



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

Negocios Electrónicos y Desarrollo WEB



PROYECTO FINAL

Integrantes:

- Arellano Hernández Israel . 318251020
- Badillo Aguilar Diego. 318095909
- Camacho Martínez Juan Carlos . 318035530
- Constantino Cruz Pablo Giovanni . 318073538
- Juarez Paniagua Cristopher Israel . 318175890
- Salgado Valdés Andrés . 318146094

SEMESTRE 2025-1

FECHA DE ENTREGA: 2 de Diciembre de 2024

Introducción

En la actualidad, la tecnología juega un papel fundamental en nuestras vidas, facilitando tareas y conectándonos con el mundo, pero también introduciendo un desafío constante: las distracciones digitales. Redes sociales, notificaciones y plataformas de entretenimiento a menudo desvían nuestra atención, dificultando el cumplimiento de objetivos personales y profesionales. Ante esta problemática, presentamos Panopticon, una aplicación diseñada para transformar el uso de dispositivos electrónicos, ayudando a los usuarios a gestionar mejor su tiempo y mantener el enfoque en lo realmente importante.

Panopticon tiene como objetivo principal eliminar las distracciones digitales para permitir que los usuarios cumplan con sus actividades de forma más eficiente y productiva, ya sea en el ámbito académico, profesional o personal. A través de herramientas inteligentes y personalizables, esta aplicación ofrece una experiencia integral que combina gestión del tiempo, optimización del uso de dispositivos y un enfoque claro en los objetivos.

Descripción y características clave

Panopticon está diseñada como una solución avanzada que permite a los usuarios crear agendas personalizadas, especificando las actividades a realizar, las aplicaciones necesarias para cada tarea y las páginas web relevantes. Todo lo que no sea esencial para cumplir con estas actividades será bloqueado o restringido, permitiendo un enfoque total en las prioridades del usuario. Entre las características destacadas de Panopticon se encuentran:

- **Gestión eficiente del tiempo y las tareas**
 - Los usuarios pueden crear agendas en las que delimitan las actividades y los recursos tecnológicos necesarios para llevarlas a cabo. Esto incluye definir qué aplicaciones y páginas web estarán disponibles y cuáles se bloquearán durante periodos específicos.
- **Sincronización inteligente con herramientas de trabajo y estudio**
 - La aplicación puede integrarse con plataformas como Google Classroom y Microsoft Teams, facilitando la organización y ejecución de tareas. Por ejemplo, al acercarse la fecha de entrega de un proyecto, se activará la función:
 - **“DEADLINE”**

- Esta herramienta bloquea todas las aplicaciones y páginas web que no sean necesarias para la actividad en curso, asegurando que el usuario pueda concentrarse completamente en el cumplimiento de su tarea. La función se desactiva automáticamente al finalizar el tiempo asignado o completar el objetivo, y en caso de ser necesario, permitirá extender el tiempo bajo ciertas condiciones.
- **Función “LADRILLO” para dispositivos móviles**
 - Diseñada para momentos en los que el usuario necesita máxima concentración, esta funcionalidad convierte el dispositivo móvil en una herramienta básica, limitando su uso a llamadas, mensajería y aplicaciones esenciales definidas previamente por el usuario. “LADRILLO” se puede activar junto con “DEADLINE” o de manera independiente, y está pensado para reducir la dependencia del celular durante actividades importantes.
- **Restricción personalizada de navegadores web**
 - Para aquellos momentos en los que el uso de internet es necesario, Panopticon permite acceder únicamente a sitios web relevantes para la tarea específica. Por ejemplo, un estudiante trabajando en una tarea de bases de datos podrá usar el navegador para investigar contenido relacionado, pero no podrá acceder a redes sociales, plataformas de streaming o sitios de entretenimiento.
- **Flexibilidad total**
 - Panopticon se adapta completamente a las necesidades del usuario, permitiendo personalizar el nivel de restricciones. Si el usuario considera adecuado utilizar plataformas como YouTube o Spotify durante ciertas actividades, la aplicación lo permitirá. De igual manera, puede configurarse para restringir el dispositivo exclusivamente a herramientas de trabajo como Microsoft Office, Jira, GitHub, Notion, Obsidian y otras plataformas productivas.

Objetivos del proyecto

Panopticon tiene como objetivo principal resolver el problema de las distracciones digitales, brindando una herramienta práctica, adaptable y eficiente para que los usuarios puedan maximizar su desempeño en cualquier ámbito. Entre los objetivos específicos destacan:

1. Proveer una plataforma que permita al usuario gestionar de manera efectiva su tiempo y enfoque.
2. Reducir la dependencia de aplicaciones y páginas web irrelevantes durante momentos de trabajo o estudio.
3. Ofrecer una solución intuitiva, accesible y fácil de personalizar para adaptarse a las diferentes necesidades de los usuarios.
4. Integrar herramientas avanzadas como “DEADLINE” y “LADRILLO” para garantizar que las distracciones se reduzcan al mínimo.
5. Promover un equilibrio saludable entre el uso de dispositivos electrónicos y el cumplimiento de metas personales y profesionales.

Requisitos

- **Funcionales**

- **Gestión de tareas y tiempo**

- Se le permite al usuario poder crear una agenda personalizada con tareas específicas, asignando las aplicaciones y páginas web necesarias y bloqueando las que no para la tarea.

- **DEADLINE**

- Se bloquean automáticamente las aplicaciones y sitios web no relacionados con la tarea en curso, activándose cerca de la fecha de entrega y desactivando cuando se complete la tarea o al finalizar el tiempo asignado. Se permite extensión de tiempo si es necesario.

- **LADRILLO**

- Limitar al dispositivo móvil a funciones básicas (llamadas, mensajería, aplicaciones esenciales), activable de forma independiente o junto con “DEADLINE”.

- **Sincronización con plataformas de trabajo**

- Integración con herramientas como Google Classroom y Microsoft Teams para facilitar la organización de tareas y proyectos

- **Funcionalidad de compras dentro de la aplicación**

- Permitir a los usuarios comprar características premium o suscripciones para desbloquear funcionalidades adicionales como nuevas herramientas de personalización o un mayor control sobre las restricciones.

- ***Métodos de pago***
 - Ofrecer métodos de pago electrónicos para las compras dentro de la aplicación, como tarjetas de crédito/débito, PayPal, y transferencias bancarias en línea, pero no permitir pagos en efectivo.
- ***No Funcionales***
 - ***Rendimiento y Escalabilidad:***
 - Debe funcionar de manera eficiente, con tiempos de respuesta rápidos y capaz de manejar un aumento de usuarios o tareas sin comprometer el rendimiento.
 - ***Usabilidad***
 - Se le permite al usuario poder crear una agenda personalizada con tareas específicas, asignando las aplicaciones y páginas web necesarias y bloqueando las que no para la tarea.
 - ***Compatibilidad y accesibilidad***
 - Debe ser compatible con diferentes sistemas operativos (Windows, macOS, Android, iOS) y navegadores principales, además de ser accesible para personas con discapacidades.
 - ***Seguridad y Privacidad***
 - La aplicación debe proteger los datos del usuario, especialmente información sobre tareas y configuraciones, garantizando el acceso seguro a funciones sensibles mediante autenticación.
 - ***Mantenibilidad***
 - El código es modular y fácil de actualizar, permitiendo agregar nuevas funciones o realizar mejoras sin afectar el funcionamiento actual.

Arquitectura del Sistema

La arquitectura de Panopticon está diseñada con un enfoque modular y escalable, utilizando tecnologías modernas y patrones de diseño que facilitan tanto su implementación como su mantenimiento a largo plazo. Se describe detalladamente la estructura de la aplicación y sus componentes clave a continuación:

1. Arquitectura General:

El sistema sigue una arquitectura **Cliente-Servidor**, donde la **interfaz de usuario** (frontend) interactúa con el **backend** a través de una serie de **APIs RESTful**. La aplicación se basa en el framework **Django**, que facilita la creación de aplicaciones web robustas y seguras.

- **Frontend:** La interfaz de usuario se construye utilizando tecnologías estándar web (HTML, CSS, JavaScript), gestionada a través del framework Django, que proporciona tanto las vistas como los controladores necesarios para la interacción con el usuario.
- **Backend:** El servidor de la aplicación, basado en **Django**, gestiona la lógica de negocio, las peticiones HTTP, la interacción con la base de datos y las integraciones con servicios externos (como APIs de Twitter y Facebook).
- **Base de Datos:** Se utiliza un sistema de gestión de bases de datos relacional que almacena información sobre los usuarios, tareas, configuraciones personalizadas y los datos de las interacciones de la aplicación.
- **APIs Externas:** El sistema interactúa con diversas APIs externas para funciones como la obtención de tweets de Twitter, la publicación de contenido en Facebook y la gestión de pagos a través de Stripe.

2. Descripción de los Componentes del Sistema:

-Frontend (Interfaz de Usuario):

- **Tecnologías:** HTML, CSS, JavaScript, Django Templates.
- **Funcionalidad:** Proporciona una interfaz limpia y fácil de usar para que los usuarios puedan interactuar con las funcionalidades de la aplicación, como la gestión de tareas, el bloqueo de aplicaciones, y la programación de publicaciones.

- **Comunicación con Backend:** Las peticiones de la interfaz (p. ej., iniciar sesión, enviar formularios, obtener datos) son manejadas por vistas de Django y se comunican con el servidor a través de solicitudes HTTP.

-Backend (Servidor y Lógica de Negocio):

- **Tecnologías:** Django (Python), Django REST Framework (para APIs RESTful).
- **Funcionalidad:** El backend maneja todas las operaciones de la lógica de negocio, como la autenticación de usuarios, la creación y gestión de tareas, la restricción de aplicaciones, la integración con plataformas externas (Google Classroom, Microsoft Teams, Twitter, Facebook) y el procesamiento de pagos.
- **Base de Datos:** La base de datos contiene modelos para usuarios, tareas, publicaciones, configuraciones de la aplicación, etc., gestionados a través del ORM de Django.
- **APIs Externas:** El backend se comunica con APIs externas para obtener tweets, gestionar publicaciones en Facebook, y realizar pagos.

- Base de Datos:

- **Funcionalidad:** Almacena toda la información persistente del sistema, incluyendo los datos de usuario, tareas, configuraciones de restricción, publicaciones realizadas y datos transaccionales de Stripe.
- **Estructura:** La base de datos está diseñada con tablas relacionadas que facilitan el acceso rápido y eficiente a los datos. Por ejemplo, se pueden tener tablas para los usuarios, productos, configuraciones de restricción, etc.

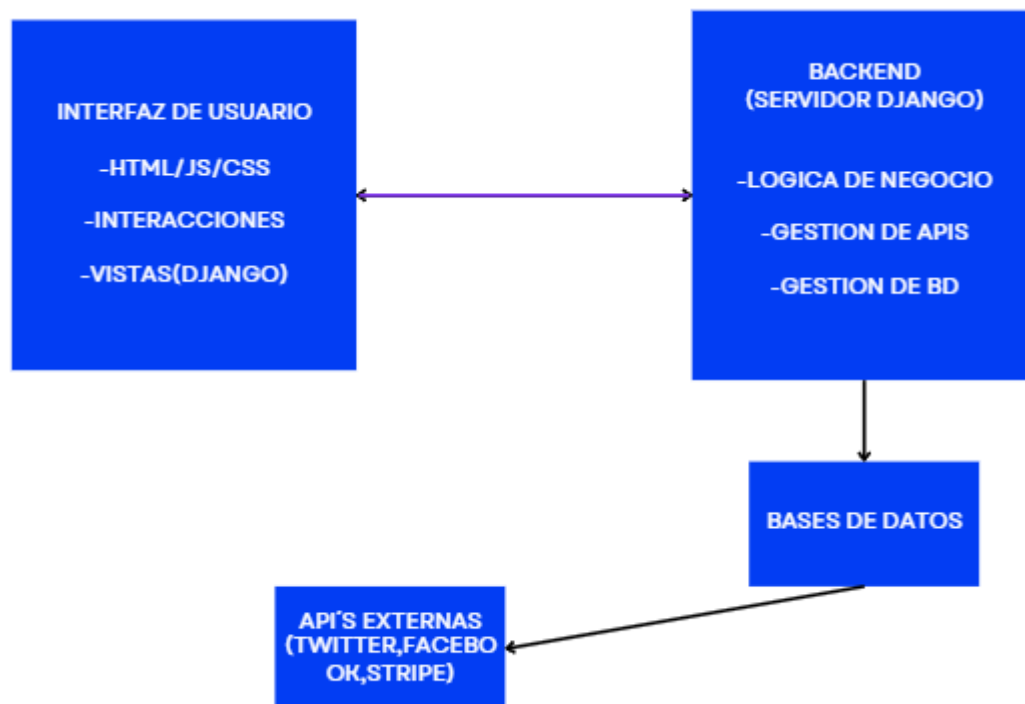
-Integración de APIs Externas:

- **Tweepy API:** Para interactuar con Twitter y obtener tweets o publicar mensajes.
- **Facebook API:** Para gestionar publicaciones en páginas de Facebook.
- **Stripe API:** Para gestionar pagos y suscripciones dentro de la aplicación.
- **Google Classroom / Microsoft Teams:** Para integrar las tareas y la sincronización con plataformas educativas y profesionales.

