TECNÓLOGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Apizaco

TRABAJO DE RESIDENCIA PROFESIONAL

“IMPLEMENTACION DE UN MODULO DE NOTIFICACIONES VIA EMAIL Y WHATSAPP EN UN PROYECTO DE DJANGO 4.0.”

PRESENTA:

OUEN DANIEL AGUILA VASQUEZ

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

ASESOR INTERNO:
MTRA. ELIZABETH CUATECONTZI CUAHUTLE

ASESOR EXTERNO

DRA. LORENA JEANNETE ÁVILA ALVARADO

APIZACO TLAX., JUNIO 2024

Instituto Tecnológico de Apizaco

Agradecimientos

Agradezco a mi familia, en especial a mi madre y mi abuela, que estuvieron presentes en cada momento de mi carrera profesional, apoyándome y motivando a superar cada reto que se me presento a lo largo de la carrera.

A mi hijo por ser el motor de mi vida, por ser esa pequeña personita por la cual me levante todos los dias con tal de superarme y ser mejor profesionista.

De igual manera Agradezco al Instituto Tecnológico de Apizaco, por darme la oportunidad de aprender mucho de esta carrera y permitirme enamorarme día con día.

Al Dr. Juan Ramos Ramos, a la Mtra. Elizabeth Cuatecontzi Cuautle, por brindarme apoyo a lo largo de mi estancia en el instituto, asi como su gran conocimiento en esta carrera y permitirme realizar este proyecto en la recta final de mi carrera universitaria.

A la Lic. Yenissa Báez Navarrete y al Lic. Luis Fernando Santos Dávila, quienes me dieron la inspiración para no rendirme y permitirme conocer y enamorarme del idioma alemán y poder aprender más acerca de mi lengua materna.

Por ultimo a mis amigos Rene y Gary quienes me acompañaron en cada aventura, malos momentos y alegrías que pasamos juntos a lo largo de la carrera.

Resumen

El desarrollo de software tiene como objetivo la optimización de uno o varios problemas, en este proyecto se abordan los desafíos que la Clínica Dental Center’s enfrenta, en la gestión de procesos como la administración de citas y la experiencia del usuario, lo que puede llevar a una experiencia insatisfactoria para los pacientes de esta, asi como en la perdida de pacientes. Como objetivo de este proyecto es desarrollar un módulo de notificaciones para mejorar la eficiencia de procesos de citas y mejorar la experiencia del usuario al utilizar la aplicación.

Este módulo puede mejorar la experiencia del usuario y aumentar el tiempo de utilización de la aplicación. Dentro de la metodología incluye análisis de requerimientos, diseño del módulo, desarrollo, pruebas, implementación y capacitación. Los resultados muestran un mayor tráfico de usuarios, mayor comunicación con los pacientes y una mayor satisfacción de los usuarios.

Índice

Capítulo 1 Generalidades del Proyecto

1.1 Introducción

Hoy en día la comunicación eficiente y oportuna es esencial para el éxito de cualquier aplicación web. Dentro de la prestación de servicios de Odontología, gestionar de manera efectiva y sistemática las citas juega un papel crucial en la atención al paciente. Este proyecto, titulado "IMPLEMENTACION DE UN MODULO DE NOTIFICACIONES VIA EMAIL Y WHATSAPP EN UN PROYECTO DE DJANGO 4.0.", busca mejorar la experiencia del usuario y aumentar el tráfico de usuarios en la aplicación

El objetivo principal de este proyecto es desarrollar e implementar un módulo que permita el envio de notificaciones a través de dos de los medios de comunicación más utilizados en la actualidad: el correo electrónico y WhatsApp.

Utilizar el correo electrónico como canal de notificación es ampliamente aceptado y utilizado en diversas aplicaciones para enviar alertas, confirmaciones y actualizaciones a los usuarios. Por otro lado, WhatsApp ha emergido como una plataforma de mensajería instantánea que ofrece una alta tasa de apertura y respuesta, convirtiéndose es una herramienta para una comunicación más directa e inmediata.

1.2 Descripción de la empresa u organización y del puesto o área del trabajo el estudiante.

Clínica Dental Center's tiene como misión brindar atención dental de calidad y un servicio amigable y personalizado a todas aquellas personas que confían su mejor expresión "SU SONRISA" con un grupo de profesionales de reconocido prestigio con más de 20 años de experiencia profesional.

Su objetivo es brindar Calidez Humana y Satisfacción a sus pacientes proporcionándoles una atención en horarios cómodos a través de citas programadas.

Además de los dentistas con experiencia, cuentan con higienistas, asistentes y personal administrativo dedicado a brindar un servicio completo, Clínica Dental Center´s utiliza equipos y técnicas avanzadas para brindar tratamientos precisos y cómodos para los pacientes.

La atención individualizada que brindan a cada paciente demuestra su enfoque en la calidez humana.

La visión de la Clínica Dental Center´s es ser un referente dentro del sector odontológico para la realización de una odontología de excelencia, calidad y mejora continua para la satisfacción de los pacientes, profesionales, colaboradores y proveedores.

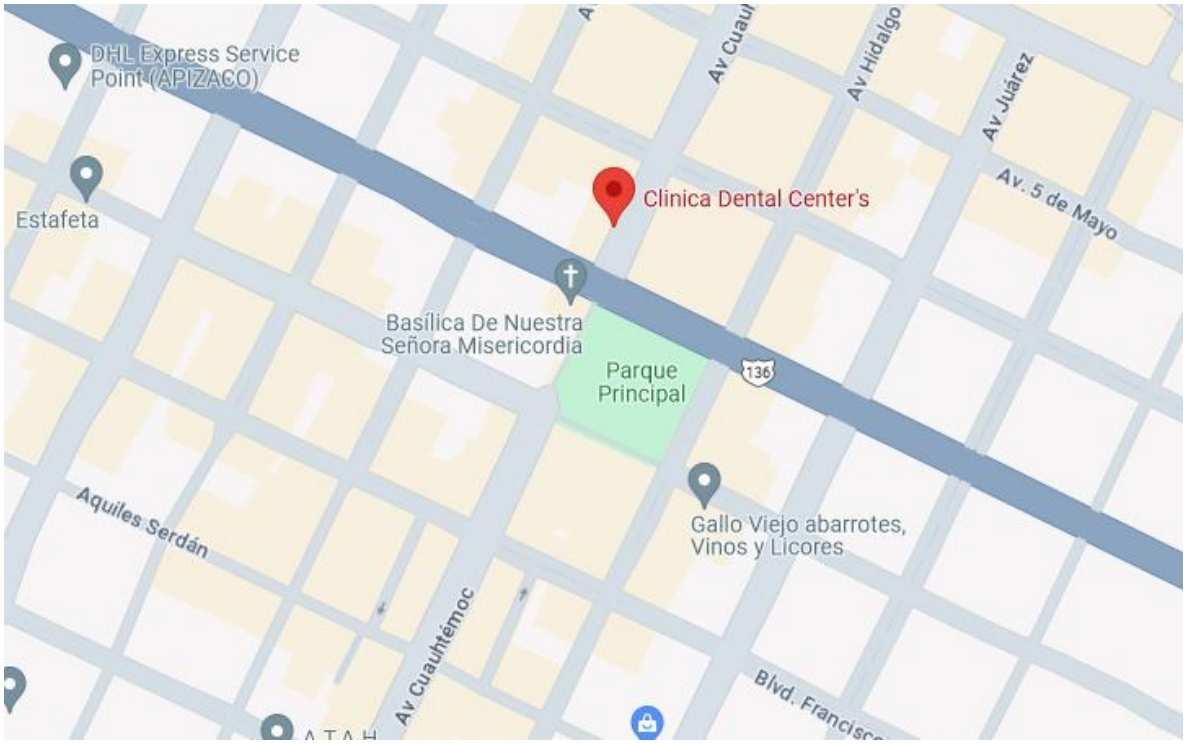
La clínica cuenta con diferentes especialidades Odontológicas:

- Odontología General: Tratamientos preventivos, limpiezas, obturaciones y extracciones.
- Ortodoncia: Corrección de la alineación dental con Brackets o alineadores transparentes.
- Implantes Dentales: Colocación de implantes para reemplazar dientes perdidos.
- Estética Dental: Blanqueamiento, carillas y restauraciones estéticas.
- Periodoncia: Tratamiento de enfermedades de las encías.
- Endodoncia: Tratamiento de conductos radiculares.
- Cirugía Oral: Extracciones complejas y otros procedimientos quirúrgicos.

Ubicación:

Av. Cuauhtémoc 206-Interior 3, Colonia Centro, 90300 Apizaco, Tlax., México

Google Maps:



Logotipo de la empresa:

1.3Problemas por resolver

Actualmente, el proyecto de la Clínica Dental Center's carece de un sistema de notificaciones que permita a los usuarios estar informados sobre sus citas, historial clínico o cambios de citas. Esto puede generar una experiencia de usuario deficiente y una falta de engagement por parte de estos.

1.4 Objetivos

General

Desarrollar e implementar un módulo de notificaciones vía email y WhatsApp para el proyecto Django 4.0, que permita a los usuarios estar informados de manera eficiente, personalizada y oportuna sobre eventos, actividades o cambios relevantes dentro de la plataforma.

Específicos:

- Diseñar e implementar un sistema de notificaciones escalable y flexible que pueda adaptarse a las necesidades cambiantes del proyecto.

- Integrar canales de comunicación adicionales, como WhatsApp, para ofrecer a los usuarios una mayor variedad de opciones para recibir notificaciones.

- Implementar un sistema de personalización de notificaciones que permita a los usuarios elegir la frecuencia, el formato y el contenido de las notificaciones que reciben.

- Integrar el sistema de notificaciones con otros sistemas de la plataforma para automatizar el envío de notificaciones y la gestión de datos.

1.5 Justificación

La implementación de un módulo de notificaciones vía email y WhatsApp para el proyecto Django 4.0 se justifica por los siguientes motivos:

- Mejorar la experiencia del usuario: Un sistema de notificaciones eficiente y personalizado permitirá a los usuarios estar informados sobre eventos, actividades o cambios relevantes dentro de la plataforma, lo que mejorará su experiencia general y aumentará su engagement.

- Aumentar la retención de usuarios: Al mantener a los usuarios informados, se puede aumentar la probabilidad de que continúen utilizando la plataforma, lo que se traduce en una mayor retención de usuarios.

- Optimizar la comunicación: La integración de canales de comunicación adicionales, como WhatsApp, permitirá a la plataforma comunicarse con los usuarios de manera más efectiva y llegar a una audiencia más amplia.

- Automatizar procesos: La integración del sistema de notificaciones con otros sistemas de la plataforma permitirá automatizar el envío de notificaciones y la gestión de datos, lo que optimizará el tiempo y los recursos.

Capítulo 2 Marco teórico

2.1 Herramientas Tecnológicas

2.1.1 Django

¿Qué es Django?

En Django (2024a) se describe a esta herramienta como un framework de desarrollo web de código abierto escrito en Python que sigue el patrón de diseño MVC (Modelo-Vista-Controlador). Se caracteriza por su enfoque rápido, seguro y escalable para crear aplicaciones web complejas. Otras características son:

- **Potente sistema de plantillas:** Django facilita la creación de interfaces web dinámicas con plantillas HTML personalizadas.
- **ORM intuitivo:** El Object-Relational Mapping (ORM) de Django permite mapear objetos Python a tablas de bases de datos de forma intuitiva.
- **Seguridad robusta:** Django incorpora características de seguridad predefinidas para proteger las aplicaciones web contra ataques comunes.
- **Escalabilidad:** Django está diseñado para manejar grandes cantidades de tráfico y usuarios.
- **Amplia comunidad:** Cuenta con una comunidad activa y vibrante que ofrece soporte y recursos.

Y tiene las siguientes ventajas:

- **Acelera el desarrollo:** Django ofrece herramientas y utilidades que agilizan el proceso de desarrollo.
- **Código limpio y mantenible:** Promueve la escritura de código Python limpio, organizado y fácil de mantener.
- **Entorno seguro:** Las características de seguridad integradas protegen las aplicaciones web contra vulnerabilidades.
- **Escalabilidad horizontal:** Permite distribuir la carga de trabajo en varios servidores para mejorar el rendimiento.
- **Amplia biblioteca de terceros:** Existe una gran cantidad de módulos y paquetes adicionales disponibles para ampliar las funcionalidades de Django.

Sin embargo, se identifican algunas desventajas:

- **Curva de aprendizaje:** Puede ser complejo para principiantes en Python o frameworks web.
- **Menos flexible que otros frameworks:** Django tiene una estructura predefinida que puede limitar la flexibilidad en algunos casos.
- **Overhead:** Puede ser más pesado que otros frameworks para proyectos pequeños.

2.1.2 Git Hub

Richardson (2022)menciona lo siguiente, GitHub es una plataforma de alojamiento de código en línea diseñada específicamente para proyectos que utilizan el sistema de control de versiones Git. Git permite a los desarrolladores rastrear cambios en el código fuente, colaborar en proyectos y revertir cambios si es necesario. GitHub, por otro lado, proporciona una interfaz gráfica de usuario (GUI) amigable y un conjunto de características que facilitan el trabajo con Git.

Funcionalidades clave de GitHub:

Almacenamiento de código: Almacena de forma segura el código fuente de tus proyectos en repositorios remotos.

Control de versiones: Permite realizar un seguimiento de los cambios realizados en el código a lo largo del tiempo.

Colaboración: Facilita la colaboración entre desarrolladores en proyectos compartidos.

Seguimiento de errores: Brinda herramientas para reportar y rastrear errores en el código.

Gestión de proyectos: Ofrece herramientas básicas para la gestión de proyectos de software.

Integración continua (CI) y entrega continua (CD): Permite automatizar la construcción, prueba e implementación de código.

