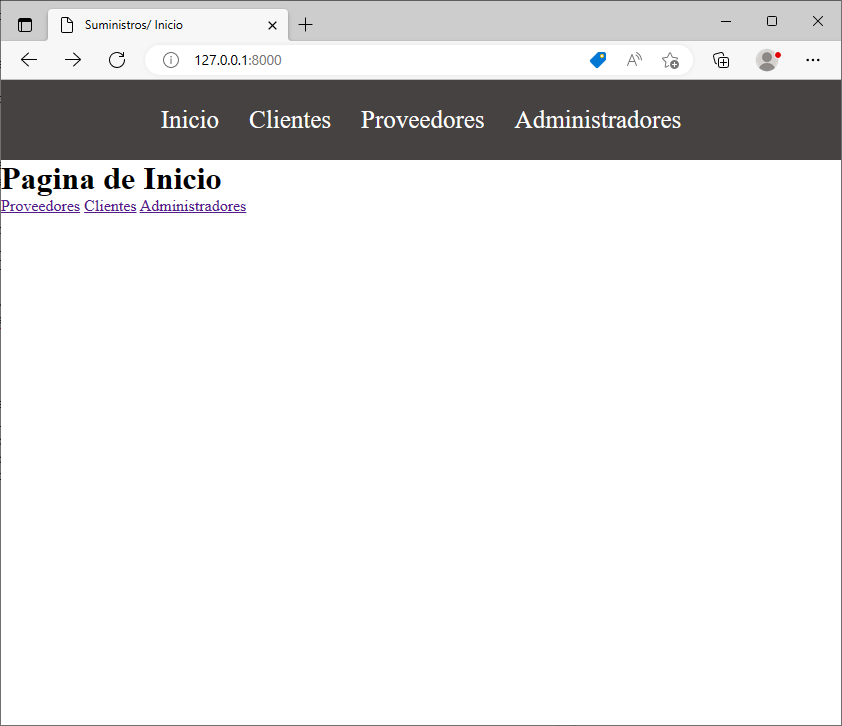
ADRIÁN JIMÉNEZ NIETO

RESPONSABLE DEL DESARROLLO DE LA APLICACIÓN

**Fecha:** 09/03/2023

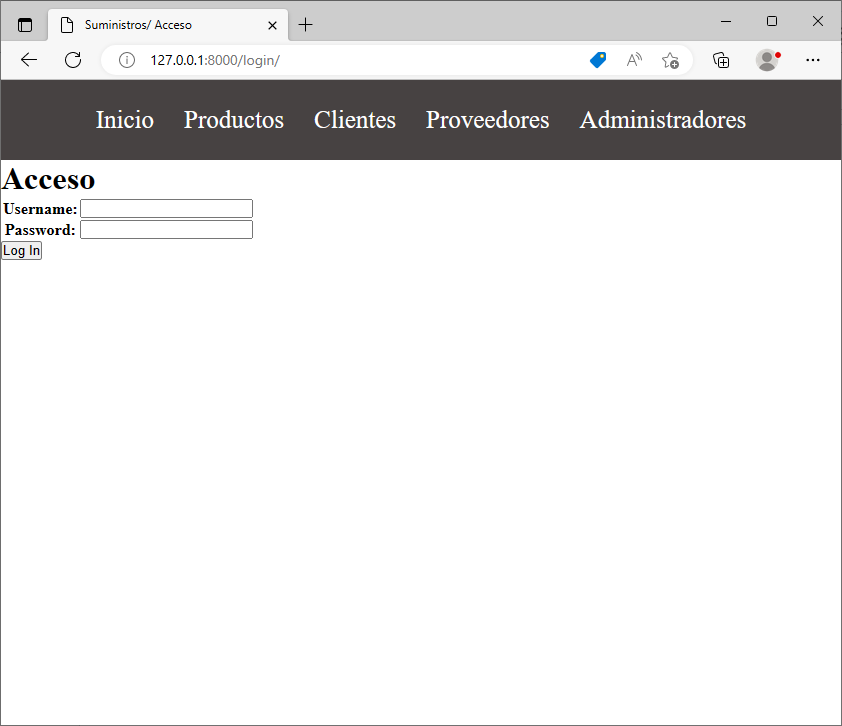
**DESARROLLO**

1. Se nos pide desarrollar una aplicación web con Python en la que haremos uso de algún framework. La aplicación consiste en un gestor para una empresa de suministros informáticos.
2. Para ello definiremos una serie de clases básicas que necesitaremos para la aplicación web. Una clase de Productos, una clase de Proveedores que relacionaremos con la anterior y una clase de Clientes.
3. Crearemos también varias App, una para gestionar los clientes, otra para gestionar los Proveedores, una para gestionar los administradores que llamaremos Admin, y otra para los Productos (para poder visualizarlos y demás).
4. Una vez generadas las app de nuestro proyecto, las instalaremos en nuestro ‘manage.py’. Después y poniendo el broche a la configuración general inicial definiremos las URLs de las app en el general(modularizando con unos ‘insert’), y las URLs de cada app encargadas de llevarnos a las vistas donde residirá nuestra lógica y renderizaremos los templates oportunos.
5. Antes de empezar a construir los modelos de cada app crearemos los directorios y archivos dedicados a la parte visual de nuestra aplicación. Crearemos los templates donde estarán las plantillas HTML que modularizaremos a través del método ‘extends’ para que quede un código mucho más limpio. Y crearemos un pequeño menú de navegación para tener algo visual con lo que empezar a trabajar. Dejo aquí abajo un primer vistazo de la aplicación.