-Seguridad:

- Autenticación y Autorización: El sistema utiliza JWT (JSON Web Tokens) para gestionar la autenticación de usuarios, asegurando que las solicitudes a recursos protegidos solo sean realizadas por usuarios autenticados.
- Protección de Datos: Se emplean técnicas estándar de cifrado y protección de datos para garantizar la seguridad de la información sensible de los usuarios.

3. Diagrama conceptual:



Descripción del diagrama:

-Interfaz de Usuario:

- Esta es la parte visible de la aplicación, donde los usuarios interactúan con el sistema. Se desarrolla utilizando HTML, CSS y JavaScript, y Django Templates para generar el contenido dinámico. La UI es responsable de mostrar las tareas, las restricciones de aplicaciones, las publicaciones programadas, etc.
- Interacciones: Los usuarios realizan acciones como iniciar sesión, crear tareas o gestionar restricciones. Estas acciones se envían al backend a través de peticiones HTTP.

-Backend (Servidor Django):

- El backend es el núcleo de la aplicación, gestionado por el servidor Django. El backend se encarga de procesar todas las solicitudes de los usuarios, como la creación de tareas, la gestión de restricciones, la autenticación de usuarios y la integración con las APIs externas.
- Lógica de Negocio: El backend maneja las reglas del negocio, como la validación de datos y la coordinación de las diferentes funciones del sistema.
- Gestión de APIs: El servidor Django también es responsable de gestionar las interacciones con APIs externas, como Twitter, Facebook y Stripe.
- Gestión de Base de Datos (BD): Django se encarga de interactuar con la base de datos, donde se almacenan los datos del usuario, las tareas, las configuraciones y las publicaciones. Utiliza ORM (Object-Relational Mapping) para simplificar las consultas y la manipulación de datos.

-Base de Datos:

- El sistema de gestión de bases de datos almacena toda la información persistente de la aplicación. Esto incluye datos como las tareas de los usuarios, las configuraciones de las agendas, las publicaciones programadas, los datos de pago y las interacciones de las APIs.
- El backend consulta y actualiza la base de datos para garantizar que los usuarios vean datos actuales y que las interacciones se registren correctamente.

-APIs Externas:

APIs externas como Twitter, Facebook y Stripe se conectan con el backend para proporcionar funcionalidades adicionales:

- Twitter (Tweepy API): Para obtener tweets de Twitter o para publicar contenido en la plataforma.
- Facebook API: Para interactuar con publicaciones en Facebook, incluyendo la programación y la eliminación de publicaciones.
- Stripe API: Para gestionar pagos y suscripciones dentro de la aplicación.

4. Flujo de Comunicación:

-Usuario accede a la aplicación web: El usuario interactúa con la interfaz de usuario a través de un navegador web, donde puede gestionar tareas, bloquear aplicaciones o gestionar sus publicaciones.

-Backend maneja las solicitudes: Las peticiones del usuario son enviadas al servidor backend (Django), que procesa la lógica de negocio y consulta o actualiza la base de datos.

-Interacción con la Base de Datos: Cuando es necesario acceder o modificar datos persistentes, el backend consulta la base de datos para recuperar o actualizar la información relevante. Esto puede incluir la creación de una nueva tarea, la actualización de la agenda o el almacenamiento de datos de pago.

-Interacción con APIs externas: Si es necesario, el servidor backend interactúa con APIs externas, como Twitter, Facebook, o Stripe, para obtener información adicional o realizar acciones (como publicar un tweet o procesar un pago).

-Respuesta al usuario: El servidor devuelve los datos procesados a la interfaz de usuario, quien los presenta de manera adecuada al usuario.

Guia de instalación y configuración

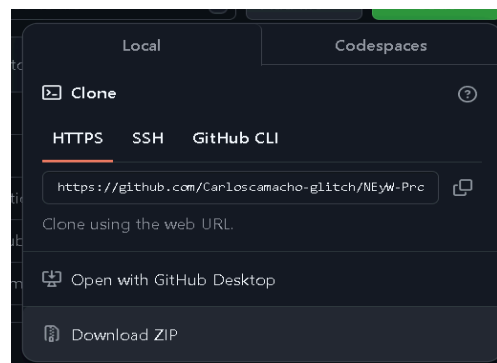
Para iniciar con la instalación de nuestra aplicación web, lo primero que debemos hacer es dirigirnos a él github en donde hemos estado trabajando esta misma:

Enlace GitHub Aplicación Web:

<https://github.com/Carloscamacho-glitch/NEyW-Proyecto-Final>

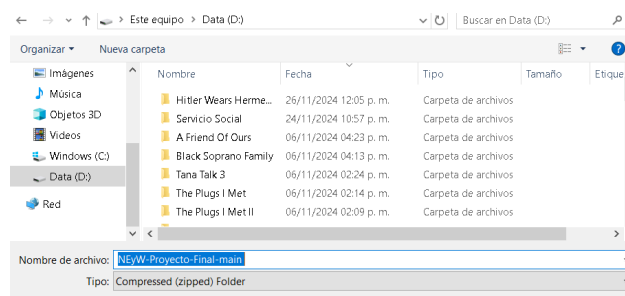
Una vez en dicho directorio, vamos a la pestaña de Code y nos dirigimos a la opción de Download Zip.

Github Aplicación Web:



Elegimos el directorio en el que queremos que se nos descargue la aplicación web. Una vez terminada la descarga y descomprimos el folder zip.

Descarga Aplicación Web desde GitHub:



Es importante tener en cuenta el directorio en donde colocamos el proyecto ya que para el correcto funcionamiento de nuestra aplicación web y que las librerías necesarias para el correcto funcionamiento de esto no afecte el funcionamiento de otros proyectos que estemos trabajando en nuestra PC, tendremos que trabajarlo dentro de un entorno virtual, creado en el directorio antes mencionado.

Para ello abrimos el cmd y nos dirigimos a el directorio en donde hayamos descargado la aplicación Web. En mi caso es la carpeta generada al copiar el proyecto de GitHub:

Directorio de la Aplicación Web:

```
D:\NEyW-Proyecto-Final>dir
El volumen de la unidad D es Data
El número de serie del volumen es: 38AE-C72E

Directorio de D:\NEyW-Proyecto-Final

26/11/2024  06:57 p. m.    <DIR>        .
26/11/2024  06:57 p. m.    <DIR>        ..
26/11/2024  06:57 p. m.                67 .gitignore
26/11/2024  06:57 p. m.    <DIR>        .vs
26/11/2024  06:57 p. m.    <DIR>        Papnoticon
26/11/2024  06:57 p. m.    <DIR>        Proyecto
26/11/2024  06:57 p. m.                317 README.md
                2 archivos          384 bytes
                5 dirs  38,931,456,000 bytes libres

D:\NEyW-Proyecto-Final>
```

Una vez allí verificamos la versión del Pip (Python Package Installer) el cual es el gestor de paquetes oficial para el lenguaje de programación Python. La funcionalidad de este es el facilitar la instalación, actualización y gestión de bibliotecas y dependencias que se requieren para el desarrollo con Python.

Comando para verificar si pip está instalado: pip --version

```
D:\NEyW-Proyecto-Final>pip --version
pip 20.2.3 from c:\users\cris\appdata\local\programs\python\python39\lib\site-packages\pip (python 3.9)
```

Después instalamos Virtualenv para poder crear entornos virtuales a través del gestor de paquetes pip. Virtualenv es una herramienta que permite crear entornos virtuales en Python. Un entorno virtual es un espacio aislado en el que puedes instalar bibliotecas y dependencias de Python sin interferir con el sistema Python global o con otros proyectos.

Comando para instalar virtualenv a través de pip: pip install virtualenv

```
D:\NEyW-Proyecto-Final>pip install virtualenv
Requirement already satisfied: virtualenv in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (20.27.1)
Requirement already satisfied: platformdirs<5,>=3.9.1 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from virtualenv) (4.3.6)
Requirement already satisfied: filelock<4,>=3.12.2 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from virtualenv) (3.16.1)
Requirement already satisfied: distlib<1,>=0.3.7 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from virtualenv) (0.3.9)
```

Para el siguiente caso podemos crear un nuevo entorno virtual mediante el comando *python -m venv (nombre que deseemos para el entorno virtual)*. O utilizar el entorno virtual previamente creado “Proyecto” este directorio contiene una instalación aislada de Python y sus bibliotecas. Además de que dentro de este nuevo directorio se copiarán los archivos del intérprete de Python y los scripts necesarios para gestionar el entorno virtual necesarios para ejecutar nuestra Aplicación Web.

Una vez creado el entorno virtual procedemos a activarlo mediante: *Proyecto\Scripts\activate*. Dicha instrucción nos ayudará a que cualquier paquete que instalemos o script que ejecutemos para esta tarea se realice dentro del entorno virtual, en lugar de la instalación global de Python.

Comando para activar entorno virtual: *Proyecto\Scripts\activate*

```
D:\NEyW-Proyecto-Final>Proyecto\Scripts\activate
(Proyecto) D:\NEyW-Proyecto-Final>
```

Verás que una vez ejecutado el comando, el nombre de nuestro entorno virtual appWeb aparece al inicio de la línea de comandos, indicando que estamos trabajando dentro de ese entorno virtual. Posteriormente a que hemos creado y activado el entorno virtual appWeb podemos instalar dentro de nuestro entorno virtual las librerías necesarias para el correcto funcionamiento de nuestra aplicación web: **Django, Pillow, BeautifulSoup4, Request, Tweepy, Stripe, Selenium y webdriver-manager**.

Django es un framework de desarrollo web que automatiza y simplifica tareas del desarrollo web. Dicha instalación se hace mediante pip al igual que el resto de librerías .

Comando para instalar django a través de pip: *pip install django*

```
(Proyecto) D:\NEyW-Proyecto-Final\Proyecto>pip install django
Collecting django
  Using cached Django-4.2.16-py3-none-any.whl (8.0 MB)
Collecting sqlparse>=0.3.1
  Using cached sqlparse-0.5.2-py3-none-any.whl (44 kB)
Collecting tzdata; sys_platform == "win32"
  Using cached tzdata-2024.2-py2.py3-none-any.whl (346 kB)
Collecting asgiref<4,>=3.6.0
  Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Collecting typing-extensions>=4; python_version < "3.11"
  Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: sqlparse, tzdata, typing-extensions, asgiref, django
Successfully installed asgiref-3.8.1 django-4.2.16 sqlparse-0.5.2 typing-extensions-4.12.2 tzdata-2024.2
```

Django-cors-headers es utilizado en proyecto Django para manejar CORS (Cross-Origin Resource Sharing).

Comando para instalar cors-headers: *pip install django-cors-headers*

```
(Proyecto) C:\Users\israe\OneDrive\Documentos\NEyDW\NEyW-Proyecto-Final-main\NEyW-Proyecto-Final-main\Papnoticon>pip install django-cors-headers
Collecting django-cors-headers
  Downloading django_cors_headers-4.6.0-py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: asgiref>=3.6 in c:\users\israe\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from django-cors-headers) (3.8.1)
Requirement already satisfied: django>=4.2 in c:\users\israe\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from django-cors-headers) (5.1.3)
Requirement already satisfied: typing-extensions>=4 in c:\users\israe\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from asgiref>=3.6->django-cors-headers) (4.12.2)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\israe\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from django>=4.2->django-cors-headers) (0.5.2)
Requirement already satisfied: tzdata in c:\users\israe\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from django>=4.2->django-cors-headers) (2024.2)
Downloading django_cors_headers-4.6.0-py3-none-any.whl (12 kB)
Installing collected packages: django-cors-headers
Successfully installed django-cors-headers-4.6.0
```

Pillow es una biblioteca de Python que permite trabajar con imágenes cuenta con funciones como: abrir, redimensionar, recortar, rotar, transformar imágenes, aplicar filtros y efectos etc..

Comando para instalar pillow a través de pip: *pip install pillow*

```
(Proyecto) D:\NEyW-Proyecto-Final\Proyecto>pip install pillow
Collecting pillow
  Using cached pillow-11.0.0-cp39-cp39-win_amd64.whl (2.6 MB)
Installing collected packages: pillow
Successfully installed pillow-11.0.0
```

beautifulsoup4, es una librería que ayuda a analizar y manipular documentos HTML y XML. Su propósito es facilitar la extracción de datos de una página web: títulos, tablas, imágenes o cualquier elemento dentro de un archivo HTML. Proporciona métodos para navegar por las etiquetas y contenidos de una página.