Comunidad: Cuenta con una amplia comunidad de desarrolladores que comparten código e interactúan entre sí.

2.1.3 Visual Studio Code

Visual Studio Code (2021) es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Es compatible con múltiples sistemas operativos como Windows, macOS y Linux. VS Code se caracteriza por ser ligero, personalizable y extensible, convirtiéndose en una herramienta popular para el desarrollo de software.

Se rescataron las siguientes características:

- **Soporte para múltiples lenguajes:** VS Code ofrece soporte integrado para una amplia variedad de lenguajes de programación, incluyendo JavaScript, Python, C++, Java, y muchos más.
- **Resaltado de sintaxis:** La sintaxis del código se resalta con diferentes colores para mejorar la legibilidad y facilitar la comprensión del código.
- **Completado inteligente:** VS Code sugiere automáticamente palabras clave, funciones y variables mientras se escribe el código, acelerando el proceso de desarrollo.
- **Depuración integrada:** Permite depurar código directamente en el editor, identificando y solucionando errores con mayor facilidad.
- **Control de versiones integrado:** Se puede integrar con sistemas de control de versiones como Git, permitiendo gestionar los cambios en el código fuente.
- **Extensible:** VS Code cuenta con un amplio ecosistema de extensiones que añaden funcionalidades adicionales, como soporte para nuevos lenguajes, herramientas de desarrollo específicas y temas personalizados.

2.1.4 Python

Según Python (2024), Python es un lenguaje de programación de alto nivel interpretado, de código abierto y propósito general. Se caracteriza por su sintaxis simple y legible, lo que lo convierte en un lenguaje fácil de

aprender y usar. Python es ampliamente utilizado en diversas áreas del desarrollo de software, como:

- **Desarrollo web:** Frameworks web populares como Django y Flask están basados en Python.
- **Ciencia de datos:** Bibliotecas como NumPy, Pandas y Scikit-learn hacen de Python una herramienta ideal para el análisis y manipulación de datos.
- **Scripting:** Python se emplea frecuentemente para automatizar tareas y crear scripts para diversas aplicaciones.
- **Desarrollo de escritorio:** PyQt y wxPython son bibliotecas que permiten crear aplicaciones de escritorio con interfaces gráficas de usuario (GUI) en Python.

2.1.5 SQLite

Según lo expresado en SQLite (2023). Es una base de datos relacional liviana y autónoma que se incluye en la mayoría de los sistemas operativos móviles y de escritorio. Es una de las bases de datos más populares del mundo, que se utiliza en una amplia gama de aplicaciones, desde aplicaciones móviles hasta sitios web y software de escritorio.

Para comprender mejor se rescataron las siguientes características:

- **Ligera:** SQLite tiene una huella de memoria pequeña, lo que la hace ideal para dispositivos embebidos y aplicaciones con recursos limitados.
- **Autónoma:** SQLite no requiere un servidor de base de datos separado, lo que la hace fácil de implementar y usar.
- **Compatible con SQL:** SQLite cumple con el estándar SQL92, lo que facilita a los desarrolladores aprender y usar.
- **Extensible:** SQLite admite extensiones que amplían su funcionalidad.
- **Gratis y de código abierto:** SQLite es gratuita y de código abierto, lo que la convierte en una opción popular para proyectos de software de código abierto y comerciales.

Se muestran algunos casos de uso como:

- **Aplicaciones móviles:** SQLite es una base de datos popular para aplicaciones móviles debido a su tamaño pequeño y rendimiento rápido.

- **Sitios web:** SQLite se puede utilizar para almacenar datos para sitios web, como listas de productos, información de clientes y datos de pedidos.
- **Software de escritorio:** SQLite se puede utilizar para almacenar datos para software de escritorio, como listas de contactos, historiales de chat y configuraciones de usuario.
- **Dispositivos embebidos:** SQLite se puede utilizar para almacenar datos en dispositivos embebidos, como sistemas de control de temperatura y dispositivos GPS.

2.1.6 SQLiteStudio

SQLiteStudio es una interfaz gráfica de usuario (GUI) gratuita y de código abierto para bases de datos SQLite. Está disponible para Windows, macOS y Linux SQLiteStudio (2024).

Se caracteriza por:

- **Navegador de bases de datos:** SQLiteStudio proporciona un navegador de bases de datos que le permite ver y editar los datos en sus bases de datos SQLite.
- **Editor SQL:** SQLiteStudio incluye un editor SQL que le permite ejecutar consultas SQL en sus bases de datos SQLite.
- **Herramientas de gestión de bases de datos:** SQLiteStudio proporciona una variedad de herramientas para administrar sus bases de datos SQLite, como herramientas para crear y eliminar bases de datos, tablas y vistas.
- **Portátil:** SQLiteStudio es portátil, lo que significa que no necesita instalarlo en su computadora. Simplemente puede descargarlo y ejecutarlo desde una unidad USB.

Tiene los siguientes casos de uso:

- **Desarrolladores:** SQLiteStudio es una herramienta útil para desarrolladores que trabajan con bases de datos SQLite. Les permite ver y editar fácilmente los datos en sus bases de datos SQLite, ejecutar consultas SQL y administrar sus bases de datos SQLite.
- **Administradores de bases de datos:** SQLiteStudio es una herramienta útil para administradores de bases de datos que necesitan administrar bases de datos SQLite. Les permite ver y editar fácilmente los datos en sus bases de datos SQLite, ejecutar consultas SQL y realizar tareas de administración de bases de datos comunes.
- **Usuarios finales:** SQLiteStudio es una herramienta útil para usuarios finales que necesitan trabajar con bases de datos SQLite.

Les permite ver y editar fácilmente los datos en sus bases de datos SQLite y ejecutar consultas SQL simples.

2.1.7 Django's Built-in Email System

En palabras de Django (2024b). Existe un sistema de correo electrónico integrado de Django el cual ocurre a través del módulo smtplib, esta función es potente y flexible

Se identifican las siguientes características

Integración Directa: Viene incluido con Django y no requiere instalación adicional

Configuración en 'setting.py': Puede definir servidores SMTP, credenciales, y configuraciones de correo electrónico en un solo lugar.

Métodos Simples: Ofrece métodos como 'send_mail' y 'EmailMessage' para enviar correos electrónicos.

Cuenta con las siguientes ventajas

Facil de Usar: Ideal para aplicaciones básicas que necesitan enviar correos electrónicos

Configuración Rápida: La configuración es directa y no requiere dependencias adicionales.

Soporte Integrado: Soporte y documentación de Django.

Lamentablemente cuenta con algunas desventajas como:

Funcionalidades Limitadas: No es ideal para las aplicaciones con necesidades avanzadas como plantillas de correo complejas o seguimiento de correos.

Escalabilidad: Puede no ser la mejor opción para grandes volúmenes de correos o necesidades específicas de entrega.

2.1.8 Twilio

De acuerdo con Twilio (2024), es una plataforma de comunicaciones en la nube que ofrece una variedad de servicios y APIs para integrar capacidades de comunicación como voz, video, y mensajería en aplicaciones y servicios web.

Muestra las siguientes características:

API Completa: Twilio ofrece una API robusta para enviar mensajes SMS y a través de WhatsApp

Documentación y Soporte: Buena documentación y soporte para integrar con varias plataformas

Entre los beneficios mas destacados se encuentran:

Flexibilidad: Ofrece funciones avanzadas como mensajes multimedia, plantillas de mensajes y seguimiento.

Escalabilidad: Escalable para grandes volúmenes de mensajes y facil de integrar con otros servicios de Twilio.

Facil Integración: Buen soporte y documentación para integrar en Django.

Es importante considerar las siguientes desventajas:

Costo: Puede ser costoso, especialmente para grandes volúmenes de mensajes o para ciertas funcionalidades avanzadas.

Dependencia Externa: Requiere una cuenta de Twilio y puede tener limitaciones basadas en el plan elegido.

2.2 Notificaciones en aplicaciones web

2.2.1 Concepto y tipos de notificaciones

Tal como afirma Frizbit (2021). Las notificaciones web son mensajes emergentes que aparecen en la pantalla del usuario, incluso cuando no está interactuando directamente con la página web. Estas notificaciones

pueden ser utilizadas para informar al usuario sobre eventos importantes, actualizaciones, mensajes nuevos o cualquier otra información relevante.

Existen diferentes tipos de notificaciones web, entre las más comunes se encuentran:

- Notificaciones push: Son mensajes enviados por el servidor de la aplicación web al dispositivo del usuario, incluso cuando este no está navegando por la página. Este tipo de notificaciones generalmente se utilizan para informar al usuario sobre eventos urgentes o importantes.

- Notificaciones in-app: Son mensajes que aparecen dentro de la aplicación web, cuando el usuario está navegando por ella. Este tipo de notificaciones se utilizan para informar al usuario sobre eventos o información relacionada con su actividad actual dentro de la aplicación.

- Notificaciones de banner: Son mensajes que aparecen en la parte superior o inferior de la pantalla del usuario, ocupando una pequeña porción del espacio visible. Este tipo de notificaciones se utilizan para informar al usuario sobre información menos urgente o para ofrecerle acciones rápidas.

- Notificaciones de barra lateral: Son mensajes que aparecen en la barra lateral de la aplicación web, ocupando un espacio vertical a lo largo de la pantalla. Este tipo de notificaciones se utilizan para mostrar información contextual o para ofrecer al usuario opciones de navegación adicionales.

La notificaciones web juegan un papel de suma importancia en la experiencia del usuario, ya que pueden mejorar la comunicación, promover acciones específicas y mejorar la experiencia de cada usuario.

2.2.2 Canales de comunicación para notificaciones.

Las notificaciones web pueden enviarse a través de diferentes canales de comunicación, como:

- Correo electrónico: El correo electrónico es un canal tradicional que puede ser utilizado para enviar notificaciones largas y detalladas.
- SMS: Los SMS son mensajes de texto cortos que pueden ser utilizados para enviar notificaciones urgentes o importantes.
- Mensajería instantánea: Las aplicaciones de mensajería instantánea, como WhatsApp o Telegram, pueden ser utilizadas para enviar notificaciones personalizadas y conversacionales.
- Notificaciones push: Las notificaciones push son mensajes enviados directamente al dispositivo del usuario, incluso cuando no está utilizando activamente la aplicación.

2.3 Django y el envío de notificaciones

2.3.1 Funcionalidades de Django para el envío de emails

Django proporciona un framework robusto para el envío de emails mediante la librería django.core.mail. Esta librería ofrece las siguientes funcionalidades:

Composición de emails: Permite crear mensajes de email con texto plano o HTML, adjuntar archivos y establecer encabezados personalizados.

Backend de email: Soporta diferentes backends para el envío de emails, incluyendo SMTP, Sendgrid, Mailgun y otros.

Plantillas de email: Permite utilizar plantillas HTML para crear emails dinámicos y personalizados.

Cola de mensajes: Se puede configurar una cola de mensajes para enviar emails de forma asíncrona.

2.3.2 Integración de librerías externas para el envío de notificaciones por SMS o mensajería instantánea

Si se requieren notificaciones por SMS o mensajería instantánea, se pueden integrar librerías externas a Django. Algunas opciones populares incluyen:

- Twilio: Ofrece una API para enviar SMS, mensajes de voz y realizar llamadas telefónicas.
- Vonage (2024): Similar a Twilio, proporciona servicios de SMS, voz y llamadas.
- Pusher (2024): Se especializa en notificaciones en tiempo real para aplicaciones web y móviles.
- Firebase (2023): Permite enviar notificaciones push a dispositivos Android e iOS.

La integración de estas librerías generalmente implica instalar la librería correspondiente, configurar las credenciales de la cuenta del servicio y crear código para enviar las notificaciones.

2.3.3 Buenas prácticas para el diseño e implementación de notificaciones en Django

Al diseñar e implementar notificaciones en Django, se recomienda seguir estas buenas prácticas:

Definir claramente el propósito de la notificación: ¿Qué se quiere comunicar al usuario? ¿Cuál es la acción deseada?

Segmentar a los usuarios: Enviar notificaciones solo a los usuarios que les sean relevantes.

Personalizar el contenido de las notificaciones: Utilizar el nombre del usuario, datos de la cuenta o información contextual para hacer que las notificaciones sean más relevantes.

Respetar la frecuencia de las notificaciones: No saturar a los usuarios con demasiadas notificaciones.

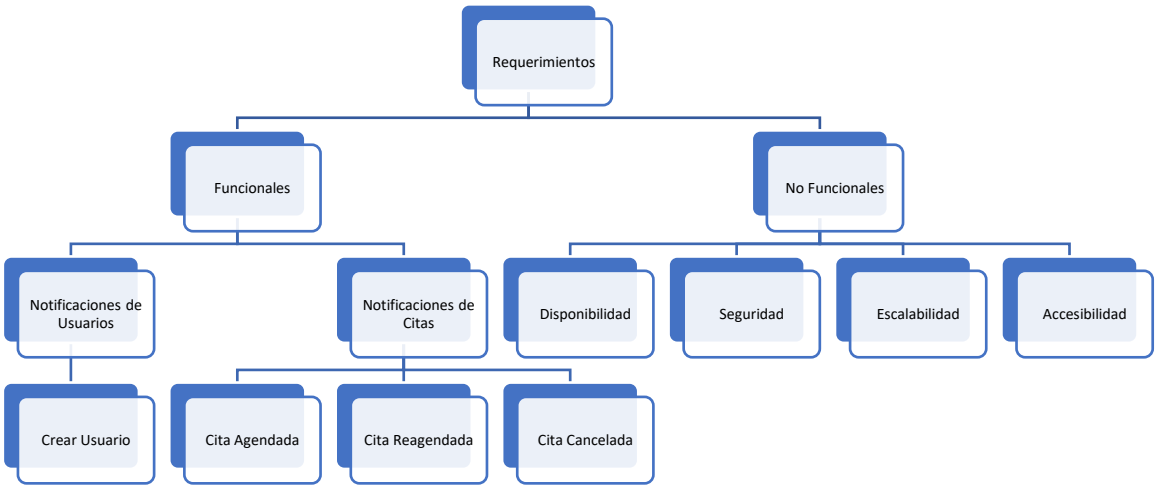
Ofrecer opciones para que los usuarios administren sus notificaciones: Permitir a los usuarios elegir qué tipos de notificaciones quieren recibir y cómo quieren recibirlas.