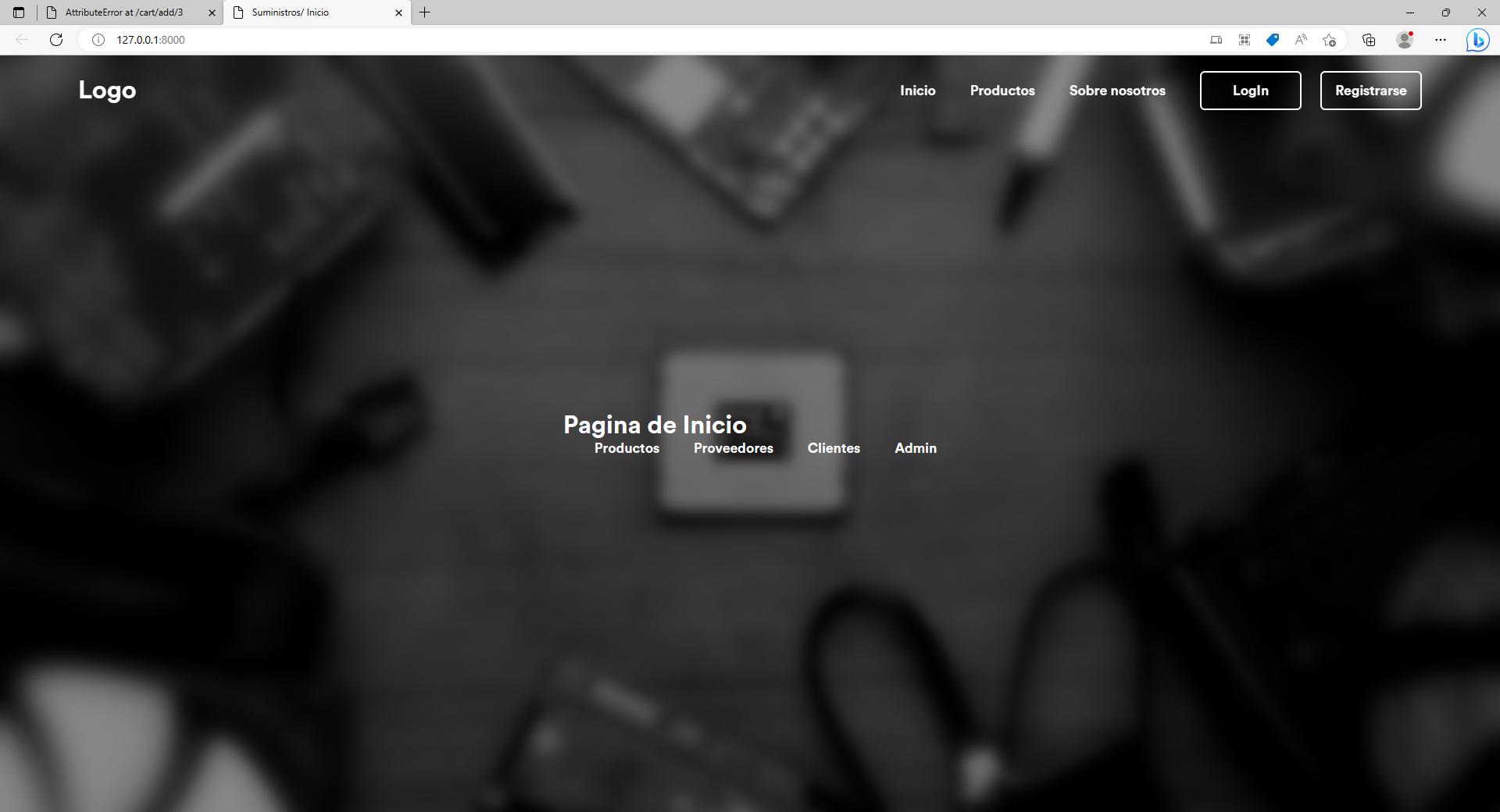
Primera vista de la App

1. Una vez todo listo es hora de preparar los modelos que van a tener cada app. En un principio desarrollaremos la aplicación web en SQLite3 pero más tarde podremos configurar la aplicación para que trabaje con postgreSQL para tener algo un poco más profesional y potente.
2. Una vez creados los modelos pasamos a crear los formularios que usaremos para registrar tanto clientes, proveedores, productos, y administradores (en el código lo llamamos Staff para no crear conflictos con la ruta ‘admin/’ propia de django de la que hablaremos más adelante).
3. Ya creados los formularios y antes de desarrollar los CRUD para los productos, clientes, proveedores y administradores. Hemos creado una pequeña app de login la cual tendrá un modelo con dos claves foráneas que relacionaran los modelos de clientes, proveedores y administradores con el modelo de login, el cual al iniciar sesión con su nombre de usuario y contraseña nos mandará al índice de la pagina, y según su nivel de acceso tendrán disponibles unas funciones u otras.



Primer prototipo Login 1

1. Tras crear esa primera aproximación a la pagina de Login crearemos dentro de nuestros ‘templates’, las plantillas correspondientes para el CRUD de los diferentes tipos de modelos en cada app. Para ello, primero vamos a hacer uso de un ‘Factory faker’ que, tras instalarlo, nos va a poblar nuestros modelos para empezar a hacer pruebas.
2. Cabe mencionar que la inserción de los formularios lo hacemos gracias a la herramienta de Django que nos permite simplemente insertar el formulario en la plantilla HTML con la instrucción ‘{{ form }}’.
3. Después de crear todos los templates de los correspondientes CURDs creamos un simple primer menú para cada uno de ellos.
4. Aprovechamos ahora para actualizar un poco nuestra página de inicio y darle un poco de estilo.



Página de inicio

1. Una vez dado el estilo a la pagina de inicio, vamos a crear una funcionalidad para poder añadir productos a nuestro carrito de la compra. Para ello crearemos una app con un modelo llamado ‘shoppingCart’ el cual poseerá dos claves foráneas, una para hacer referencia al usuario que está comprando y otra para añadir productos.

**CONCLUSIONES**