Comando para instalar beautifulsoup4 a través de pip: *pip install beautifulsoup4*

```
(Proyecto) D:\NEyW-Proyecto-Final\Proyecto>pip install beautifulsoup4
Requirement already satisfied: beautifulsoup4 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (4.12.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from beautifulsoup4) (2.6)
```

Requests, es una librería que sirve para hacer solicitudes HTTP, esta permite conectarte a una página web y descargar su contenido HTML. Además de que soporta los métodos HTTP GET y POST.

Comando para instalar requests a través de pip: *pip install requests*

```
(Proyecto) D:\NEyW-Proyecto-Final\Proyecto>pip install requests
Requirement already satisfied: requests in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (2.32.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests) (2024.8.30)
Requirement already satisfied: idna<4,>=2.5 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests) (2.2.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests) (3.4.0)
```

Tweepy es una biblioteca que actúa como una interfaz para interactuar con la Twitter API. Realiza funciones como: publicar tweets automáticamente, leer el timeline de tu cuenta, buscar tweets que contengan palabras claves, gestionar seguidores, etc.

Comando para instalar Tweepy a través de pip: *pip install Tweepy*

```
(Proyecto) D:\NEyW-Proyecto-Final\Proyecto>pip install Tweepy
Collecting Tweepy
  Downloading tweepy-4.14.0-py3-none-any.whl (98 kB)
    |#####| 98 kB 1.2 MB/s
Collecting requests-oauthlib<2,>=1.2.0
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Collecting oauthlib<4,>=3.2.0
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
    |#####| 151 kB 3.3 MB/s
Requirement already satisfied: requests<3,>=2.27.0 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from Tweepy) (2.32.3)
Requirement already satisfied: idna<4,>=2.5 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests<3,>=2.27.0->Tweepy) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests<3,>=2.27.0->Tweepy) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests<3,>=2.27.0->Tweepy) (2024.8.30)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests<3,>=2.27.0->Tweepy) (3.4.0)
Installing collected packages: oauthlib, requests-oauthlib, Tweepy
Successfully installed Tweepy-4.14.0 oauthlib-3.2.2 requests-oauthlib-1.3.1
```

Stripe es una plataforma que facilita la recepción de pagos en línea. Ofrece herramientas para gestionar suscripciones, emitir reembolsos, prevenir fraudes y analizar datos financieros de transacciones en línea.

Comando para instalar Stripe a través de pip: *pip install Stripe*

```
(Proyecto) D:\NEyW-Proyecto-Final\Proyecto>pip install Stripe
Collecting Stripe
  Downloading stripe-11.3.0-py2.py3-none-any.whl (1.6 MB)
    |#####| 1.6 MB 1.7 MB/s
Requirement already satisfied: requests>=2.20; python_version >= "3.0" in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from Stripe) (2.32.3)
Requirement already satisfied: typing-extensions>=4.5.0; python_version >= "3.7" in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from Stripe) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests>=2.20; python_version >= "3.0"->Stripe) (3.4.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests>=2.20; python_version >= "3.0"->Stripe) (2024.8.30)
Requirement already satisfied: idna<4,>=2.5 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests>=2.20; python_version >= "3.0"->Stripe) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests>=2.20; python_version >= "3.0"->Stripe) (2.2.3)
Installing collected packages: Stripe
Successfully installed Stripe-11.3.0
```


Selenium es una herramienta utilizada para automatizar navegadores web. Permite interactuar con páginas web de la misma manera que lo haría un usuario, como hacer clic en botones, llenar formularios, navegar entre páginas, tomar capturas de pantalla, entre otros.

Comando para instalar Selenium a través de pip: *pip install selenium*

```
(Proyecto) D:\NEyW-Proyecto-Final\Papnoticon>pip install selenium
Collecting selenium
  Downloading selenium-4.27.1-py3-none-any.whl (9.7 MB)
    |████████████████████| 9.7 MB 1.7 MB/s
Requirement already satisfied: urllib3[socks]<3,>=1.26 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from selenium) (2.2.3)
Collecting trio-websocket<=0.9
  Downloading trio_websocket-0.11.1-py3-none-any.whl (17 kB)
Requirement already satisfied: typing_extensions<=4.9 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from selenium) (4.12.2)
Collecting trio<=0.17
  Downloading trio-0.27.0-py3-none-any.whl (481 kB)
    |████████████████████| 481 kB ...
Requirement already satisfied: certifi>=2021.10.8 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from selenium) (2024.8.30)
Collecting websocket-client<=1.8
  Downloading websocket_client-1.8.0-py3-none-any.whl (58 kB)
    |████████████████████| 58 kB ...
Collecting pysocks!=1.5.7,<2.0,>=1.5.6; extra == "socks"
  Downloading PySocks-1.7.1-py3-none-any.whl (16 kB)
```

WebDriver es una herramienta que forma parte del Selenium y se utiliza para controlar y automatizar navegadores web de forma programática.

Comando para instalar WebDriver a través de pip: *pip install webdriver-manager*

```
(Proyecto) D:\NEyW-Proyecto-Final\Papnoticon>pip install webdriver-manager
Collecting webdriver-manager
  Downloading webdriver_manager-4.0.2-py2.py3-none-any.whl (27 kB)
Collecting python-dotenv
  Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Requirement already satisfied: requests in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from webdriver-manager) (2.32.3)
Collecting packaging
  Downloading packaging-24.2-py3-none-any.whl (65 kB)
    |████████████████████| 65 kB 1.5 MB/s
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests->webdriver-manager) (3.4.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests->webdriver-manager) (2.2.3)
Requirement already satisfied: idna<4,>=2.5 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests->webdriver-manager) (3.10)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\cris\appdata\local\programs\python\python39\lib\site-packages (from requests->webdriver-manager) (2024.8.30)
Installing collected packages: python-dotenv, packaging, webdriver-manager
Successfully installed packaging-24.2 python-dotenv-1.0.1 webdriver-manager-4.0.2
```

Una vez que con todas las librerías necesarias nos dirigimos a el directorio `\NEyW-Proyecto-Final\Papnoticon` en donde se encuentra el código principal `manage.py`.

Para comprobar que nuestra instalación fue correcta colocamos el comando `python manage.py runserver`. El cual mediante el framework Django iniciar un servidor de desarrollo local. El cual nos permitirá probar y visualizar la aplicación web panopticon.

Comando para iniciar servidor local : `python manage.py runserver`

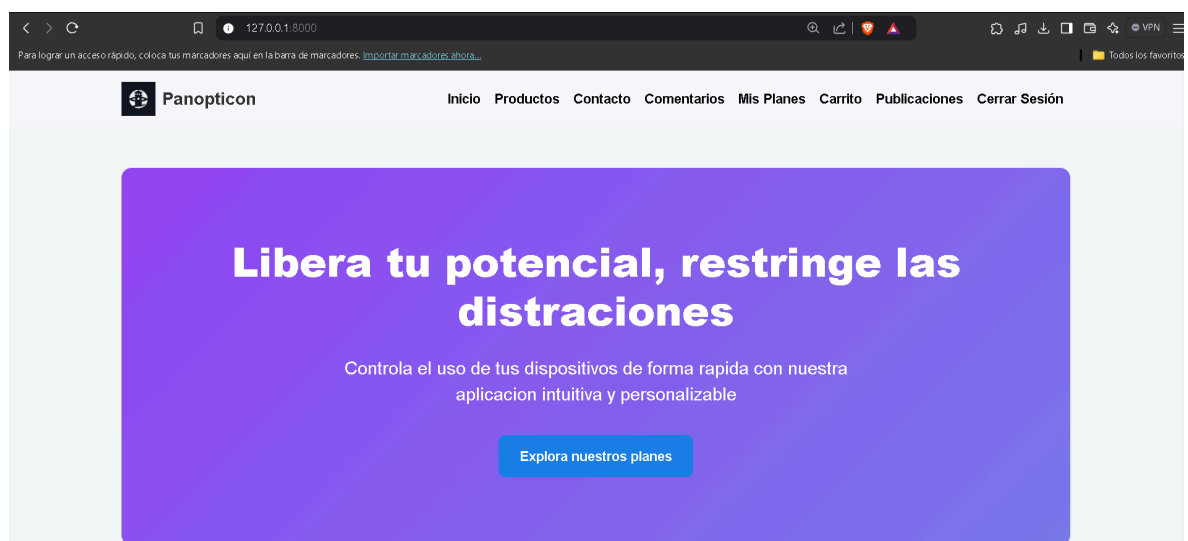
```
(Proyecto) D:\NEyW-Proyecto-Final\Papnoticon>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 26, 2024 - 22:38:48
Django version 4.2.16, using settings 'Papnoticon.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[26/Nov/2024 22:38:57] "GET / HTTP/1.1" 200 6409
[26/Nov/2024 22:38:58] "GET /static/Marca/Logo.png HTTP/1.1" 200 40245
Not Found: /favicon.ico
```

Para poder visualizar nuestra aplicación Web entramos a la url que nos despliega la consola, en cualquier navegador web:

Url Aplicacion Web Panopticon: `http://127.0.0.1:8000/`



Dado que esta aplicación está en su modo beta, la base de datos no es estática, por lo que es necesario seguir una serie de pasos para utilizarla correctamente:

1. **Eliminar la configuración actual:** Primero, se debe borrar el archivo **del db.sqlite3** para reiniciar la base de datos y eliminar cualquier configuración previa.
2. **Generar migraciones:** Luego, se ejecutará ***python manage.py makemigrations*** que creará los archivos de migración necesarios para reflejar la estructura definida en los modelos.
3. **Aplicar las migraciones:** A continuación, ***python manage.py migrate*** se encargará de aplicar estos cambios a la base de datos, creando las tablas y configuraciones necesarias.
4. **Crear un superusuario:** Finalmente, se debe generar un superusuario ***python manage.py createsuperuser***, lo que permitirá acceder al panel de administración para configurar y personalizar la aplicación según se desee.

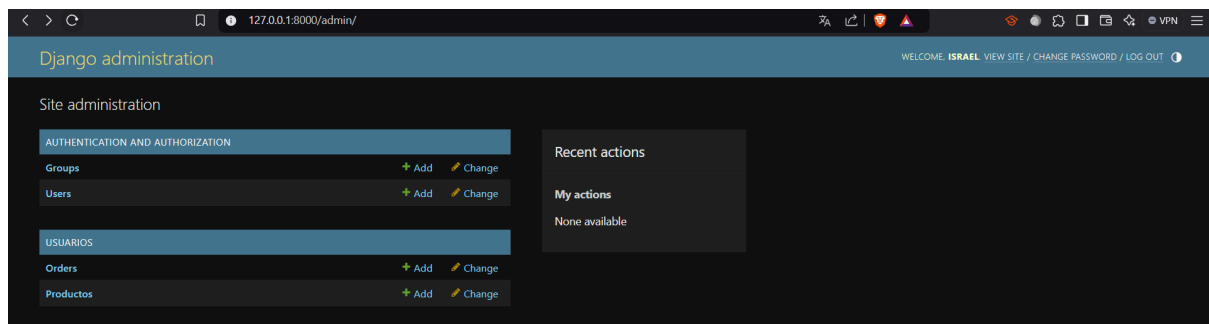
Comandos para configurar la base de datos:

- *del db.sqlite3*
- *python manage.py makemigrations*
- *python manage.py migrate*
- *python manage.py createsuperuser*

```
(Proyecto) C:\Users\israel\OneDrive\Documentos\NEyDW\NEyW-Proyecto-Final-main\NEyW-Proyecto-Final-main\Papnoticon>del db.sqlite3
(Proyecto) C:\Users\israel\OneDrive\Documentos\NEyDW\NEyW-Proyecto-Final-main\NEyW-Proyecto-Final-main\Papnoticon>python manage.py makemigrations
No changes detected
(Proyecto) C:\Users\israel\OneDrive\Documentos\NEyDW\NEyW-Proyecto-Final-main\NEyW-Proyecto-Final-main\Papnoticon>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, usuarios
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
  Applying usuarios.0001_initial... OK
  Applying usuarios.0002_cartitem_producto_stock_alter_producto_imagen_order_and_more... OK

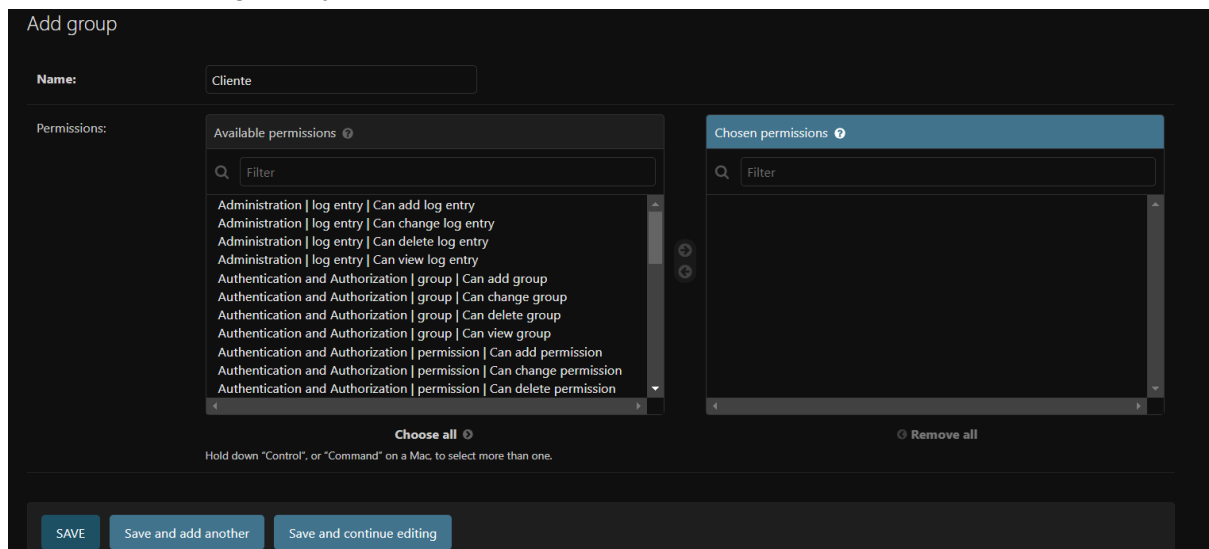
(Proyecto) C:\Users\israel\OneDrive\Documentos\NEyDW\NEyW-Proyecto-Final-main\NEyW-Proyecto-Final-main\Papnoticon>python manage.py createsuperuser
Username (leave blank to use 'israel'): israel
Email address: israelsocum@hotmail.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Para configurar los roles de las personas, observar los usuarios registrados, los productos, entre otras cosas necesitamos poner la siguiente direccion ip: <http://127.0.0.1:8000/admin>



Las configuraciones iniciales que se deben de realizar son las siguientes:

1. **Creación del rol Cliente:** Necesitamos crear este rol para que los usuarios se puedan registrar y autenticar.



2. **Agregar Productos:** Es necesario agregar un producto nuevo para poder observar todas las funcionalidades iniciales de la aplicación.

APIs:

Panopticon

Nuestro proyecto cuenta con la API PANOPTICON, la cual es desglosamos a continuación mediante la siguiente descripción realizada en swagger:

PapnoticonAPI.json

```
openapi: 3.0.0
info:
  title: Papnoticon API
  description: API para manejar usuarios y autenticación en el proyecto Papnoticon.
  version: 1.0.0
servers:
  - url: http://localhost:8000
    description: Servidor de desarrollo
paths:
  /login:
    post:
      summary: Iniciar sesión
      description: Autenticar a un usuario y obtener un token de acceso.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                username:
                  type: string
                  example: usuario123
                password:
                  type: string
                  format: password
                  example: contraseña_secreta
      responses:
        '200':
          description: Inicio de sesión exitoso
          content:
            application/json:
              schema:
                type: object
                properties:
                  token:
                    type: string
                    example: eyJhbGciOiJIUzI1NiIsInR5cGU6IjY0Ij0:
          description: Credenciales inválidas
  /logout:
    post:
      summary: Cerrar sesión
```

description: Invalidar el token de sesión del usuario.

responses:

'200':

description: Sesión cerrada exitosamente

/usuarios:

get:

summary: Listar usuarios

description: Obtén una lista de usuarios registrados.

responses:

'200':

description: Lista de usuarios

content:

application/json:

schema:

type: array

items:

type: object

properties:

id:

type: integer

example: 1

username:

type: string

example: usuario123

email:

type: string

example: usuario@example.com

post:

summary: Crear usuario

description: Registrar un nuevo usuario en el sistema.

requestBody:

required: true

content:

application/json:

schema:

type: object

properties:

username:

type: string

example: nuevo_usuario

password:

type: string

format: password

example: nueva_contraseña

email:

type: string

example: usuario_nuevo@example.com

responses:

'201':

description: Usuario creado exitosamente

La API cuenta con las siguientes rutas::

Rutas de la api Panopticon obtenidas mediante Swagger Editor:

default ^	
POST	/login Iniciar sesión
POST	/logout Cerrar sesión
GET	/usuarios Listar usuarios
POST	/usuarios Crear usuario

❖ /login (POST):

Utiliza el método POST y está diseñada para permitir a los usuarios iniciar sesión en el sistema. Para autenticarse, el usuario debe proporcionar su nombre de usuario (username) y contraseña (password). Si las credenciales son correctas, la API responde con un código de estado 200 y devuelve un token de acceso (token) en formato JWT, que puede usarse para acceder a recursos protegidos. En caso de credenciales incorrectas, la API responde con un código de estado 401, indicando que no se pudo autenticar al usuario.

❖ /logout (POST):

La ruta /logout, también con el método POST, permite cerrar la sesión de un usuario invalidando su token de acceso. Si la operación se realiza correctamente, la API responde con un código de estado 200 para indicar que la sesión se ha cerrado exitosamente.

❖ /usuarios (GET, POST):

La ruta /usuarios soporta dos métodos: GET y POST. Con el método GET, la API permite obtener una lista de todos los usuarios registrados en el sistema. La respuesta incluye detalles como el identificador único del usuario (id), su nombre de usuario (username) y su correo electrónico (email). Esta operación responde con un código de estado 200 al completarse correctamente.

El método POST, en cambio, está diseñado para registrar nuevos usuarios. Se espera que el cliente envíe en el cuerpo de la solicitud un objeto JSON con los campos obligatorios username, password y email. Si la operación es exitosa, la API responde con un código de estado 201, confirmando que el usuario ha sido creado. En caso de errores, como datos faltantes o inválidos, la API devuelve un código de estado 400.

API de Registro y Productos

PapnoticonAPI.json

openapi: 3.0.0

info:

title: API de Registro y Productos

description: Endpoints para registrar usuarios y listar productos con información adicional sobre planes y precios.

version: 1.0.0

servers:

- url: http://localhost:8000

description: Servidor de desarrollo

paths:

/registro:

post:

summary: Registrar un nuevo usuario

description:

Permite registrar un nuevo usuario en el sistema. Si el registro es exitoso, el usuario se asocia al grupo `Cliente` y es redirigido a la página de inicio de sesión.

requestBody:

required: true

content:

application/x-www-form-urlencoded:

schema:

type: object

properties:

username:

type: string

example: nuevo_usuario

password:

type: string

format: password

example: contrasena_segura

email:

type: string

example: usuario@example.com

responses:

'302':

description: Registro exitoso y redirección a la página de inicio de sesión.

'400':

description: Datos inválidos en el formulario de registro.

/productos:

get:

summary: Listar productos

description:

Devuelve una lista de productos disponibles en el sistema. Si se proporciona un parámetro de búsqueda (`q`), los resultados se filtran por nombre del producto que coincida parcialmente con la consulta. Adicionalmente, se incluye información sobre planes y precios obtenida de servicios externos.

parameters:

- name: q

in: query


```
description: Parámetro opcional para filtrar productos por nombre.
required: false
schema:
  type: string
  example: nombre_producto
responses:
  '200':
    description: Lista de productos con información adicional de precios y planes.
    content:
      application/json:
        schema:
          type: object
          properties:
            productos:
              type: array
              items:
                type: object
                properties:
                  id:
                    type: integer
                    example: 1
                  nombre:
                    type: string
                    example: Producto A
                  descripcion:
                    type: string
                    example: Descripción del producto A.
            precios_y_planes:
              type: object
              properties:
                Familiar:
                  type: string
                  example: "$20/mes"
                Individual:
                  type: string
                  example: "$10/mes"
                Estudiantes:
                  type: string
                  example: "$5/mes"
```

La API cuenta con las siguientes rutas:

Rutas de la api Registro y Productos obtenidas mediante Swagger Editor:

default ^

POST

/registro Registrar un nuevo usuario

✓

GET

/productos Listar productos

✓

❖ Ruta /registro (POST):

La ruta /registro permite registrar un nuevo usuario en el sistema. Cuando se envían datos válidos mediante un formulario, como el nombre de usuario, la contraseña y el correo electrónico, el usuario se crea exitosamente y se le asigna automáticamente al grupo Cliente. Después del registro, el usuario es redirigido a la página de inicio de sesión para autenticar su cuenta. En caso de que los datos proporcionados no sean válidos, el sistema devuelve un mensaje de error indicando que el registro no fue exitoso.

❖ Ruta /productos (GET):

La ruta /productos devuelve una lista de productos disponibles en el sistema. Si el cliente proporciona un parámetro de búsqueda que, la lista se filtra para mostrar únicamente aquellos productos cuyo nombre coincida parcialmente con la consulta. Además, junto con los productos, la ruta incluye información sobre precios y planes obtenida de servicios externos, lo que permite a los usuarios comparar fácilmente opciones como planes "Familiar", "Individual" o "Estudiantes".

API de Comentarios de Twitter

Comentarios de Twitter.json
<p>openapi: 3.0.0</p> <p>info:</p> <ul style="list-style-type: none">title: API de Comentarios de Twitterdescription: API que obtiene tweets recientes con un hashtag específico y devuelve el texto, autor y URLs de medios adjuntos.version: 1.0.0 <p>paths:</p> <ul style="list-style-type: none">/obtener-tweets:<ul style="list-style-type: none">get:<ul style="list-style-type: none">summary: Obtener comentarios (tweets) recientes con un hashtagdescription: Recupera una lista de tweets recientes que contienen un hashtag específico. Devuelve el texto del tweet, el autor y los medios adjuntos (si existen).parameters:<ul style="list-style-type: none">- name: queryin: querydescription: El hashtag o consulta para buscar tweets. (Por defecto, #Hola)required: falseschema:<ul style="list-style-type: none">type: stringexample: "#Hola" <p>responses:</p> <ul style="list-style-type: none">'200':<ul style="list-style-type: none">description: Lista de tweets obtenida exitosamentecontent:<ul style="list-style-type: none">application/json:<ul style="list-style-type: none">schema:<ul style="list-style-type: none">type: arrayitems:<ul style="list-style-type: none">type: objectproperties:<ul style="list-style-type: none">texto:<ul style="list-style-type: none">type: stringdescription: El contenido del tweet.example: "Este es un tweet de ejemplo con el hashtag #Hola."autor:<ul style="list-style-type: none">type: stringdescription: Nombre de usuario del autor del tweet.example: "usuario123"media_urls:<ul style="list-style-type: none">type: arraydescription: Lista de URLs de los medios adjuntos al tweet (por ejemplo, imágenes).items:<ul style="list-style-type: none">type: stringformat: url

```
example: "https://pbs.twimg.com/media/imagen.jpg"
'500':
  description: Error al conectar con la API de Twitter
  content:
    application/json:
      schema:
        type: object
        properties:
          error:
            type: string
            description: Mensaje de error retornado por la API de Twitter.
            example: "Error al conectar con la API de Twitter: TweepyException"
```

Ruta de la api de Comentarios de Twitter obtenida mediante Swagger Editor:

default

GET

/obtener-tweets Obtener comentarios (tweets) recientes con un hashtag

❖ /obtener-tweets

Utiliza el método GET para obtener tweets recientes relacionados con un hashtag específico, que se pasa como un parámetro opcional de búsqueda llamado query. Si no se especifica ningún parámetro de búsqueda, se utiliza #Hola por defecto. El servidor se conecta a la API de Twitter utilizando el token de acceso (Bearer token) configurado en los ajustes, y realiza una búsqueda para recuperar tweets recientes con información adicional sobre el autor (nombre de usuario) y los medios adjuntos (como imágenes). El resultado de la solicitud será una lista de tweets, que incluye el texto del tweet, el nombre de usuario del autor, y las URLs de los medios adjuntos, si los hay. Si no se encuentran tweets o ocurre un error de conexión con la API de Twitter, se devuelve un mensaje de error.

API de Contacto

API de Contacto.json

```
openapi: 3.0.0
info:
  title: Contact API
  description: API para manejar el formulario de contacto, el envío de mensajes y la visualización de comentarios.
  version: 1.0.0
paths:
  /contacto:
    get:
      summary: Mostrar formulario de contacto
      description: Carga un formulario de contacto para que los usuarios puedan enviar mensajes al administrador del sistema.
      responses:
        '200':
          description: Formulario de contacto cargado exitosamente
          content:
            text/html:
              schema:
                type: string
              example: "<html><body>Formulario de contacto</body></html>"
    post:
      summary: Procesar envío del formulario de contacto
      description: Procesa los datos enviados a través del formulario de contacto. Si los datos son válidos, se puede manejar el mensaje (por ejemplo, enviarlo por correo electrónico o almacenarlo en la base de datos) y se informa al usuario que el mensaje ha sido enviado.
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
            properties:
              nombre:
                type: string
                description: Nombre del remitente.
                example: "Juan Pérez"
              email:
                type: string
                description: Correo electrónico del remitente.
                format: email
                example: "juan.perez@example.com"
              mensaje:
                type: string
                description: Contenido del mensaje enviado por el usuario.
                example: "Hola, estoy interesado en sus servicios."
      responses:
        '200':
          description: Mensaje enviado exitosamente
          content:
```

```
application/json:
  schema:
    type: object
    properties:
      mensaje_enviado:
        type: boolean
        description: Indica si el mensaje fue enviado exitosamente.
        example: true
'400':
  description: Error en los datos enviados
  content:
    application/json:
      schema:
        type: object
        properties:
          error:
            type: string
            description: Descripción del error ocurrido durante el envío del formulario.
            example: "Faltan campos obligatorios."
```

/enviar_mensaje:

```
post:
  summary: Enviar mensaje
  description: Procesa el formulario de contacto y envía el mensaje si los datos son válidos.
  requestBody:
    required: true
    content:
      application/x-www-form-urlencoded:
        schema:
          type: object
          properties:
            nombre:
              type: string
              description: Nombre del remitente.
              example: "Juan Pérez"
            email:
              type: string
              description: Correo electrónico del remitente.
              format: email
              example: "juan.perez@example.com"
            mensaje:
              type: string
              description: Contenido del mensaje enviado por el usuario.
              example: "Hola, estoy interesado en sus servicios."
```

responses:

```
'200':
  description: Mensaje enviado correctamente
  content:
    application/json:
      schema:
        type: object
        properties:
          mensaje:
            type: string
```

```

        example: "Mensaje enviado correctamente"
'400':
  description: Error al procesar el mensaje
  content:
    application/json:
      schema:
        type: object
        properties:
          error:
            type: string
            example: "Faltan campos obligatorios o error en los datos."