Monitorear y analizar el rendimiento de las notificaciones: Realizar un seguimiento de las tasas de apertura, clics y conversiones para evaluar la efectividad de las notificaciones.

Siguiendo estas buenas prácticas, se pueden crear notificaciones que sean útiles, relevantes y apreciadas por los usuarios.

Capítulo 3 Desarrollo

3.1 Requerimientos



3.1.1 Requerimientos funcionales

1. Notificaciones de Usuarios

Crear Usuario

El sistema debe enviar una notificación por correo electrónico al paciente informando sobre la creación de su usuario, junto con los datos proporcionados al sistema.

2. Notificaciones de citas

Citas agendadas:

El sistema debe enviar una notificación por correo electrónico y/o WhatsApp al paciente y al odontólogo informando sobre la cita agendada, incluyendo fecha, hora, nombre del odontólogo y paciente, y otros detalles relevantes.

Citas reprogramadas:

El sistema debe enviar una notificación por correo electrónico y/o WhatsApp al paciente y al odontólogo informando sobre la reprogramación de la cita, incluyendo la nueva fecha, hora, nombre del odontólogo y paciente, y otros detalles relevantes.

Citas canceladas:

El sistema debe enviar una notificación por correo electrónico y/o WhatsApp al paciente y al odontólogo informando sobre la cancelación de la cita, incluyendo el motivo de la cancelación (si se conoce), y otros detalles relevantes.

3.1.2 Requerimientos no funcionales

Disponibilidad: El módulo de notificaciones debe estar disponible las 24 horas del día, los 7 días de la semana.

Seguridad: El sistema debe garantizar la seguridad de las notificaciones, protegiendo los datos personales de los pacientes y odontólogos.

Escalabilidad: El sistema debe ser escalable para poder manejar un alto volumen de notificaciones.

Accesibilidad: El sistema debe ser accesible para usuarios con discapacidades.

A continuación, se mencionan las configuraciones de librerías y API seleccionada, durante esta configuración se utilizan algunos datos que se obtienen directamente de la base de datos, durante la realización de este módulo solo nos centraremos en configurarlos en codigo.

3.2 Configuración de la librería Django's Built-in Email System

Para la configuración de las notificaciones de decidió la implementación de la librería core mail, con la cual al realizar acciones específicas el sistema mandara un correo electrónico al usuario, primero se deberán realizar algunas configuraciones en el proyecto.

Esta configuración se lleva a cabo a lo largo de **6 pasos**, los cuales se muestran en las siguientes ilustraciones:

Paso 1: Se debera ir al apartado de administrar tu Cuenta de Google en cualquier navegador

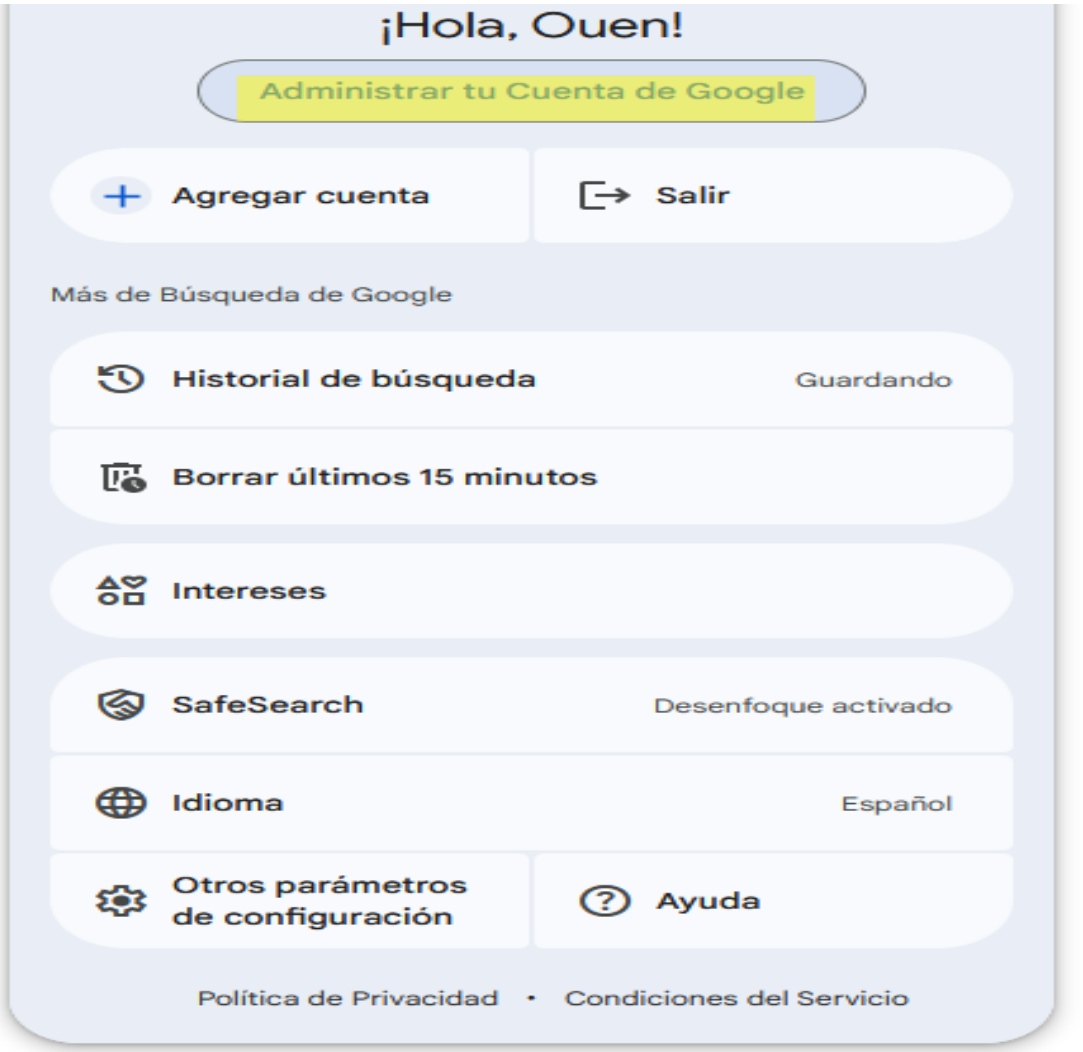


Ilustración 1 Configuración cuenta Google

Paso 2: Una vez abierta, se buscará el apartado de Contraseñas de aplicaciones

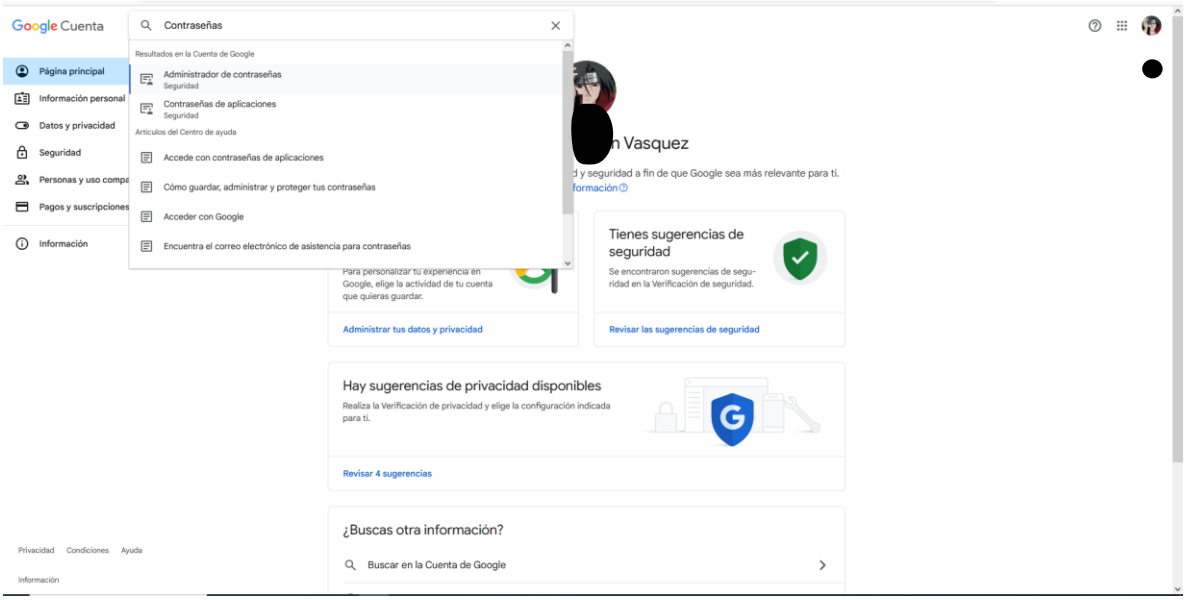


Ilustración 2 Configuración cuenta Google

Paso 3: Validar las credenciales del perfil a utilizar

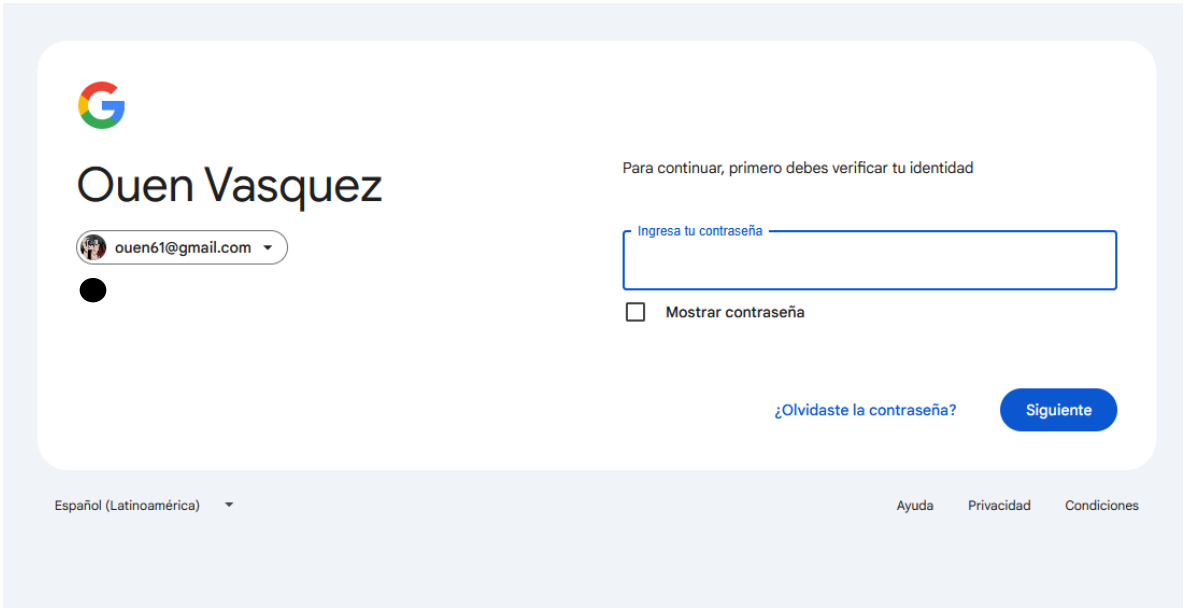


Ilustración 3 Configuración cuenta Google

Paso 4: Ingresar un nombre de la aplicación y dar clic en Crear

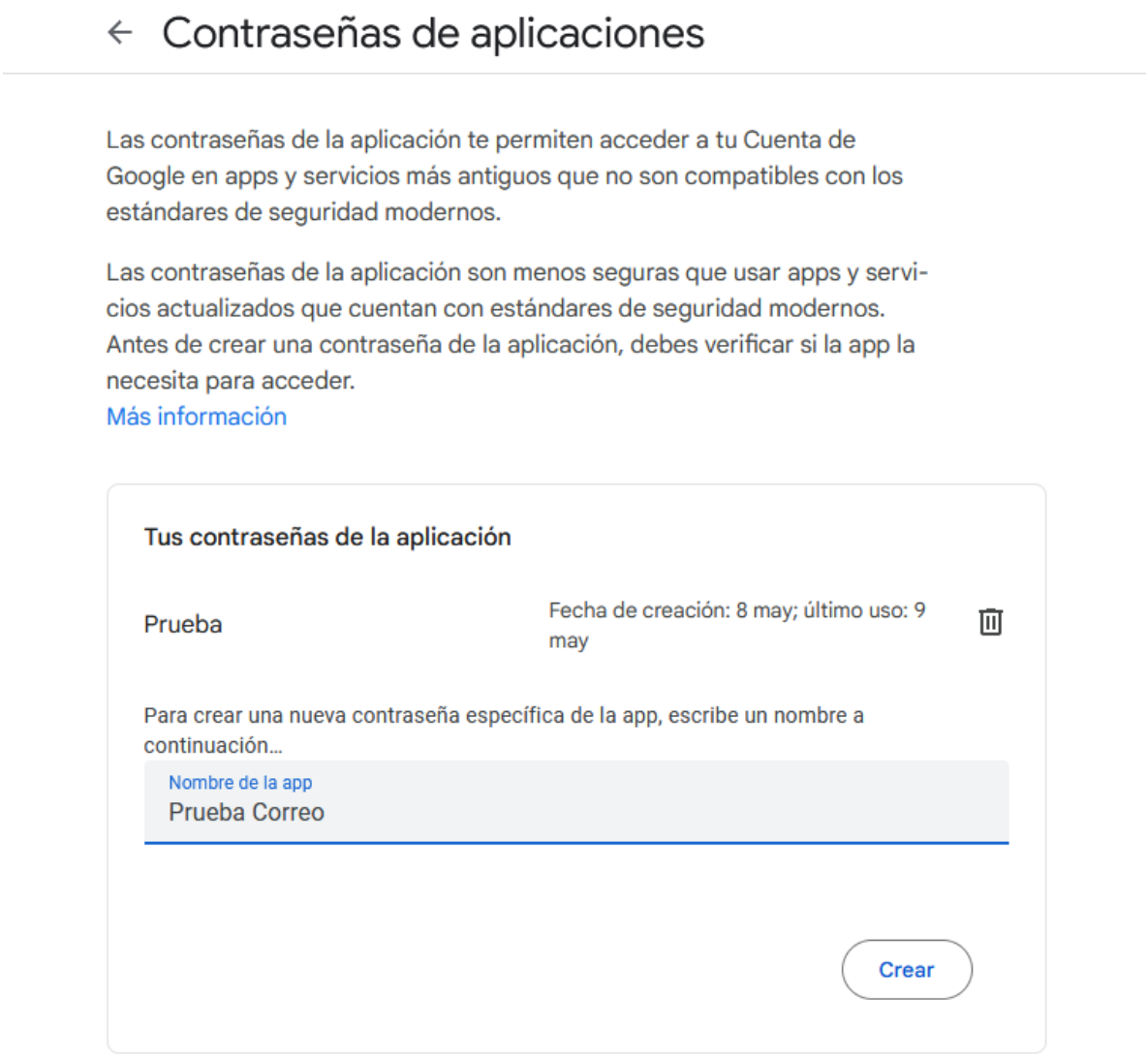


Ilustración 4 Configuración de la aplicación

Paso 5: Una vez validados los datos, mostrara la contraseña de aplicación para las configuraciones en el sistema

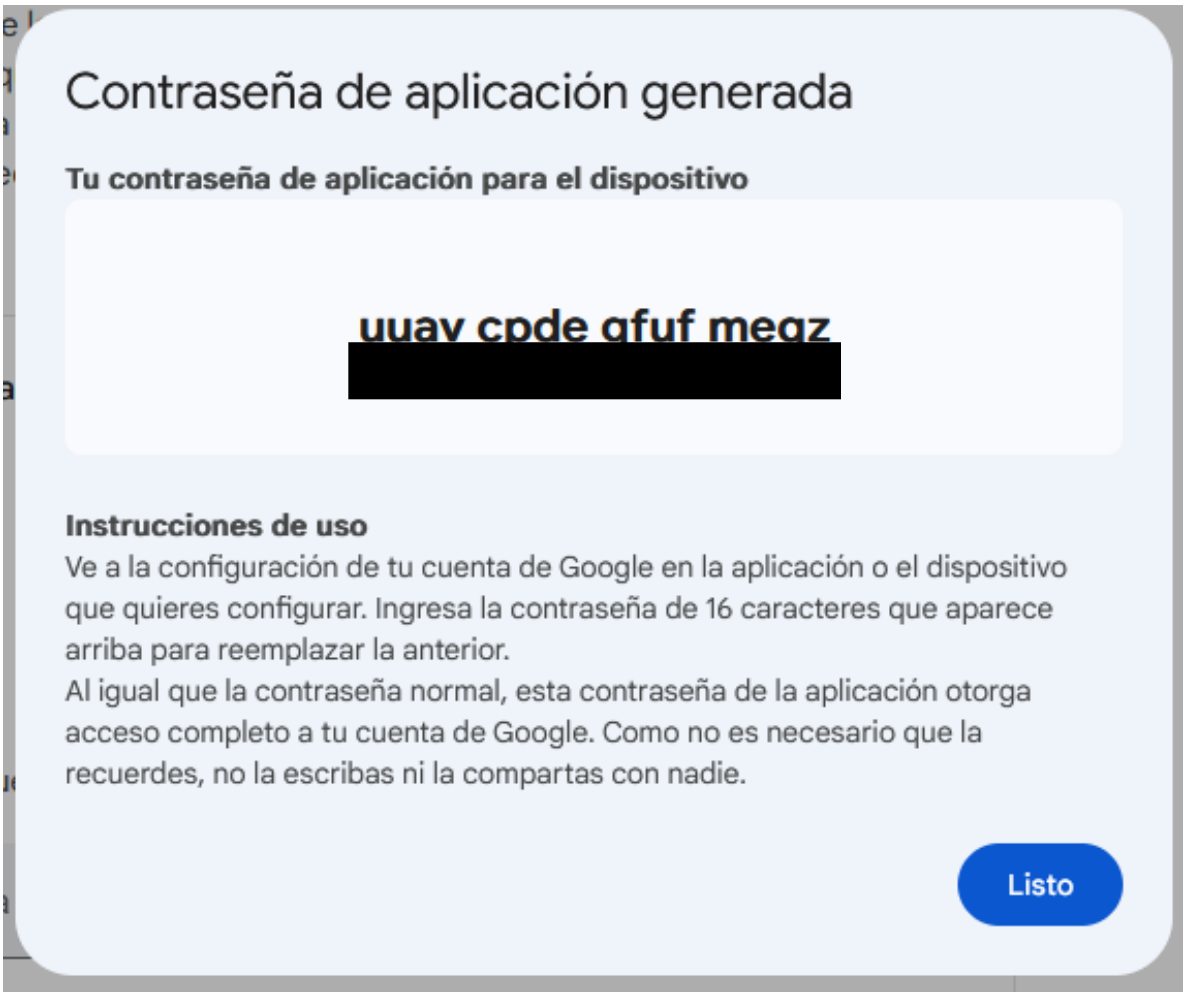


Ilustración 5 Configuración de la aplicación

Paso 6: Se agregan esas variables en este caso en el archivo setting.py del proyecto de Django en el que se trabaja

```
settings.py
Fenrir > settings.py > ...
256
257   EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
258   EMAIL_HOST = 'smtp.googlemail.com'
259   EMAIL_PORT = 587
260   EMAIL_USE_TLS = True
261   EMAIL_HOST_USER = 'ouen61@gmail.com'
262   EMAIL_HOST_PASSWORD = [redacted]
263   EMAIL_USE_LTS = True
```

Ilustración 6 Configuración del codigo del proyecto

A continuación, en base a los pasos anteriores, ahora se aplicaran unas líneas de codigo las cuales servirán para recuperar los datos del usuario y a su vez enviar el correo, se realiza la misma configuración para todos los tipos de notificación.

3.2.1 Crear Usuario

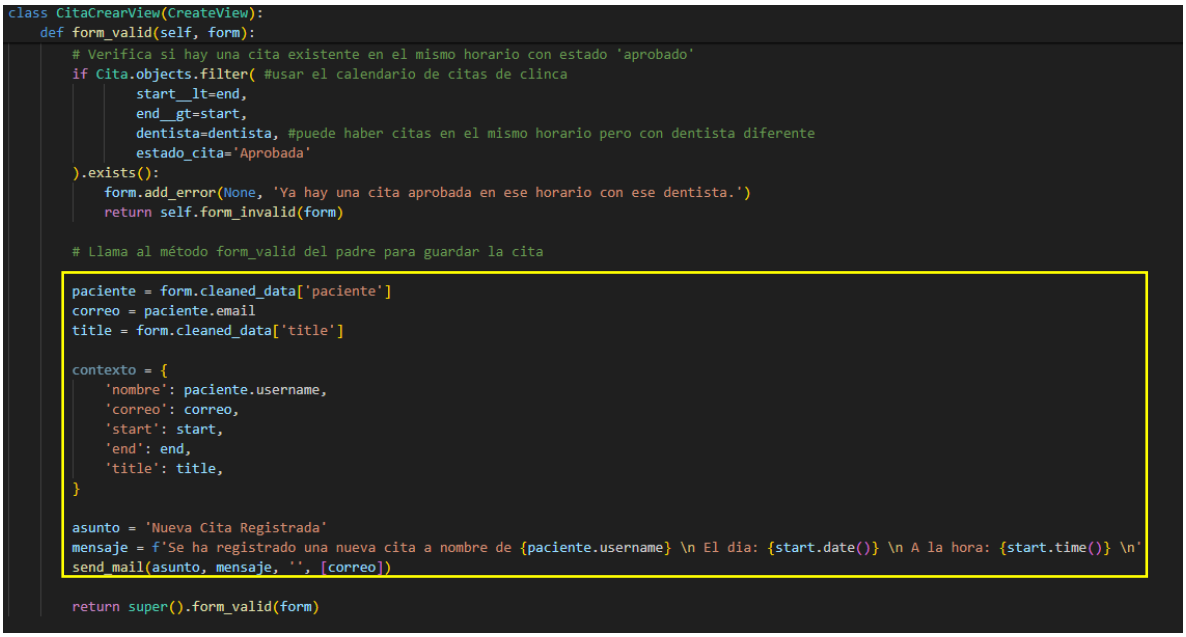
Como primer apartado, al crear un nuevo usuario el sistema mandara un correo electrónico, para esta configuración, mediante las vistas, se agregan, los datos que llevara el correo, asi como se recuperan los datos, como el correo del usuario que servirá como destinatario, como se muestran en las siguientes imágenes la funcionalidad del mismo sistema.

```python
class UserCreateViewPaciente(CreateView):
    model = CustomUser
    form_class = CustomUserCreationFormDentista
    template_name = 'register/register_user_paciente.html'
    success_url = reverse_lazy('home')

    def form_valid(self, form):
        user = form.save(commit=False)
        user.created_by = self.request.user
        user.tipo_usuario = 'paciente'

        user.save()

        nombre_usuario = user.username
        correo = user.email
        contraseña = form.cleaned_data['password1']
        contexto = {'nombre': nombre_usuario, 'correo': correo, 'contraseña': contraseña}

        asunto = 'Nuevo Usuario Registrado'
        mensaje = f'Se ha registrado de manera correcta: {nombre_usuario}\n Correo Electronico: {correo }\n Contraseña: {contraseña}'
        send_mail(asunto, mensaje, '', [correo])
```

Ilustración 7 Codigo para enviar el correo, y la configuración de este

3.2.2 Crear Cita

Como siguiente apartado, al crear una nueva cita el sistema enviara un correo electrónico con los datos de esta, la configuración en codigo se muestra en la siguiente ilustración.



```python
class CitaCrearView(CreateView):
    def form_valid(self, form):
        # Verifica si hay una cita existente en el mismo horario con estado 'aprobado'
        if Cita.objects.filter( #usar el calendario de citas de clinca
                start__lt=end,
                end__gt=start,
                dentista=dentista, #puede haber citas en el mismo horario pero con dentista diferente
                estado_cita='Aprobada'
        ).exists():
            form.add_error(None, 'Ya hay una cita aprobada en ese horario con ese dentista.')
            return self.form_invalid(form)

        # Llama al método form_valid del padre para guardar la cita

        paciente = form.cleaned_data['paciente']
        correo = paciente.email
        title = form.cleaned_data['title']

        contexto = {
            'nombre': paciente.username,
            'correo': correo,
            'start': start,
            'end': end,
            'title': title,
        }

        asunto = 'Nueva Cita Registrada'
        mensaje = f'Se ha registrado una nueva cita a nombre de {paciente.username} \n El dia: {start.date()} \n A la hora: {start.time()} \n'
        send_mail(asunto, mensaje, '', [correo])

        return super().form_valid(form)
```

Ilustración 8 Codigo para enviar correos electrónicos, en este caso para crear una cita

3.2.3 Reagendar cita

Para el siguiente apartado que es reagendar una cita, se debera configurar el siguiente codigo, una vez este configurado el sistema enviara el correo electrónico con los nuevos datos de la cita.

```python
class CitaEditarView(UpdateView):
    model = Cita
    form_class = CitaEditarForm
    template_name = 'editar_cita.html'  # Reemplaza con el nombre de tu plantilla
    success_url = reverse_lazy('Agenda')

    def form_valid(self, form):
        start = form.cleaned_data['start']
        end = form.cleaned_data['end']
        dentista = form.cleaned_data['dentista']

        paciente = form.cleaned_data['paciente']
        correo = paciente.email
        title = form.cleaned_data['title']
        dentista_nombre = f"{dentista.first_name} {dentista.last_name}"

        contexto = {
            'nombre': paciente.username,
            'correo': correo,
            'start': start,
            'end': end,
            'title': title,
            'dentista': dentista_nombre,
        }

        asunto = 'Cita Reagendada'
        mensaje = f'Se ha reagendado la cita a nombre de {paciente.username}\n Para El dia: {start.date()} \n A la hora: {start.time()} \n
        con el Dentista: {dentista_nombre}'
        send_mail(asunto, mensaje, '', [correo])

        return super().form_valid(form)
```

Ilustración 9 Codigo para enviar correo electrónico de reagendar cita

3.2.4 Cancelar Cita

Para el siguiente apartado que es sobre cancelar la cita, se configura el siguiente codigo que se encargara de enviar un correo electrónico con los datos de la cita cancelada.

```python
class CitaEliminarView(DeleteView):
    model = Cita
    template_name = 'eliminar_cita.html'
    success_url = reverse_lazy('Agenda')

    def delete(self, request, *args, **kwargs):
        self.object = self.get_object()

        paciente = self.object.paciente
        correo = paciente.email

        start = self.object.start
        dentista = self.object.dentista
        dentista_nombre_completo = f"{dentista.first_name} {dentista.last_name}"

        asunto = 'Cita Cancelada'
        mensaje = f'Se ha cancelado la cita a nombre de {paciente.username} \n' \
                  f'El día: {start.date()} \n' \
                  f'A la hora: {start.time()} \n' \
                  f'Dentista: {dentista_nombre_completo}'
        send_mail(asunto, mensaje, '', [correo])

        return super().delete(request, *args, **kwargs)
```

Ilustración 10 Codigo para enviar correo electrónico de eliminar cita

Una vez realizadas todas las configuraciones para el envio de correo electrónico, se realizara la siguiente configuración de la API que es la encargada de enviar los datos ahora por medio de WhatsApp.