/comentarios:
  get:
    summary: Ver comentarios
    description: Devuelve la vista con los comentarios o mensajes enviados a través del formulario de contacto.
    responses:
      '200':
        description: Vista de comentarios cargada exitosamente
        content:
          text/html:
            schema:
              type: string
              example: "<html><body>Comentarios y mensajes enviados</body></html>"

```

La API cuenta con las siguientes rutas::

Rutas de la API de Contacto obtenidas mediante Swagger Editor:			
default		^	
GET	/contacto	Mostrar formulario de contacto	▼
POST	/contacto	Procesar envío del formulario de contacto	▼
POST	/enviar_mensaje	Enviar mensaje	▼
GET	/comentarios	Ver comentarios	▼

❖ **/contacto (GET):**

Esta ruta permite cargar y mostrar un formulario de contacto en el que los usuarios pueden ingresar su nombre, correo electrónico y mensaje para contactar al administrador del sistema. La solicitud GET recupera el formulario, que es devuelto en formato HTML. Es útil para mostrar al usuario la interfaz de contacto donde podrá enviar su mensaje. La respuesta esperada es un código 200, lo que indica que el formulario se cargó correctamente.

❖ **/contacto (POST):**

Esta ruta procesa los datos enviados a través del formulario de contacto. Cuando un usuario llena el formulario con su nombre, correo electrónico y mensaje, la solicitud POST envía estos datos al servidor para su validación y procesamiento. Si los datos son válidos, el sistema puede realizar diversas acciones, como almacenar el mensaje en la base de datos o enviarlo por correo electrónico. La respuesta esperada para una solicitud exitosa es un código 200, con un mensaje que confirma que el mensaje fue enviado correctamente. Si los datos enviados son incorrectos o incompletos, el servidor responde con un código 400, junto con un mensaje que describe el error, como la falta de campos obligatorios.

❖ **/enviar_mensaje (POST):**

Esta ruta procesa los datos enviados desde el formulario de contacto. Si la solicitud es correcta, se devuelve un mensaje de éxito en formato JSON. Si hay errores, se devuelve un mensaje con la causa del error.

❖ **/comentarios (GET):**

Esta ruta muestra una página con los comentarios o mensajes enviados a través del formulario de contacto. Es útil para que los administradores visualicen los mensajes recibidos.

API publicaciones de facebook

API publicaciones de facebook.json

openapi: 3.0.0

info:

title: API de Publicaciones en Facebook

description: API para gestionar publicaciones, programaciones, imágenes y eliminaciones en una página de Facebook.

version: 1.0.0

paths:

/registrar_publicacion:

post:

summary: Registrar una nueva publicación en el archivo de log

description: Esta ruta permite registrar una publicación en un archivo de log, incluyendo el post_id, mensaje y tipo de publicación.

requestBody:

required: true

content:

application/json:

schema:

type: object

properties:

post_id:

type: string

description: ID del post a registrar.

example: "12345"

mensaje:

type: string

description: Mensaje de la publicación.

example: "Nuevo contenido disponible."

tipo:

type: string

description: Tipo de publicación (ejemplo: 'texto', 'imagen').

example: "texto"

responses:

'200':

description: Publicación registrada exitosamente en el archivo de log.

content:

application/json:

schema:

type: object

properties:

status:

type: string

example: "success"

message:

type: string

example: "Publicación registrada en el archivo de log."

'500':
description: Error al escribir en el archivo de log.
content:
application/json:
schema:
type: object
properties:
status:
type: string
example: "error"
message:
type: string
example: "Error al escribir en el archivo."

/publicar_en_pagina:
post:
summary: Publicar mensaje en una página de Facebook
description: Esta ruta permite realizar una publicación en una página de Facebook utilizando la API Graph de Facebook.
requestBody:
required: true
content:
application/json:
schema:
type: object
properties:
page_id:
type: string
description: ID de la página en la que se publicará el mensaje.
example: "1234567890"
mensaje:
type: string
description: Mensaje a publicar en la página.
example: "Este es un mensaje de prueba."
page_access_token:
type: string
description: Token de acceso de la página de Facebook.
example: "EAAGm0PX4ZCpsBAOaZB6Yq4xL7v0F8w"
responses:
'200':
description: Mensaje publicado exitosamente en la página.
content:
application/json:
schema:
type: object
properties:
status:
type: string
example: "success"

```

    post_id:
      type: string
      example: "post_12345"
'400':
  description: Error al publicar el mensaje en la página.
  content:
    application/json:
      schema:
        type: object
        properties:
          status:
            type: string
            example: "error"
          message:
            type: string
            example: "Error al conectar con la API de Facebook."

/programar_publicacion:
  post:
    summary: Programar una publicación en una página de Facebook
    description: Esta ruta permite programar una publicación en una página de Facebook
    en un tiempo futuro.
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              page_id:
                type: string
                description: ID de la página en la que se programará la publicación.
                example: "1234567890"
              mensaje:
                type: string
                description: Mensaje que se publicará en la página.
                example: "Este mensaje se publicará en el futuro."
              tiempo_programado:
                type: string
                description: Tiempo programado para la publicación en formato UNIX
(timestamp).
                example: "1633035600"
              page_access_token:
                type: string
                description: Token de acceso de la página de Facebook.
                example: "EAAGm0PX4ZCpsBAOaZB6Yq4xL7v0F8w"
  responses:
    '200':
      description: Publicación programada exitosamente.

```

content:
application/json:
schema:
type: object
properties:
status:
type: string
example: "success"
post_id:
type: string
example: "post_12345"

'400':
description: Error al programar la publicación.

content:
application/json:
schema:
type: object
properties:
status:
type: string
example: "error"
message:
type: string
example: "Error al conectar con la API de Facebook."

/subir_imagen:

post:

summary: Subir una imagen a una página de Facebook

description: Esta ruta permite subir una imagen junto con un mensaje en una página de Facebook.

requestBody:

required: true

content:

multipart/form-data:

schema:

type: object

properties:

page_id:

type: string

description: ID de la página en la que se subirá la imagen.

example: "1234567890"

imagen_path:

type: string

description: Ruta del archivo de imagen en el servidor.

example: "/path/to/image.jpg"

mensaje:

type: string

description: Mensaje que acompañará la imagen.

example: "Aquí va la imagen."

page_access_token:
type: string
description: Token de acceso de la página de Facebook.
example: "EAAGm0PX4ZCpsBAOaZB6Yq4xL7v0F8w"

responses:
'200':
description: Imagen subida exitosamente.
content:
application/json:
schema:
type: object
properties:
status:
type: string
example: "success"
post_id:
type: string
example: "post_12345"

'400':
description: Error al subir la imagen.
content:
application/json:
schema:
type: object
properties:
status:
type: string
example: "error"
message:
type: string
example: "Error al subir la imagen."

/eliminar_publicacion:
delete:
summary: Eliminar una publicación de Facebook
description: Esta ruta permite eliminar una publicación de Facebook mediante su ID.
parameters:
- in: query
name: post_id
required: true
description: ID de la publicación que se desea eliminar.
schema:
type: string
example: "12345"
- in: query
name: page_access_token
required: true
description: Token de acceso de la página de Facebook.
schema:

```

    type: string
    example: "EAAGm0PX4ZCpsBAOaZB6Yq4xL7v0F8w"
responses:
  '200':
    description: Publicación eliminada exitosamente.
    content:
      application/json:
        schema:
          type: object
          properties:
            status:
              type: string
              example: "success"
            message:
              type: string
              example: "Publicación eliminada correctamente."
  '400':
    description: Error al eliminar la publicación.
    content:
      application/json:
        schema:
          type: object
          properties:
            status:
              type: string
              example: "error"
            message:
              type: string
              example: "Error al eliminar la publicación."

```

La API cuenta con las siguientes rutas::

❖ **/registrar_publicacion (POST):**

Esta ruta permite registrar los detalles de una publicación en un archivo de log local. Recibe como parámetros el identificador único del post (post_id), el mensaje que se desea registrar, y el tipo de publicación (por ejemplo, texto, imagen, video, etc.). El log incluye la fecha y hora en que se registró la publicación, proporcionando un historial detallado de las acciones realizadas en la plataforma.

❖ **/publicar_en_pagina (POST):**

Este endpoint permite publicar un mensaje en una página de Facebook específica. Para ello, se requiere el ID de la página donde se publicará el contenido, el mensaje que será publicado, y un token de acceso válido de la página. La API de Facebook se utiliza para gestionar esta operación, garantizando la autenticación y la correcta publicación en el destino deseado.

❖ **/programar_publicacion (POST):**

Diseñada para facilitar la gestión de publicaciones planificadas, esta ruta permite programar un mensaje para ser publicado automáticamente en una página de Facebook en el futuro. Los parámetros incluyen el ID de la página, el mensaje que se publicará, la hora programada en formato Unix timestamp, y un token de acceso válido. Esto resulta útil para administrar campañas o publicaciones periódicas sin necesidad de intervención manual en el momento de la publicación.

❖ **/subir_imagen (POST):**

A través de esta ruta, es posible subir una imagen a una página de Facebook acompañada de un mensaje descriptivo. El endpoint requiere el ID de la página, la ruta del archivo de imagen en el servidor, el mensaje que se asociará a la imagen, y un token de acceso válido. Además de subir la imagen, la API devuelve el ID de la publicación creada si la operación se realiza con éxito.

❖ **/eliminar_publicacion (DELETE):**

Este endpoint permite eliminar una publicación específica de una página de Facebook utilizando su identificador único (post_id). Se requiere un token de acceso válido de la página para autenticar la solicitud. La eliminación es permanente y asegura que el contenido ya no sea visible para los seguidores de la página, lo que puede ser útil para corregir errores o retirar contenido desactualizado.

API para Carrito de Compras y Pasarela de Pagos

API para Carrito de Compras y Pasarela de Pagos.json

openapi: 3.0.0

info:

title: API para Carrito de Compras y Pasarela de Pagos

description: API para gestionar el carrito de compras y las pasarelas de pago con Stripe.

version: 1.0.0

paths:

/agregar_al_carrito/{producto_id}:

post:

summary: Agregar un producto al carrito de compras

description: Permite agregar un producto al carrito de compras. Si el producto ya está en el carrito, actualiza la cantidad.

parameters:

- in: path

name: producto_id

required: true

description: ID del producto a agregar al carrito.

schema:

type: integer

requestBody:

required: true

content:

application/x-www-form-urlencoded:

schema:

type: object

properties:

cantidad:

type: integer

description: Cantidad de productos a agregar.

example: 2

responses:

'200':

description: Producto agregado al carrito con éxito.

'400':

description: Error al agregar el producto al carrito.

'404':

description: Producto no encontrado o stock insuficiente.

/ver_carrito:

get:

summary: Ver los productos en el carrito

description: Permite visualizar todos los productos que el usuario tiene en su carrito con el subtotal y el total.

responses:

'200':

description: Vista de los productos en el carrito.


```

content:
  application/json:
    schema:
      type: object
      properties:
        cart_items_with_subtotal:
          type: array
          items:
            type: object
            properties:
              item:
                type: object
                description: Detalle del producto en el carrito.
              subtotal:
                type: number
                format: float
                description: Subtotal del producto (precio * cantidad).
            total:
              type: number
              format: float
              description: Total del carrito.

```

```

/actualizar_carrito:
  post:
    summary: Actualizar la cantidad de productos en el carrito
    description: Permite actualizar la cantidad de productos en el carrito. Si la cantidad es mayor al stock, devuelve un error.
    requestBody:
      required: true
      content:
        application/x-www-form-urlencoded:
          schema:
            type: object
            properties:
              cantidad_{item_id}:
                type: integer
                description: Nueva cantidad del producto con `item_id` en el carrito.
                example: 3
    responses:
      '200':
        description: Carrito actualizado con éxito.
      '400':
        description: Error al actualizar el carrito debido a cantidad no válida o stock insuficiente.

```

```

/eliminar_del_carrito/{item_id}:
  delete:
    summary: Eliminar un producto del carrito
    description: Elimina un producto del carrito por su ID.

```

```
parameters:
  - in: path
    name: item_id
    required: true
    description: ID del producto a eliminar del carrito.
    schema:
      type: integer
responses:
  '200':
    description: Producto eliminado del carrito con éxito.
  '404':
    description: Producto no encontrado en el carrito.
```

/checkout:

```
post:
  summary: Iniciar proceso de pago
  description: Inicia el proceso de pago utilizando Stripe para realizar el checkout de los
productos en el carrito.
responses:
  '200':
    description: Sesión de Stripe creada con éxito.
    content:
      application/json:
        schema:
          type: object
          properties:
            id:
              type: string
              description: ID de la sesión de Stripe creada.
              example: "cs_test_a1b2c3d4e5f6"
  '400':
    description: Error al crear la sesión de pago.
```

/pago_exitoso:

```
get:
  summary: Confirmación de pago exitoso
  description: Endpoint para manejar la confirmación de un pago exitoso y procesar el
pedido.
parameters:
  - in: query
    name: session_id
    required: true
    description: Identificador de la sesión de pago.
    schema:
      type: string
responses:
  '200':
    description: Pedido procesado y pago realizado con éxito.
  '400':
```

description: Error al procesar el pedido o al realizar el pago.

'404':

description: No se pudo encontrar los artículos del carrito.

/mis_planes:

get:

summary: Ver los planes/pedidos del usuario

description: Permite al usuario ver sus pedidos completados.

responses:

'200':

description: Lista de pedidos completados del usuario.

content:

text/html:

schema:

type: string

description: Página HTML que muestra los pedidos completados del usuario.

'404':

description: No se encontraron pedidos completados para el usuario.

/descargar_archivo_txt/{pedido_id}:

get:

summary: Descargar archivo de texto de un pedido

description: Permite al usuario descargar un archivo de texto con los detalles de un pedido específico.

parameters:

- in: path

name: pedido_id

required: true

description: ID del pedido para generar el archivo de texto.

schema:

type: integer

responses:

'200':

description: Archivo de texto generado y listo para descargar.

content:

text/plain:

schema:

type: string

example: |

Detalles del Pedido #1234

Fecha: 12-11-2024

Total: \$100.00

'404':

description: Pedido no encontrado o no disponible para el usuario.

La API cuenta con las siguientes rutas::

Rutas de la API para Carrito de Compras y Pasarela de Pagos mediante Swagger Editor:

POST	/agregar_al_carrito/{producto_id}	Agregar un producto al carrito de compras	▼
GET	/ver_carrito	Ver los productos en el carrito	▼
POST	/actualizar_carrito	Actualizar la cantidad de productos en el carrito	▼
DELETE	/eliminar_del_carrito/{item_id}	Eliminar un producto del carrito	▼
POST	/checkout	Iniciar proceso de pago	▼
GET	/pago_exitoso	Confirmación de pago exitoso	▼
GET	/mis_planes	Ver los planes/pedidos del usuario	▼
GET	/descargar_archivo_txt/{pedido_id}	Descargar archivo de texto de un pedido	▼

❖ /agregar_al_carrito/{producto_id} (POST):

Esta ruta permite agregar un producto al carrito de compras de un usuario. Si el producto ya está en el carrito, el sistema actualiza la cantidad del producto según lo que se indique en la solicitud. Si el producto no está en el carrito, se agrega con la cantidad especificada. Antes de agregar o actualizar, se realiza una validación para asegurar que la cantidad solicitada no exceda el stock disponible del producto. En caso de que la cantidad solicitada sea mayor al stock disponible, el sistema devuelve un error indicando que no hay suficiente inventario. Si la cantidad es válida, el producto se agrega correctamente al carrito.

❖ /ver_carrito (GET):

Esta ruta permite al usuario visualizar los productos que ha agregado a su carrito de compras. Para cada producto, se muestra su cantidad y el subtotal calculado multiplicando el precio por la cantidad. Además, se calcula el total del carrito, sumando los subtotales de todos los productos. Si no hay productos en el carrito, se devuelve una respuesta vacía o un mensaje indicando que el carrito está vacío.

❖ **/actualizar_carrito (POST):**

Esta ruta permite actualizar la cantidad de un producto ya presente en el carrito del usuario. Al recibir una solicitud, el sistema verifica si la nueva cantidad solicitada es válida. Si la cantidad es mayor que el stock disponible, se muestra un mensaje de error. También se valida que la cantidad no sea menor que 1, ya que el mínimo permitido es 1. Si la cantidad es válida, el carrito se actualiza con la nueva cantidad. Si el usuario intenta establecer una cantidad inválida, se le informará sobre el error con un mensaje adecuado.

❖ **/eliminar_del_carrito/{item_id}` (DELETE):**

Esta ruta permite al usuario eliminar un producto específico de su carrito de compras. El producto a eliminar se identifica mediante su `item_id`, que es un identificador único dentro del carrito del usuario. Si el producto existe en el carrito, se elimina de la base de datos y el usuario recibe un mensaje de éxito. Si no se encuentra el producto o el ID es incorrecto, se devuelve un error indicando que el producto no fue encontrado.

❖ **/checkout (POST):**

Esta ruta inicia el proceso de pago mediante Stripe, creando una sesión de pago para el usuario. La solicitud genera una sesión de pago en la que se incluyen todos los productos presentes en el carrito del usuario. La sesión se configura con los detalles del pago y se genera un enlace para que el usuario complete el proceso de pago en la plataforma de Stripe. Al finalizar el pago, el sistema podrá verificar si la transacción fue exitosa y procesar el pedido. Si la sesión se crea correctamente, se devuelve un ID de sesión para que el usuario continúe el pago.

❖ **/pago_exitoso (GET):**

Este endpoint se activa cuando el usuario ha completado el pago exitosamente a través de Stripe. Utilizando el `session_id` proporcionado por Stripe, el sistema verifica que la transacción haya sido exitosa. Si la compra es confirmada, se crea un pedido en el sistema, asignando los productos del carrito a dicho pedido. Además, se actualiza el stock de los productos comprados y se vacía el carrito del usuario. Si el pago no se ha completado correctamente o si no se recibe un `session_id`, se informa al usuario que el pago no se procesó correctamente.

❖ **/mis_planes (GET):**

Este endpoint devuelve los pedidos completados por el usuario, mostrando una lista de los pedidos que han sido procesados y pagados exitosamente. Si el usuario no tiene pedidos completados, se devuelve una respuesta vacía o un mensaje indicando que no se han encontrado pedidos.

❖ /descargar_archivo_txt/{pedido_id} (GET):

Este endpoint permite al usuario descargar un archivo de texto con los detalles de un pedido específico. El archivo incluye información relevante como el ID del pedido, la fecha de creación y el total de la compra. Esta funcionalidad es útil para aquellos usuarios que deseen guardar un registro de su pedido en formato de texto. El archivo se genera dinámicamente y se prepara para su descarga, proporcionando al usuario un archivo con la extensión `.txt` que contiene los detalles completos de su compra.

API de Gestión de Publicaciones en Facebook

API de Gestión de Publicaciones en Facebook.json
<pre>openapi: 3.0.0 info: title: API de Gestión de Publicaciones en Facebook version: 1.0.0 description: API para gestionar publicaciones en una página de Facebook, incluyendo su visualización, publicación, programación y eliminación. paths: /mostrar_publicacion: get: summary: Mostrar la página de publicaciones description: Renderiza una página HTML para la gestión de publicaciones si el usuario está autenticado y tiene permisos de administrador. Si no está autenticado, redirige al login. responses: '200': description: Página renderizada correctamente. content: text/html: schema: type: string '302': description: Redirección al login si el usuario no está autenticado. /publicar: post: summary: Publicar un mensaje en Facebook description: Permite a un usuario administrador publicar un mensaje en una página de Facebook. requestBody: required: true content: application/json: schema: type: object</pre>

```

    properties:
      mensaje:
        type: string
        description: El mensaje que se desea publicar.
        example: "Este es un mensaje de prueba."
  responses:
    '200':
      description: Publicación realizada con éxito.
      content:
        application/json:
          schema:
            type: object
            properties:
              id:
                type: string
                description: ID de la publicación creada.
    '403':
      description: Permiso denegado. El usuario no es administrador.
    '405':
      description: Método no permitido.

/programar:
  post:
    summary: Programar una publicación en Facebook
    description: Permite programar una publicación en una página de Facebook para un
momento futuro. Solo accesible por administradores autenticados.
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              mensaje:
                type: string
                description: El mensaje a publicar.
                example: "Este mensaje será publicado más tarde."
              tiempo_programado:
                type: integer
                description: Timestamp Unix indicando cuándo debe realizarse la publicación.
                example: 1715049600
  responses:
    '200':
      description: Publicación programada con éxito.
      content:
        application/json:
          schema:
            type: object
            properties:

```

```

      id:
        type: string
        description: ID de la publicación programada.
    '403':
        description: Permiso denegado. El usuario no es administrador.
    '500':
        description: Error interno del servidor.

/subir_imagen:
  post:
    summary: Subir una imagen con un mensaje a Facebook
    description: Permite a los administradores autenticados subir una imagen acompañada
de un mensaje a una página de Facebook.
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            type: object
            properties:
              mensaje:
                type: string
                description: El mensaje que acompañará a la imagen.
                example: "Esta es una publicación con imagen."
              imagen:
                type: string
                format: binary
                description: La imagen a subir.
    responses:
      '200':
        description: Imagen subida con éxito.
        content:
          application/json:
            schema:
              type: object
              properties:
                post_id:
                  type: string
                  description: ID de la publicación creada en Facebook.
      '401':
        description: Usuario no autenticado.
        content:
          application/json:
            schema:
              type: object
              properties:
                status:
                  type: string
                  example: error

```



```

    message:
      type: string
      example: "Usuario no autenticado."
'403':
  description: Permiso denegado. El usuario no es administrador.
  content:
    application/json:
      schema:
        type: object
        properties:
          status:
            type: string
            example: error
          message:
            type: string
            example: "Permiso denegado."
'500':
  description: Error interno del servidor.
  content:
    application/json:
      schema:
        type: object
        properties:
          status:
            type: string
            example: error
          message:
            type: string
            example: "Error interno del servidor."

/eliminar:
  delete:
    summary: Eliminar una publicación de Facebook
    description: Permite a los administradores autenticados eliminar una publicación
    específica en una página de Facebook usando su ID.
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              post_id:
                type: string
                description: ID de la publicación a eliminar.
                example: "1234567890"
    responses:
      '200':
        description: Publicación eliminada con éxito.

```

```
content:
  application/json:
    schema:
      type: object
      properties:
        status:
          type: string
          example: success
        message:
          type: string
          example: "Publicación eliminada."
```

'401':

description: Usuario no autenticado.

```
content:
  application/json:
    schema:
      type: object
      properties:
        status:
          type: string
          example: error
        message:
          type: string
          example: "Usuario no autenticado."
```

'403':

description: Permiso denegado. El usuario no es administrador.

```
content:
  application/json:
    schema:
      type: object
      properties:
        status:
          type: string
          example: error
        message:
          type: string
          example: "Permiso denegado."
```

'400':

description: Falta el ID de la publicación.