3.3 Configuración de Twilio

Como segunda forma de notificación, se implementará el envio de mensajes de WhatsApp, para esto se requiere una aplicación externa, en este caso twilio, se tomó como principal apoyo, ya que otorga la libertad de configurar un Bot el cual contestara los mensajes, de acuerdo con las especificaciones que se le otorguen, y otorga la libertad de personalizar los mensajes, para el cliente.

La configuración de esta API se lleva a cabo durante **10 pasos** los cuales se ejemplifican con las siguientes ilustraciones:

Paso 1: Una vez ingresado con una cuenta a twilio se creara una nueva cuenta (aplicación)

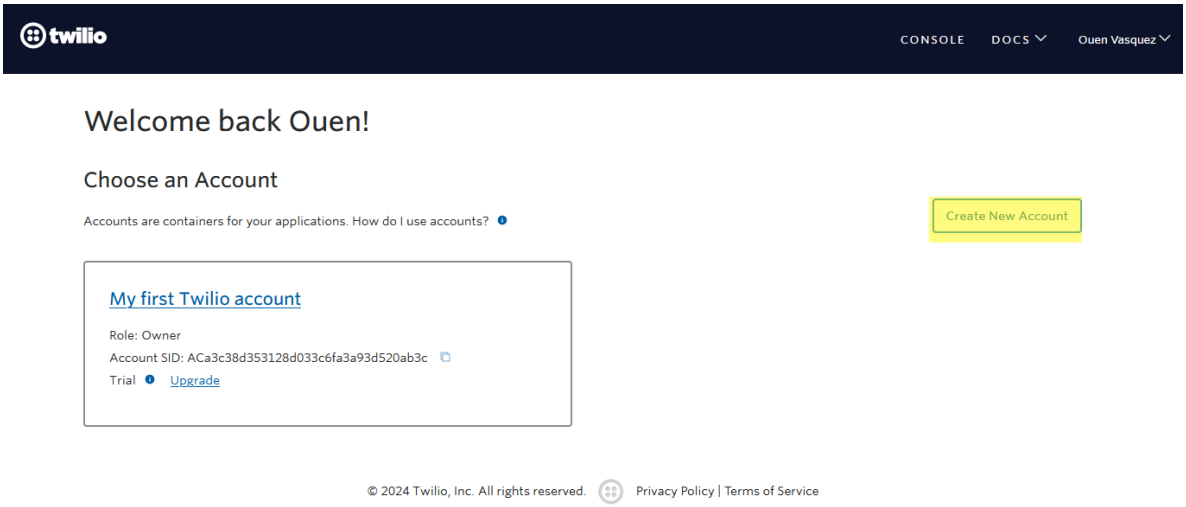


Ilustración 11 Configuración de twilio

Paso 2: Proporcionar un nombre de esta

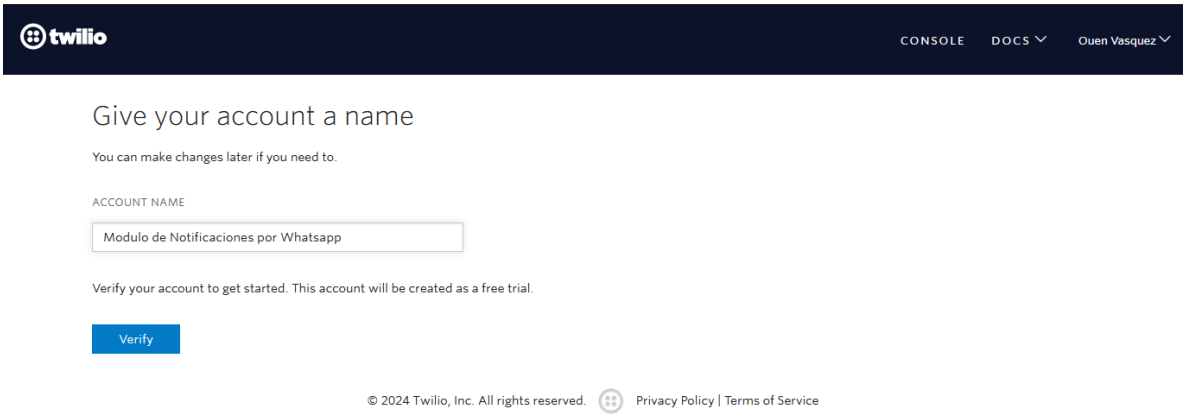


Ilustración 12 Configuración de twilio

Paso 3: Verificar tu identidad, por medio de número telefónico

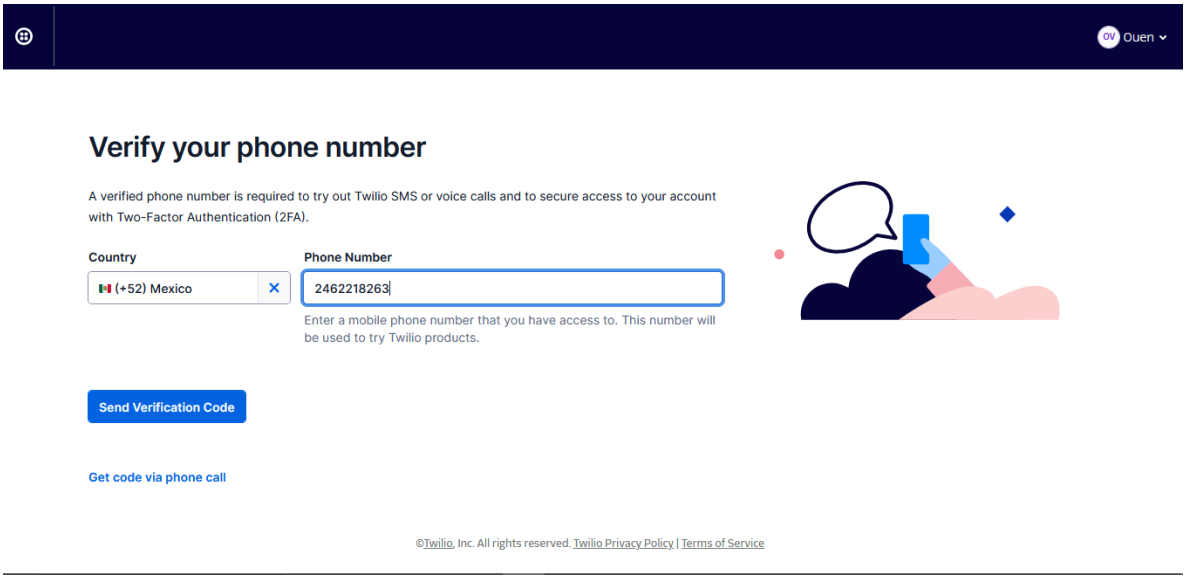


Ilustración 13 Configuración de twilio

Paso 4: Ingresar el codigo que proporciono twilio por medio de mensaje

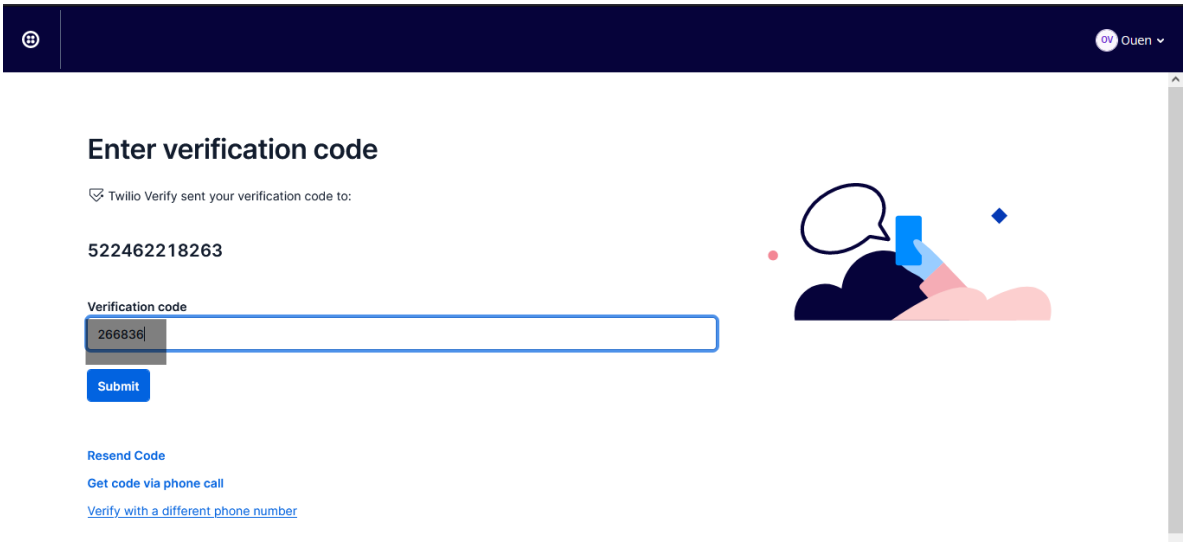


Ilustración 14 Configuración de twilio

Paso 5: Configurar la aplicación de acuerdo con las necesidades del sistema a implementar

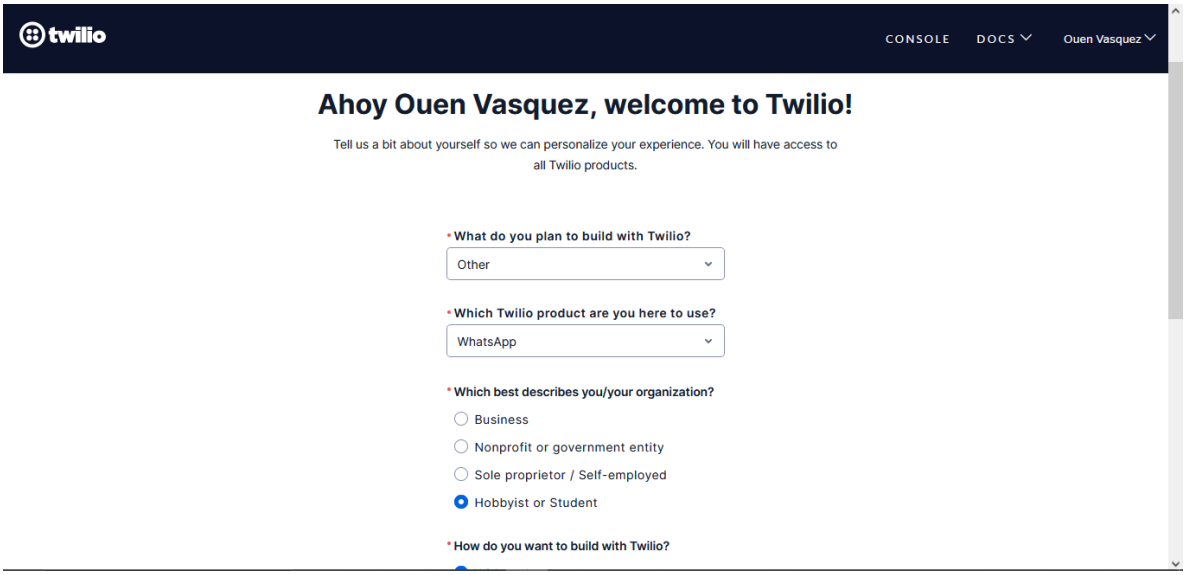


Ilustración 15 Configuración de twilio

Paso 6: Dar clic en el nombre de la aplicación para buscar llaves de acceso

Ilustración 16 Configuración de twilio

Paso 7: Copiar las llaves de acceso para configurarlas en el sistema

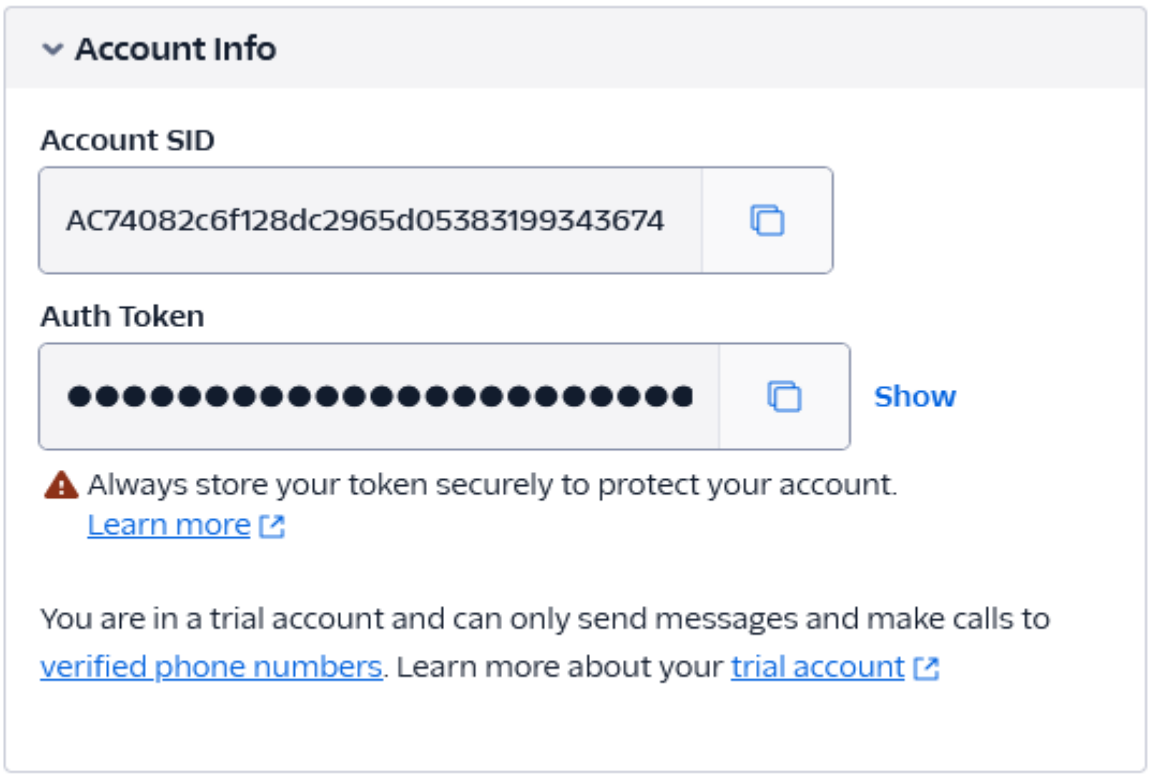


Ilustración 17 Configuración de twilio

Paso 8: Instalar el paquete twilio en el entorno virtual que se esté trabajando para poder utilizarlo en el sistema

```
(Clinica) C:\Users\ouen_\Desktop\ModuloDentalSandBox- Notificaciones>pip install twilio
Collecting twilio
  Using cached twilio-9.2.1-py2.py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: requests>=2.0.0 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from twilio) (2.25.1)
Requirement already satisfied: PyJWT<3.0.0,>=2.0.0 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from twilio) (2.8.0)
Requirement already satisfied: aiohttp>=3.8.4 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from twilio) (3.9.5)
Requirement already satisfied: aiohttp-retry>=2.8.3 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from twilio) (2.8.3)
Requirement already satisfied: aiosignal>=1.1.2 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from aiohttp>=3.8.4->twilio) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from aiohttp>=3.8.4->twilio) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from aiohttp>=3.8.4->twilio) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from aiohttp>=3.8.4->twilio) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from aiohttp>=3.8.4->twilio) (1.9.4)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from requests>=2.0.0->twilio) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from requests>=2.0.0->twilio) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from requests>=2.0.0->twilio) (1.26.
18)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ouen_\miniconda3\envs\clinica\lib\site-packages (from requests>=2.0.0->twilio) (2023.11.
17)
Using cached twilio-9.2.1-py2.py3-none-any.whl (1.8 MB)
Installing collected packages: twilio
Successfully installed twilio-9.2.1
```

Ilustración 18 Configuración de twilio

Paso 9: En el archivo setting.py crear las variables con los nombres que se muestran a continuación y pegar las llaves de acceso, para poder utilizar la API en el sistema

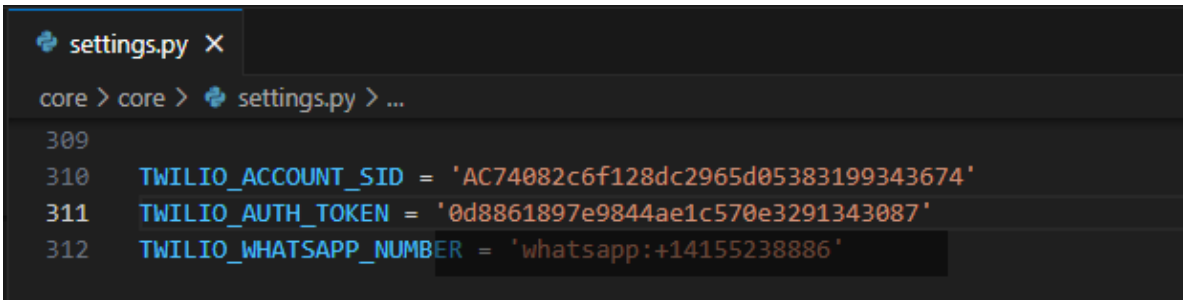


Ilustración 19 Configuración de twilio

Paso 10: Enviar el siguiente mensaje al número de twilio para conectar el WhatsApp a utilizar

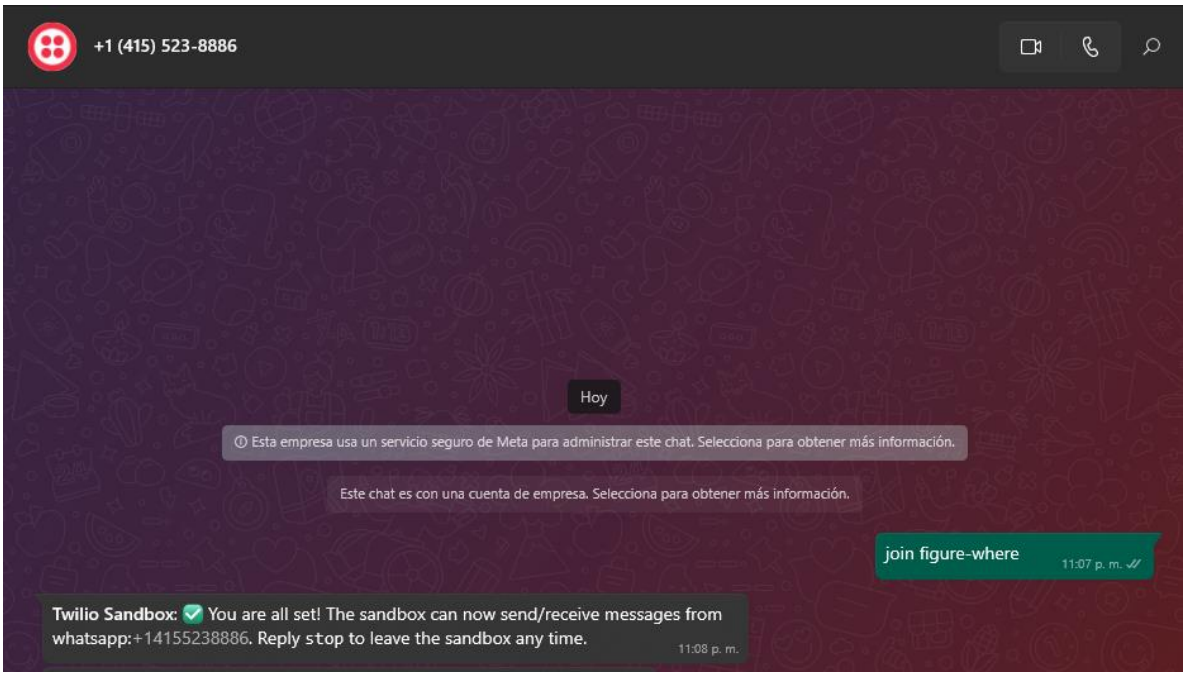


Ilustración 20 Configuración de twilio

Para que twilio pueda enviar el WhatsApp que deseamos se debe configurar el siguiente codigo, donde se especifican datos del paciente y la cita, y algunas funciones predefinidas por twilio, copiamos esta configuración a los diferentes apartados con los diferentes mensajes que se desea como se muestra en las siguientes ilustraciones.

3.3.1 Crear Cita

```
telefono = paciente.celular

client = Client(settings.TWILIO_ACCOUNT_SID, settings.TWILIO_AUTH_TOKEN)
whatsapp_message = f'Hola {paciente.username},\n' \
                   f'Se ha registrado una nueva cita con los siguientes detalles:\n' \
                   f'Fecha: {start.date()}\n' \
                   f'Hora: {start.time()}\n' \
                   f'Título: {title}\n' \
                   f'Dentista: {dentista}\n'
client.messages.create(
    body=whatsapp_message,
    from_=settings.TWILIO_WHATSAPP_NUMBER,
    to=f'whatsapp:{telefono}'
)
```

Ilustración 21 Codigo para enviar WhatsApp al registrar una cita

3.3.2 Reagendar Cita

```
telefono = paciente.celular

client = Client(settings.TWILIO_ACCOUNT_SID, settings.TWILIO_AUTH_TOKEN)
whatsapp_message = f'Hola {paciente.username},\n' \
                   f'Se ha reagendado una cita con los siguientes detalles:\n' \
                   f'Fecha: {start.date()}\n' \
                   f'Hora: {start.time()}\n' \
                   f'Título: {title}\n' \
                   f'Dentista: {dentista}\n'
client.messages.create(
    body=whatsapp_message,
    from_=settings.TWILIO_WHATSAPP_NUMBER,
    to=f'whatsapp:{telefono}'
)
```

Ilustración 22 Codigo para enviar WhatsApp al Reagendar una cita

3.3.3 Eliminar Cita

```
telefono = paciente.celular

client = Client(settings.TWILIO_ACCOUNT_SID, settings.TWILIO_AUTH_TOKEN)
whatsapp_message = f'Hola {paciente.username},\n' \
                   f'Se ha cancelado una cita con los siguientes detalles:\n' \
                   f'Fecha: {start.date()}\n' \
                   f'Hora: {start.time()}\n' \
                   f'Dentista: {dentista}\n'
client.messages.create(
    body=whatsapp_message,
    from_=settings.TWILIO_WHATSAPP_NUMBER,
    to=f'whatsapp:{telefono}'
)
```

Ilustración 23 Codigo para enviar WhatsApp al eliminar una cita

Capítulo 4 Resultados

Como resultados de la implementación del módulo tenemos una serie de ilustraciones las cuales muestran la parte grafica como lo son los formularios que son con los que interactúa el usuario, y los correos y mensajes vía WhatsApp que el usuario recibe en su teléfono celular.

4.1 Resultados Email

4.1.1 Crear Usuario

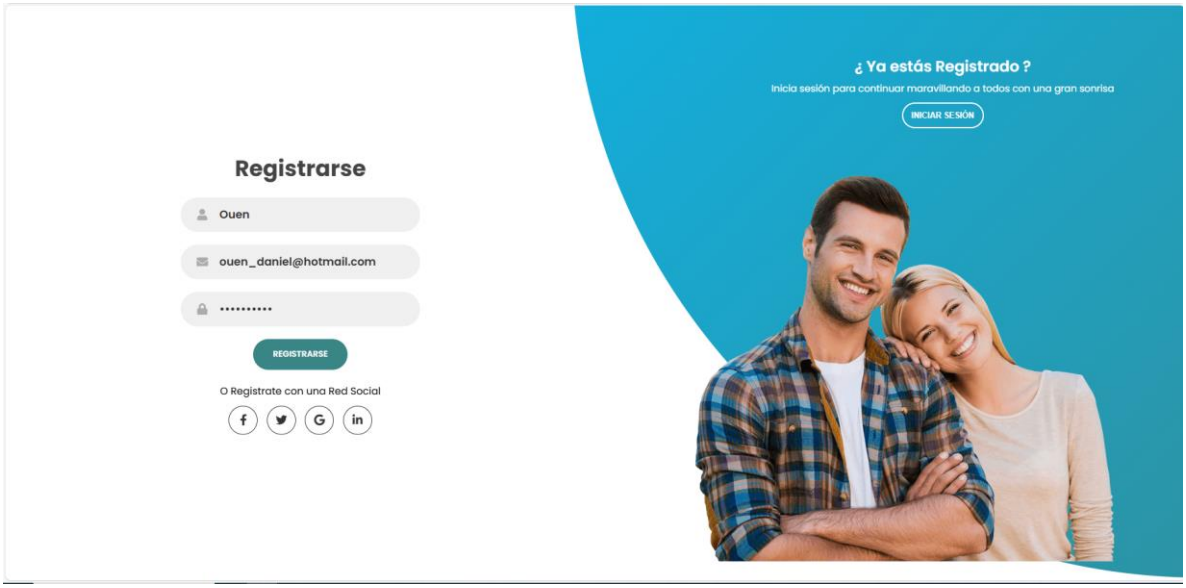


Ilustración 24 Registro de nuevo usuario desde el formulario del sistema

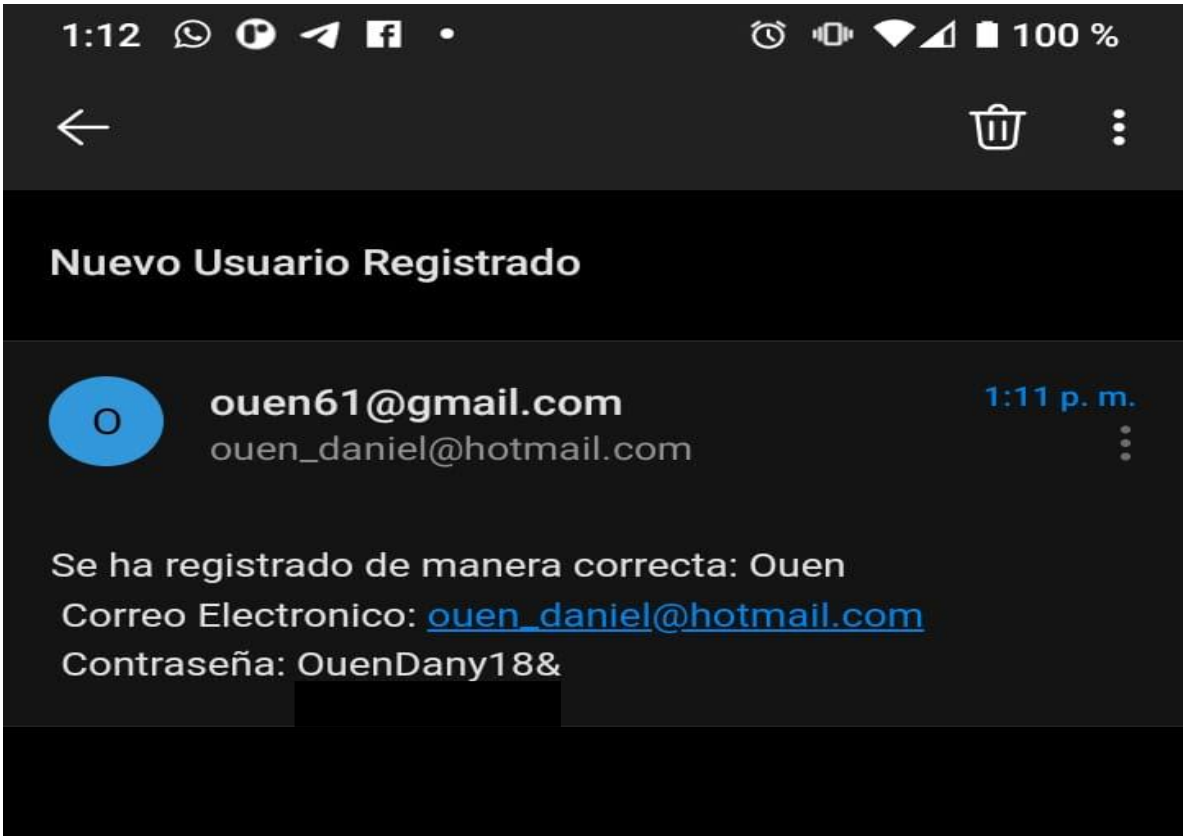


Ilustración 25 Correo enviado por el sistema con los datos que el usuario ingreso en el sistema