```
content:
  application/json:
    schema:
      type: object
      properties:
        status:
          type: string
          example: error
        message:
          type: string
```

example: "Falta el ID de la publicación."

La API cuenta con las siguientes rutas::

Rutas de la API de Gestión de Publicaciones en Facebook mediante Swagger Editor:

default

GET

/mostrar_publicacion

Mostrar la página de publicaciones

POST

/publicar

Publicar un mensaje en Facebook

POST

/programar

Programar una publicación en Facebook

POST

/subir_imagen

Subir una imagen con un mensaje a Facebook

DELETE

/eliminar

Eliminar una publicación de Facebook

❖ **/mostrar_publicacion (GET):**

Esta ruta permite a los administradores autenticados acceder a una página HTML para gestionar publicaciones en Facebook. Si el usuario tiene permisos de administrador y está autenticado, el sistema renderiza una interfaz para visualizar, publicar o programar contenido. Si no está autenticado, se redirige automáticamente al login. En caso exitoso, retorna un código 200 con la página HTML; si no cumple las condiciones, devuelve un código 302 con una redirección.

❖ **/publicar (POST):**

Permite a los usuarios con permisos de administrador publicar un mensaje en una página de Facebook. El cliente envía un cuerpo JSON con el mensaje a publicar, y si la solicitud es válida, el sistema crea la publicación y devuelve el ID de la misma en un código 200. Si el usuario no es administrador, se devuelve un código 403 indicando "Permiso denegado". Si el método HTTP no es POST, se responde con un código 405.

❖ **/programar (POST):**

Ofrece la funcionalidad de programar una publicación en una página de Facebook para una fecha futura. El cliente debe enviar un cuerpo JSON que incluya el mensaje a publicar y el momento de publicación en formato timestamp Unix. Si la programación es exitosa, devuelve un código 200 junto con el ID de la publicación. Si el usuario no tiene permisos de administrador, devuelve un código 403. En caso de errores internos del servidor, como problemas con la programación, retorna un código 500.

❖ **/subir_imagen (POST):**

Permite a los administradores autenticados subir una imagen acompañada de un mensaje a una página de Facebook. La solicitud debe incluir el mensaje y la imagen en formato multipart/form-data. La respuesta devuelve el post_id de la publicación creada, o un error si el usuario no está autenticado o no tiene permisos.

❖ **/eliminar (DELETE):**

Permite a los administradores autenticados eliminar una publicación en Facebook proporcionando el post_id de la publicación en el cuerpo de la solicitud. La respuesta confirma la eliminación o indica un error si el usuario no está autenticado, no tiene permisos o falta el post_id.

Manual de Usuario para Panopticon

1. Introducción

Bienvenido al manual de usuario de **Panopticon**, una plataforma avanzada diseñada para optimizar la productividad mediante la restricción de distracciones y la gestión eficiente del tiempo. A través de este sistema, los usuarios pueden organizar sus actividades diarias, controlar el acceso a aplicaciones distractoras y, en versiones avanzadas, realizar publicaciones programadas en redes sociales como Facebook. Este documento tiene como objetivo proporcionar una guía exhaustiva sobre cómo navegar por el sitio, acceder a las funcionalidades disponibles y maximizar el uso de las herramientas ofrecidas.

2. Requisitos Previos

Para acceder a todas las funcionalidades de **Panopticon**, es necesario cumplir con los siguientes requisitos previos:

- **Acceso a Internet:** Se recomienda una conexión estable para garantizar un uso fluido de la plataforma.
- **Cuenta de Usuario:** Es necesario registrarse e iniciar sesión para acceder a funcionalidades avanzadas, como la publicación en Facebook.
- **Navegador Web Compatible:** La plataforma es compatible con Google Chrome, Mozilla Firefox, Safari y Microsoft Edge.

3. Guía de Navegación

La estructura del sitio web está organizada de manera clara y eficiente para garantizar una navegación intuitiva. En la parte superior de la página, encontrarás la **barra de navegación** (app bar) que te permite acceder a las principales secciones del sitio.

3.1 Menú Principal

A continuación se describen las secciones accesibles desde la barra de navegación:

1. Inicio

La página inicial proporciona un resumen de las principales funcionalidades del sitio, destacando las características más relevantes y los planes de suscripción disponibles. Esta página está diseñada para ofrecerte una visión general del sistema y sus capacidades.

2. Productos

En esta sección, los usuarios pueden explorar los productos disponibles, comparar las distintas opciones según sus características y precios, y tomar decisiones informadas sobre cuál es el más adecuado para sus necesidades. Además, se pueden realizar búsquedas y filtrados para facilitar la selección.

3. Contacto

Si necesitas asistencia técnica o deseas realizar consultas sobre el uso de la plataforma, esta sección te permitirá enviar mensajes directamente al equipo de soporte. Aquí podrás completar un formulario con tus datos y especificar el tipo de consulta que deseas realizar.

4. Comentarios

Esta sección está destinada a recoger retroalimentación de los usuarios. Puedes compartir tus opiniones sobre el sitio y sugerir mejoras. Las valoraciones y comentarios ayudan al equipo de desarrollo a mejorar la experiencia del usuario.

5. Mis Planes

En este apartado, podrás revisar los planes que has adquirido, ver sus detalles y verificar las características disponibles según tu suscripción. Si no has adquirido un plan, se te redirigirá a la sección de productos para explorar las opciones disponibles.

6. Carrito

Si has añadido productos a tu cesta de compras, puedes revisarlos en esta sección antes de proceder con el pago. Si tu carrito está vacío, tendrás la opción de explorar los productos disponibles para añadirlos.

7. Publicaciones

Esta sección está habilitada solo para usuarios registrados e iniciados sesión. Desde aquí, podrás gestionar tus publicaciones en Facebook, incluyendo la creación de nuevos mensajes, la programación de publicaciones y la carga de imágenes.

8. Cerrar Sesión

Una vez que hayas terminado de utilizar la plataforma, puedes finalizar tu sesión

haciendo clic en este enlace. Esto garantizará que tu cuenta quede protegida y cerrada correctamente.

4. Funcionalidades Avanzadas

4.1 Inicio de Sesión

El acceso a funcionalidades avanzadas, como la gestión de publicaciones en redes sociales, requiere que el usuario inicie sesión en su cuenta. Para ello, es necesario seguir los siguientes pasos:

1. En la barra de navegación, haz clic en el enlace **Iniciar Sesión**.
2. Ingresa tu nombre de usuario y contraseña.
3. Haz clic en **Acceder** para iniciar sesión.

Si no tienes una cuenta, podrás registrarte en el enlace **Registrarse**. Asegúrate de proporcionar la información correcta durante el registro para evitar problemas de acceso.

4.2 Publicaciones en Facebook

Una de las funcionalidades más avanzadas de **Panopticon** es la capacidad de realizar publicaciones en Facebook directamente desde la plataforma. Para acceder a esta funcionalidad, es necesario que el usuario haya iniciado sesión.

Características disponibles en la sección Publicaciones:

- **Publicación inmediata:**

Los usuarios pueden escribir un mensaje directamente en el cuadro de texto y publicarlo en su perfil de Facebook de forma inmediata.

Pasos para publicar inmediatamente:

1. Haz clic en la sección **Publicaciones**.
 2. Escribe el contenido que desees compartir en el campo de texto.
 3. Haz clic en **Publicar** para enviarlo a Facebook.
- **Programación de publicaciones:**

La plataforma permite programar publicaciones para fechas y horas futuras, lo que

facilita la planificación de contenido.

Pasos para programar una publicación:

1. En la sección **Publicaciones**, escribe tu mensaje en el cuadro de texto.
2. Selecciona la fecha y hora en la que deseas que la publicación sea publicada.
3. Haz clic en **Programar**.

- **Subida de imágenes:**

Puedes acompañar tus publicaciones con imágenes desde tu dispositivo local.

Pasos para subir una imagen:

1. Haz clic en el ícono de imagen en la sección de **Publicaciones**.
2. Selecciona una imagen desde tu computadora.
3. Añade un mensaje opcional si lo deseas.
4. Haz clic en **Subir**.

- **Eliminación de publicaciones:**

Esta opción permite eliminar publicaciones previamente realizadas desde la plataforma.

Pasos para eliminar una publicación:

1. En la sección **Publicaciones**, busca el ID de la publicación que deseas eliminar.
2. Haz clic en **Eliminar**.

5. Resolución de Problemas Comunes

5.1 Problemas al Iniciar Sesión

Si experimentas dificultades al intentar iniciar sesión, considera las siguientes soluciones:

- Verifica que el nombre de usuario y la contraseña sean correctos.
- Si has olvidado tu contraseña, utiliza la opción **Recuperar contraseña** para restablecerla.

5.2 Publicaciones no Realizadas

Si una publicación no se realiza correctamente, revisa lo siguiente:

- Asegúrate de haber iniciado sesión correctamente en tu cuenta.

- Verifica que todos los campos requeridos (mensaje, fecha, imagen) estén completos.
- Si el problema persiste, contacta con el soporte técnico a través de la sección **Contacto**.

6. Contacto y Soporte

Para obtener asistencia adicional o resolver problemas específicos, puedes ponerte en contacto con el equipo de soporte de **Panopticon**:

- **Formulario de contacto:** Accede a la sección **Contacto** en el menú para completar un formulario.
- **Correo electrónico:** envía un correo a soporte@panopticon.com.
- **Teléfono:** Llama al número de soporte: **+123 456 7890**.

*Este manual está diseñado para proporcionarte una visión detallada y formal de cómo utilizar **Panopticon** de manera eficiente. Si tienes alguna pregunta o necesitas asistencia adicional, no dudes en ponerte en contacto con nuestro equipo de soporte.*

Pruebas

-1.

En Postman con método GET

```
<!DOCTYPE html>
<html lang="en">
<head>

  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Página de Inicio</title>
  <!-- Enlace a Bootstrap -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
  <!-- Íconos de Bootstrap -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.8.1/font/bootstrap-icons.css" rel="stylesheet">
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="/static/css/stylesMain.css"> <!-- Archivo CSS externo -->
</head>
<body>
  <!-- Barra de Navegación -->
  <nav class="navbar navbar-expand-lg navbar-light">
    <div class="container">
      <a class="navbar-brand d-flex align-items-center" href="/">
        
        Panopticon
      </a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ms-auto">
          <li class="nav-item"><a class="nav-link" href="/">Inicio</a></li>
          <li class="nav-item"><a class="nav-link" href="/productos/">Productos</a></li>
          <li class="nav-item"><a class="nav-link" href="/contacto/">Contacto</a></li>
          <li class="nav-item"><a class="nav-link" href="/comentarios/">Comentarios</a></li>
          <!-- Verificación de usuario autenticado -->

          <li class="nav-item"><a class="nav-link" href="/login/">Iniciar Sesión</a></li>

        </ul>
      </div>
    </div>
  </nav>
```

```

</div>
</nav>

<!-- Hero Section -->
<div class="container mt-5">
  <div class="hero-section">
    <h1 class="hero-title">Libera tu potencial, restringe las distracciones</h1>
    <p class="hero-subtitle">Controla el uso de tus dispositivos de forma rapida con nuestra aplicacion intuitiva y
personalizable</p>
    <a href="/productos/" class="btn btn-primary-custom mt-3">Explora nuestros planes</a>
  </div>
</div>

<!-- Features Section -->
<div class="container features-section">
  <h2 class="features-title">Caracteristicas fundamentales</h2>
  <p>Panopticon ofrece una variedad de funciones para ayudar a mantenerte concentrado y productivo en lo que de
verdad importa</p>
  <div class="row mt-4">
    <div class="col-md-6">
      <div class="feature-card">
        <i class="feature-icon bi bi-calendar-check"></i>
        <h5>Agenda</h5>
        <p>Planea y programa tus tareas vinculando los calendarios de Google o desde cero en nuestra app</p>
      </div>
    </div>
    <div class="col-md-6">
      <div class="feature-card">
        <i class="feature-icon bi bi-lock"></i>
        <h5>Restricción de Aplicaciones</h5>
        <p>Bloquea las aplicaciones y sitios web que te distraen de tus tareas</p>
      </div>
    </div>
  </div>
  <div class="row mt-4">
    <div class="col-md-6">
      <div class="feature-card">
        <i class="feature-icon bi bi-bar-chart"></i>
        <h5>Estadísticas de Rendimiento</h5>
        <p>Ve tus estadísticas de rendimiento para tener una mejor bitácora de tu eficiencia al usar nuestra
aplicación</p>
      </div>
    </div>
  </div>
</div>

```

```

        <div class="feature-card">
            <i class="feature-icon bi bi-alarm"></i>
            <h5>Deadline</h5>
            <p>Activa el modo "deadline" para bloquear todas las aplicaciones que no sean necesarias para realizar tu
tarea</p>
        </div>
    </div>
</div>

<div class="container features-section">
    <h2 class="features-title">Panes disponibles</h2>
    <p>Ofrecemos tres modalidades distintas para usar Panopticon</p>
    <div class="row mt-4">
        <div class="col-md-4">
            <div class="feature-card">
                <i class="feature-icon fas fa-bug"></i>
                <h5>Pepe Grillo</h5>
                <p>Ideal para usuarios que solo requieren un guia simple que los ayude a realizar sus tareas</p>
            </div>
        </div>
        <div class="col-md-4">
            <div class="feature-card">
                <i class="feature-icon fas fa-user-ninja"></i>
                <h5>Sensei</h5>
                <p>Para aquellos usuarios que requieran una mayor restriccion para las distracciones</p>
            </div>
        </div>
        <div class="col-md-4">
            <div class="feature-card">
                <i class="feature-icon fas fa-eye"></i>
                <h5>Gran Hermano</h5>
                <p>Pensado para empresas que busquen un mayor rendimiento de sus empleados en horas laborales</p>
            </div>
        </div>
    </div>
</div>

<!-- Footer -->
<footer>
    <p>&copy; 2024 Panopticon. Todos los derechos reservados.</p>
</footer>

<!-- Enlace a Bootstrap JS -->

```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>  
</body>  
</html>
```

Comprobamos el correcto funcionamiento de las rutas y endpoints asociados a la aplicación. En primer lugar, se verificaron las peticiones GET a las rutas de navegación principales, como la página de inicio (/), productos (/productos/), contacto (/contacto/), y comentarios (/comentarios/), asegurando que todas respondieran correctamente con un código de estado 200.

Además, se revisó la funcionalidad del inicio de sesión mediante la comprobación de la ruta /login/, verificando que la autenticación del usuario sea procesada adecuadamente a través de una solicitud POST y que la respuesta incluya un token de autenticación válido. Esto garantiza que los usuarios puedan iniciar sesión correctamente, lo que es un aspecto crítico de la funcionalidad de la página.

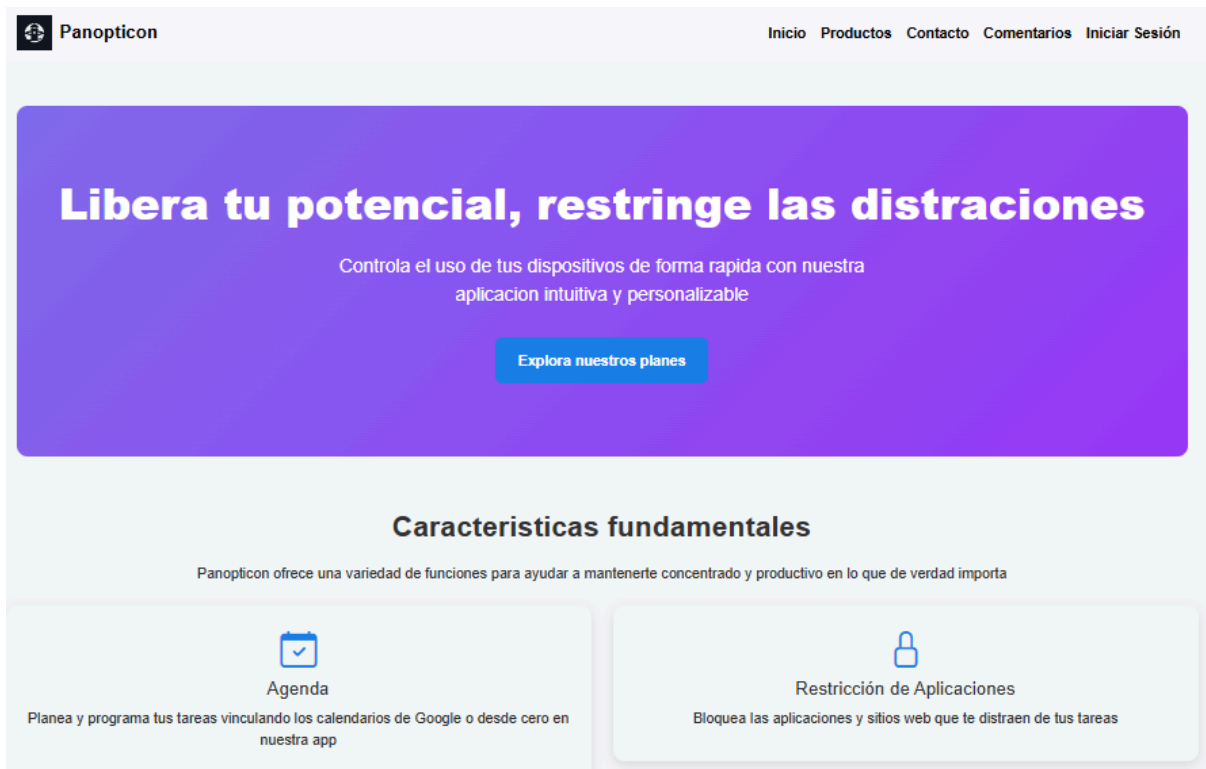
Para las secciones de productos y planes disponibles, se verificó que la información dinámica se cargue correctamente mediante las solicitudes GET a los endpoints correspondientes, asegurando que los datos se muestren sin errores. También se validó que la interacción con el botón de "Explora nuestros planes" redirigiera al usuario a la página de productos sin inconvenientes.

-2. Las rutas definidas en la página web, como se muestra en la imagen, confirman que todas las funcionalidades clave están correctamente implementadas y accesibles. Cada patrón de URL está asociado a una operación específica, como el manejo del carrito, la autenticación de usuarios y la gestión de archivos, asegurando que estas funciones se ejecuten correctamente. Además, la configuración de rutas para archivos estáticos y dinámicos garantiza que la página maneje contenido multimedia de manera efectiva.

Using the URLconf defined in `Panopticon.urls`, Django tried these URL patterns, in this order:

```
1. admin/
2. [name='pagina_inicio']
3. registro/ [name='registro']
4. login/ [name='login']
5. logout/ [name='logout']
6. productos/ [name='productos']
7. agregar_al_carrito/<int:producto_id>/ [name='agregar_al_carrito']
8. ver_carrito/ [name='ver_carrito']
9. actualizar_carrito/ [name='actualizar_carrito']
10. eliminar_del_carrito/<int:item_id>/ [name='eliminar_del_carrito']
11. contacto/ [name='contacto']
12. comentarios/ [name='comentarios']
13. checkout/ [name='checkout']
14. pago_exitoso/ [name='pago_exitoso']
15. mis_planes/ [name='mis_planes']
16. descargar_txt/<int:pedido_id>/ [name='descargar_archivo_txt']
17. descargar_exe/<int:pedido_id>/ [name='descargar_archivo_exe']
18. facebook/ [name='mostrar_publicacion']
19. publicar/ [name='publicar']
20. programar/ [name='programar']
21. subir_imagen/ [name='subir_imagen']
22. eliminar/ [name='eliminar']
23. ^media/(?P<path>.*)$
```

-3.La página web se desempeña de manera óptima, asegurando que todas las opciones interactivas disponibles para el usuario funcionen correctamente en todo momento. Cada componente, desde los botones hasta los formularios y enlaces, responde de forma adecuada y eficiente, garantizando que los usuarios puedan interactuar con la plataforma sin experimentar fallos. Además, todas las opciones de selección se presentan de manera clara y accesible, lo que permite una navegación fluida y sin interrupciones. La implementación técnica ha sido diseñada para ofrecer una experiencia de usuario intuitiva, donde las funciones se despliegan de manera exitosa y se completan con éxito todas las acciones que el usuario emprende, lo que contribuye a una experiencia satisfactoria y profesional.



Welcome Back to Panopticon

Login to your account

Username

diegobadillo

Password

.....

Log In

New to Panopticon? [Sign up here](#)

Contáctanos

¿Tienes preguntas? Envíanos un mensaje y te responderemos lo antes posible.

Nombre

Correo Electrónico

Asunto

Mensaje

Enviar Mensaje

Inicio

Productos

Contacto

Comentarios


Mis Planes

Carrito

Publicaciones

Cerrar Sesión

Mis Planes Comprados



Aún no ha comprado nada.

© 2024 Panopticon. Todos los derechos reservados.

62

Conclusiones y Trabajos Futuro

Conclusión

El desarrollo de la aplicación "Panopticon" ha permitido abordar de manera innovadora y efectiva uno de los desafíos más significativos de la era digital contemporánea: la gestión de las distracciones tecnológicas. En un contexto donde las plataformas de entretenimiento, redes sociales y notificaciones constantes juegan un papel fundamental en la dispersión de la atención, "Panopticon" ofrece una solución integral que facilita la optimización del tiempo y la concentración en tareas de alta prioridad.

La aplicación ha sido diseñada con un enfoque centrado en el usuario, proporcionando herramientas personalizables que permiten la creación de agendas adaptadas a las necesidades individuales, con la capacidad de restringir el acceso a aplicaciones y páginas web no esenciales. Funcionalidades avanzadas como la herramienta "DEADLINE", que bloquea automáticamente las distracciones cercanas a una fecha de entrega, y la opción "LADRILLO", que convierte el dispositivo en una herramienta básica para evitar la tentación de aplicaciones distractoras, destacan por su capacidad para maximizar el rendimiento personal en contextos de alta demanda, como el ámbito académico o profesional.

Además, la integración de plataformas externas como Google Classroom y Microsoft Teams refuerza la capacidad de la aplicación para sincronizarse con las herramientas de trabajo y estudio más utilizadas, mejorando la organización y ejecución de tareas. La funcionalidad para gestionar publicaciones en redes sociales, como Twitter y Facebook, es otra característica que amplía el alcance de la plataforma, permitiendo a los usuarios no solo gestionar su tiempo, sino también interactuar de manera efectiva en plataformas digitales.

Desde el punto de vista técnico, el proyecto ha demostrado un sólido diseño arquitectónico, utilizando tecnologías modernas como Django, que aseguran una implementación escalable y de alto rendimiento. El uso de un entorno virtual para el desarrollo garantiza que el sistema sea modular, fácil de mantener y actualizar, lo que permite la incorporación de nuevas funcionalidades sin comprometer el funcionamiento existente.

No obstante, a pesar de los logros alcanzados, el proyecto aún presenta áreas de oportunidad para su evolución. En el futuro, sería conveniente explorar la posibilidad de integrar algoritmos de inteligencia artificial para ofrecer recomendaciones personalizadas sobre la

gestión del tiempo o la implementación de un sistema de análisis de productividad basado en el comportamiento del usuario. Además, la expansión de la compatibilidad con más plataformas y dispositivos aumentaría la accesibilidad de la aplicación y mejoraría su competitividad en el mercado.

Trabajos Futuros

A pesar de que "Panopticon" ha logrado cumplir con sus objetivos fundamentales de optimización del tiempo y reducción de distracciones, el entorno digital y las necesidades de los usuarios continúan evolucionando. Por lo tanto, existen diversas áreas de mejora y expansión que podrían ser exploradas en futuros trabajos para hacer la aplicación aún más robusta y funcional. A continuación, se detallan algunas líneas de desarrollo futuro:

1. Integración con Nuevas Plataformas y Servicios

Uno de los próximos pasos podría ser la ampliación de la integración de "Panopticon" con otras plataformas de trabajo y productividad utilizadas globalmente. Además de Google Classroom y Microsoft Teams, se podrían incorporar herramientas como Trello, Slack, Asana y Notion, que son ampliamente usadas tanto en entornos académicos como profesionales. Esta integración permitiría una gestión más fluida de las tareas y una mejor sincronización entre las diferentes herramientas que el usuario ya utiliza.

2. Implementación de Inteligencia Artificial para Optimización del Tiempo

La incorporación de algoritmos de inteligencia artificial (IA) podría transformar "Panopticon" en una herramienta más personalizada y eficiente. A través de la IA, la aplicación podría aprender de los hábitos del usuario, sugiriendo horarios ideales para realizar tareas o incluso recomendando técnicas de productividad como la técnica Pomodoro basada en patrones de trabajo previos. Además, un sistema de análisis predictivo podría alertar a los usuarios sobre posibles períodos de baja productividad y sugerir ajustes en sus agendas.

3. Desarrollo de un Sistema de Retroalimentación y Mejora Continua

Incluir una funcionalidad de retroalimentación en tiempo real sería útil para mejorar la experiencia del usuario. A través de un sistema de análisis de datos, la aplicación podría proporcionar informes sobre cómo se ha gestionado el tiempo, identificando áreas de mejora y sugiriendo ajustes en las estrategias de trabajo. Además, los

usuarios podrían recibir recomendaciones para mejorar su enfoque y productividad basadas en su comportamiento pasado.

4. Ampliación de Funcionalidades para Móviles y Multiplataforma

Dado que las distracciones digitales son especialmente prevalentes en dispositivos móviles, una versión de "Panopticon" optimizada para smartphones y tabletas permitiría a los usuarios aplicar las restricciones y herramientas de gestión de tiempo de manera más accesible y en cualquier lugar. Esta versión podría incluir la integración con aplicaciones móviles populares como WhatsApp, Instagram y Facebook, bloqueando únicamente las distracciones más problemáticas mientras permite el uso de aplicaciones esenciales para el trabajo.

5. Mejoras en la Seguridad y la Privacidad de los Datos

La protección de los datos del usuario es un aspecto fundamental para cualquier plataforma que gestione información sensible. En el futuro, se podría mejorar la seguridad de la aplicación mediante la implementación de medidas como la autenticación multifactorial (MFA) y el cifrado avanzado de datos. Además, ofrecer una mayor transparencia en cuanto a la gestión de los datos personales y las políticas de privacidad podría aumentar la confianza de los usuarios en la plataforma.

6. Desarrollo de una Comunidad de Usuarios

Crear una comunidad en línea dentro de la aplicación podría fomentar la interacción entre los usuarios, permitiéndoles compartir consejos sobre productividad, estrategias de gestión del tiempo o incluso realizar comparaciones sobre su desempeño en tareas específicas. Este enfoque de colaboración podría enriquecer la experiencia de los usuarios, creando un espacio de apoyo y aprendizaje mutuo.

7. Optimización del Desempeño en Dispositivos con Bajo Rendimiento

Aunque la aplicación está diseñada para funcionar de manera eficiente, se podrían realizar mejoras adicionales para optimizar su rendimiento en dispositivos con recursos limitados, como teléfonos de gama baja o PCs con especificaciones modestas. La implementación de versiones más ligeras de la aplicación o la optimización del código podría garantizar una experiencia fluida para todos los usuarios, sin importar el dispositivo que utilicen.