4.1.2 Crear Cita

Ilustración 26 Formulario para agendar cita, al dar clic el sistema envía un correo electrónico con los datos previamente configurados

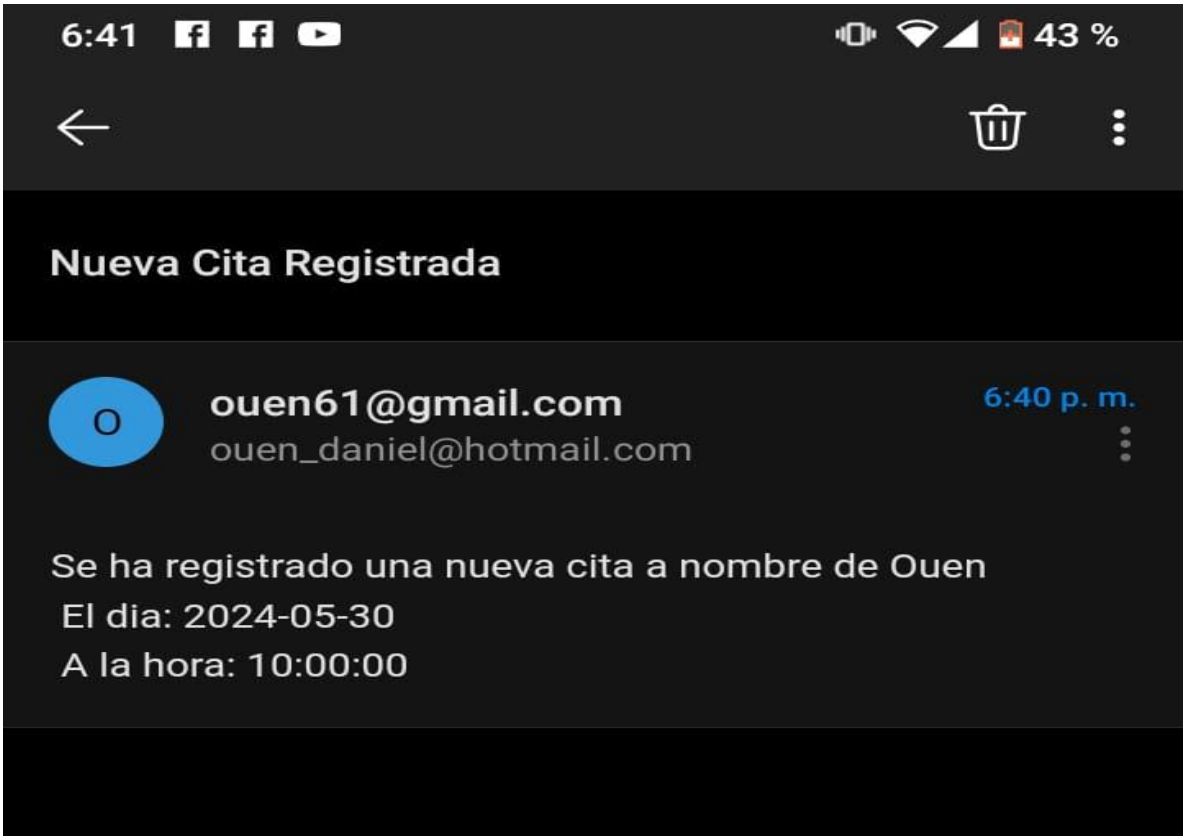


Ilustración 27 Correo recibido por el sistema, con los datos de la cita

4.1.3 Reagendar Cita

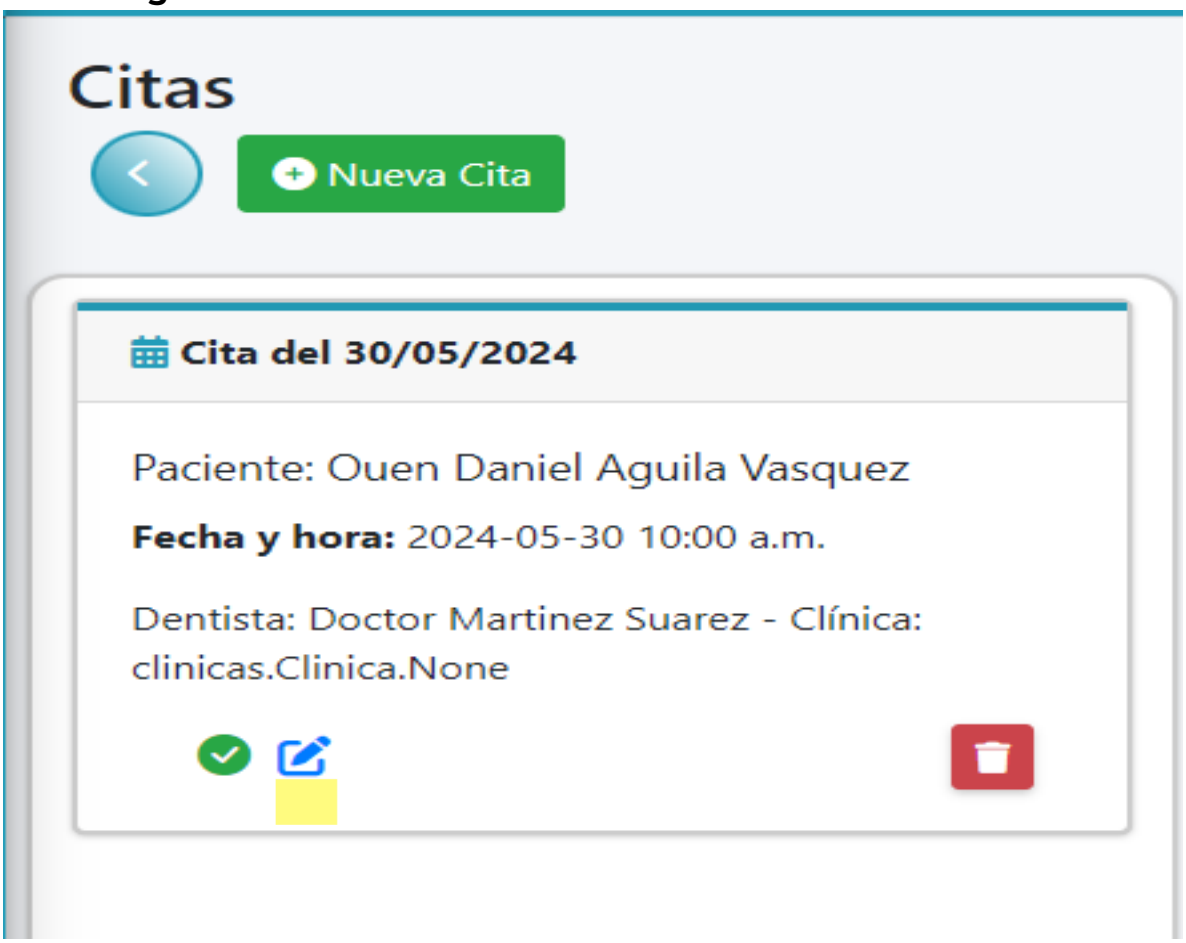


Ilustración 28 Al dar clic en el icono subrayado, se mostrará una ventana emergente, para editar algún dato de la cita

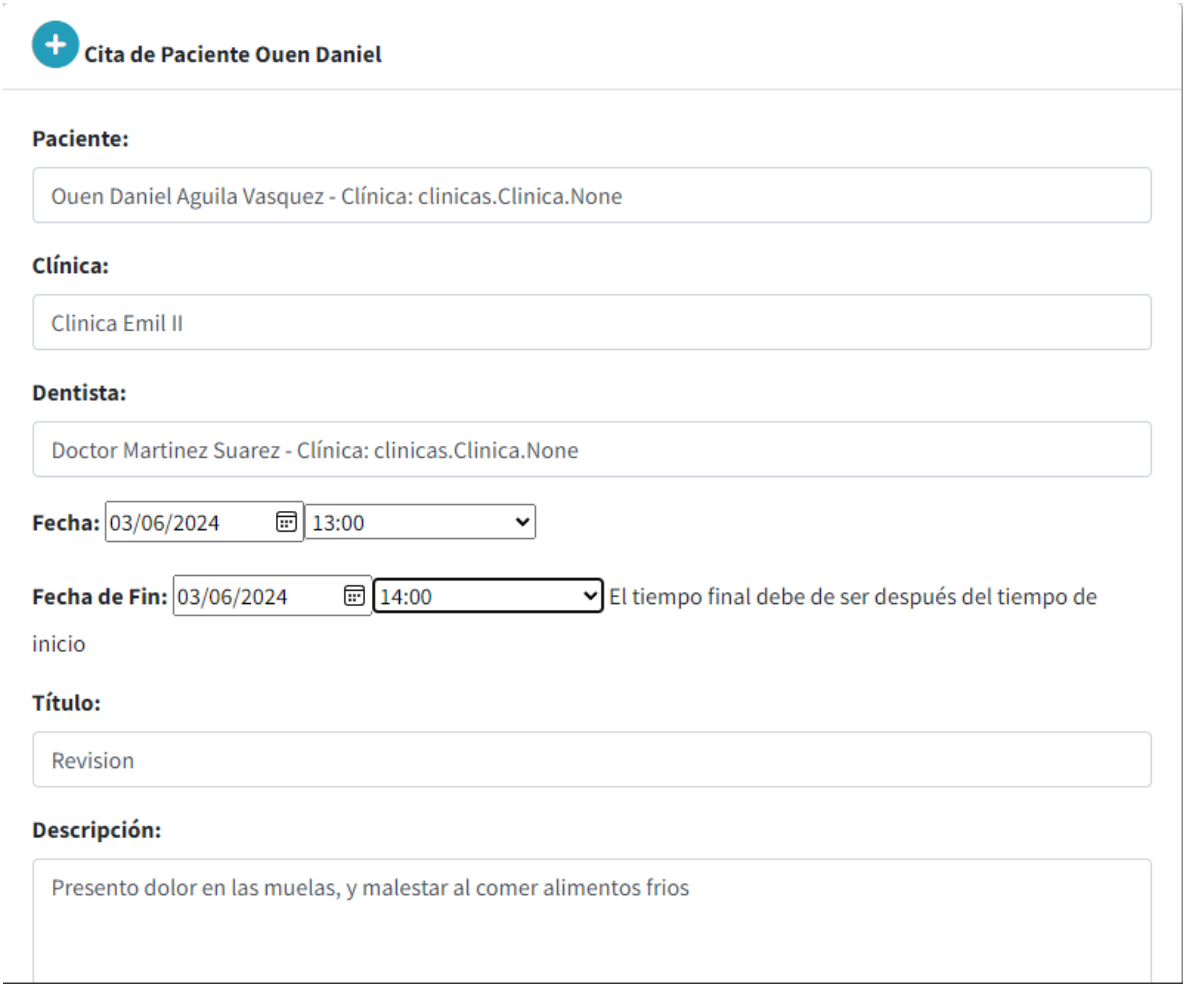


Ilustración 29 Formulario para editar los datos de la cita previamente agendada

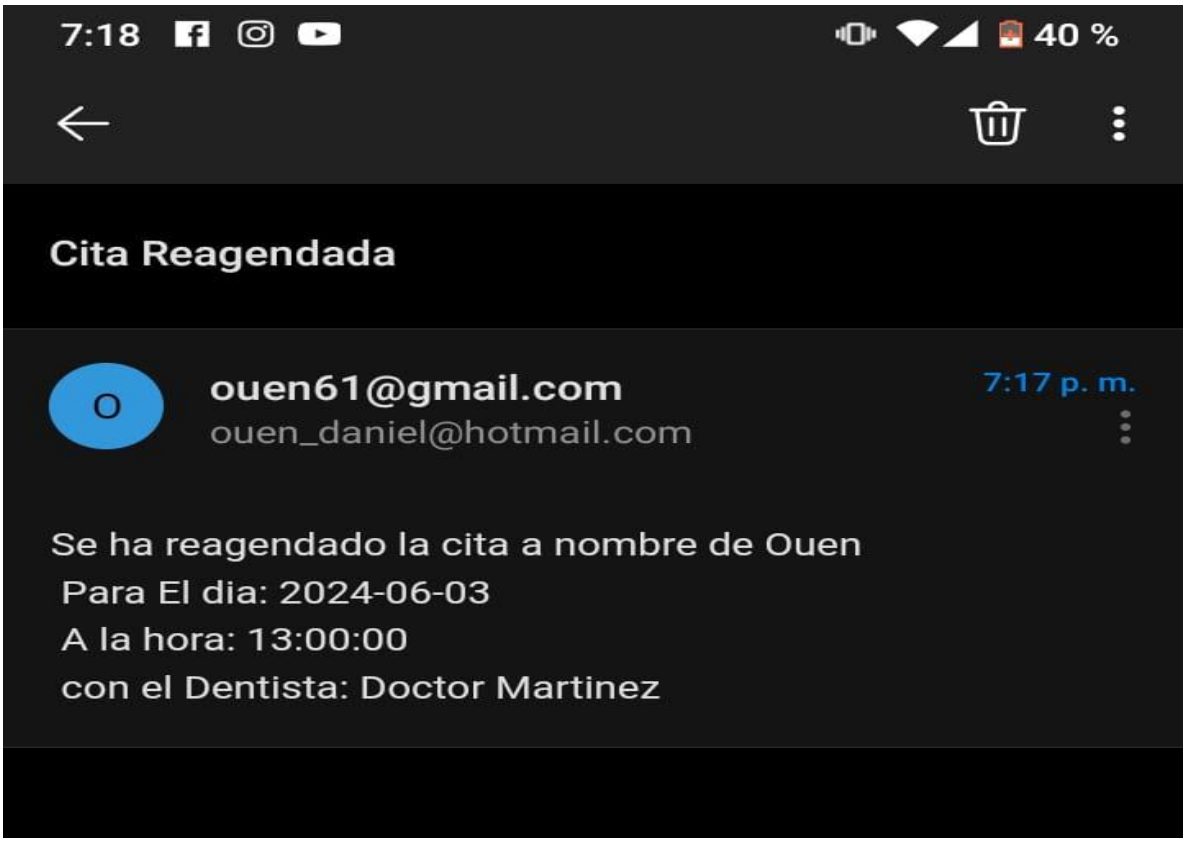


Ilustración 30 Correo electrónico de confirmación con los datos reagendados

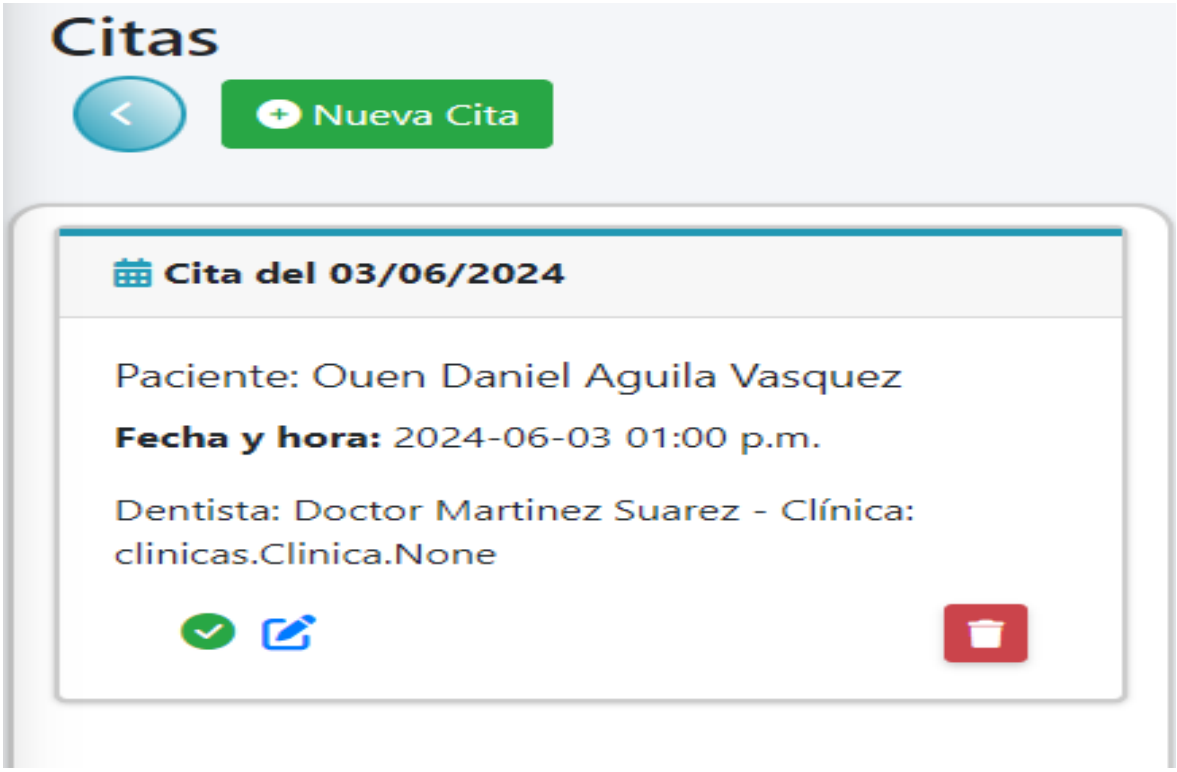


Ilustración 31 Lista de las citas próximas, ya con los datos cambiados

4.1.4 Cancelar Cita

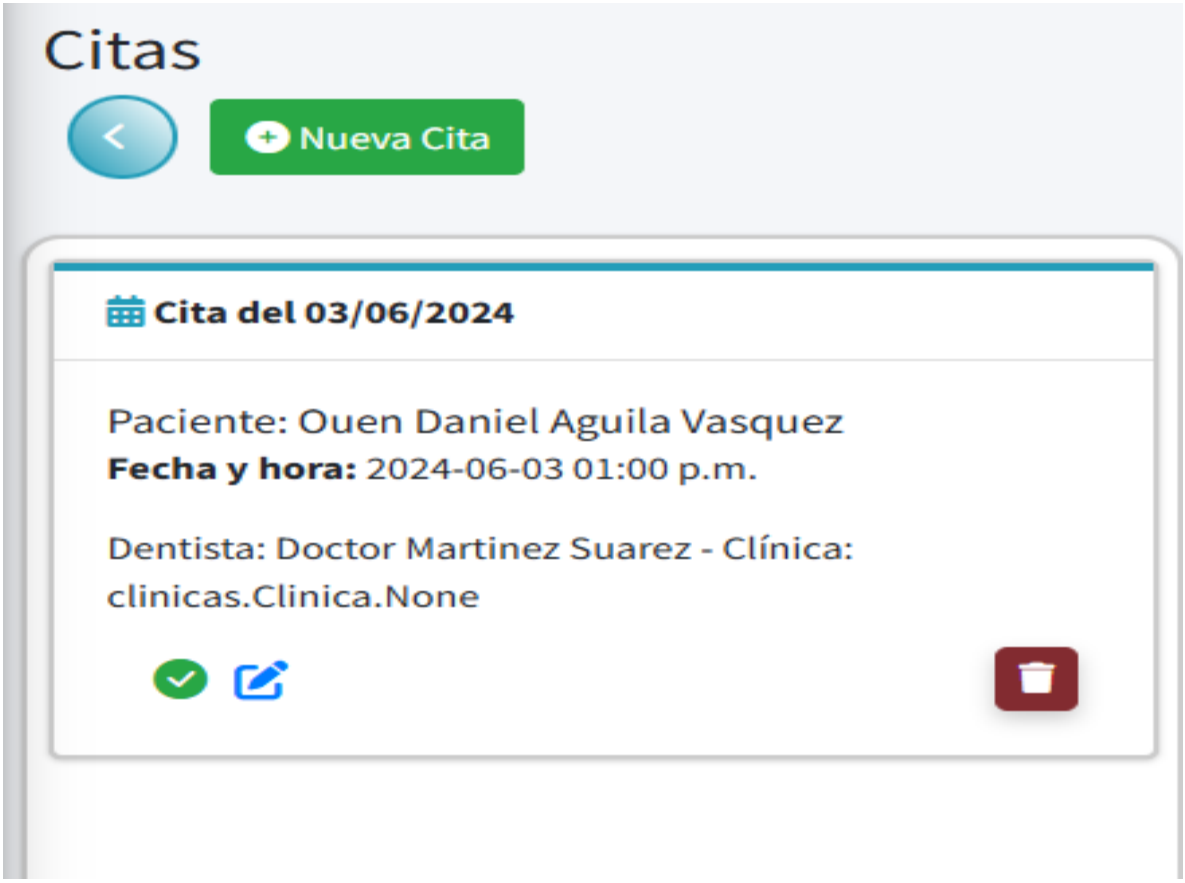


Ilustración 32 Para eliminar la cita, se dará clic al botón rojo con la papelera de reciclaje

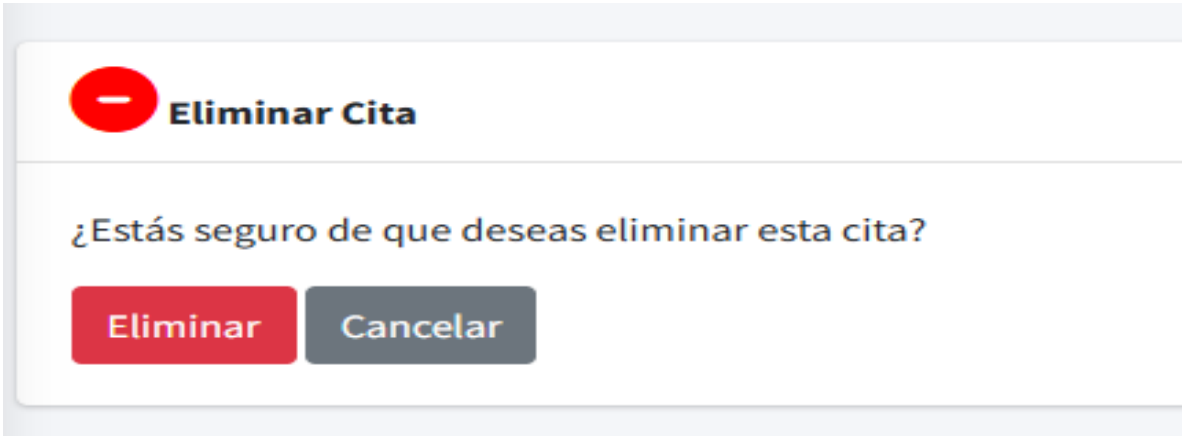


Ilustración 33 Ventana emergente, preguntando si se está seguro de eliminar la cita

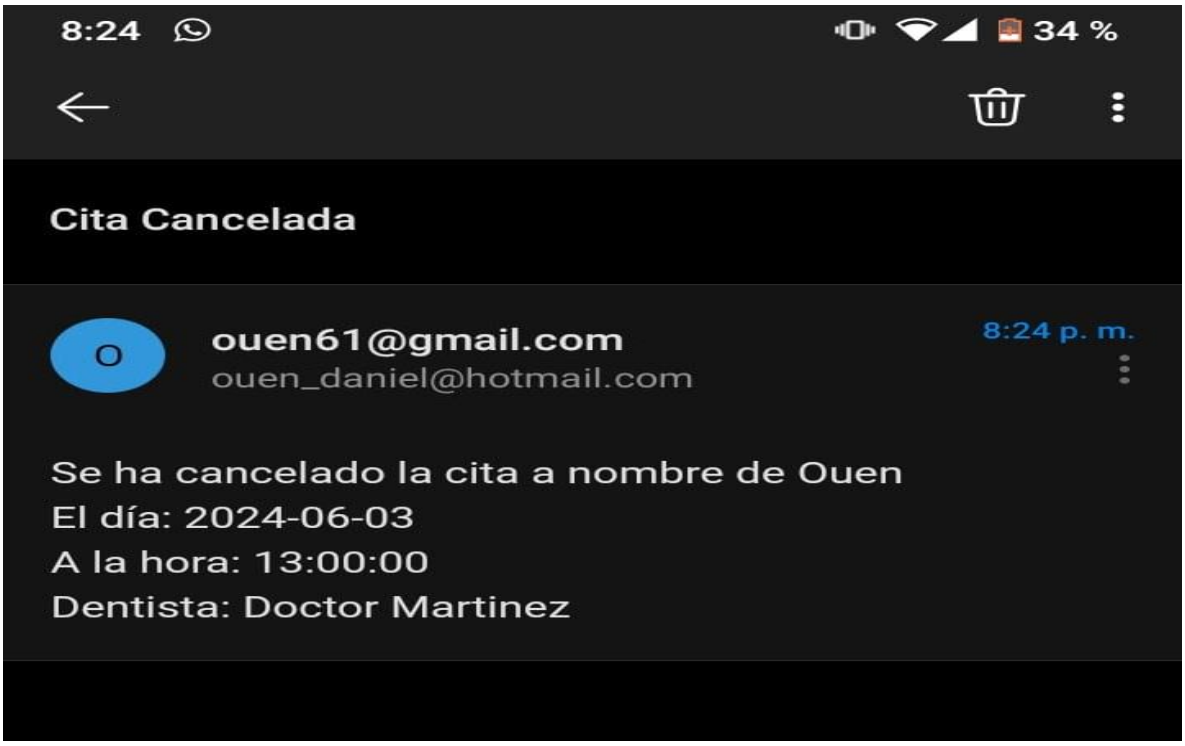


Ilustración 34 Correo de confirmación, de la eliminación de la cita

4.2 Resultados WhatsApp

4.2.1 Crear Cita

Ilustración 35 Mensaje de registro de cita

4.2.2 Reagendar Cita

Ilustración 36 Mensaje de reagendar cita

4.2.3 Cancelar Cita

Ilustración 37 Mensaje de cancelación de cita

Conclusiones

Para concluir este proyecto la implementación del módulo de notificaciones ha demostrado ser una mejora significativa en la comunicación y la eficiencia operativa del sistema. La integración de estas funcionalidades permite una notificación oportuna y eficaz a los usuarios, asegurando que reciban información relevante en tiempo real a través de los canales más utilizados.

La configuración y personalización de los servicios de notificación se realizaron de manera fluida, aprovechando las capacidades de Django 4.0 para manejar estas tareas de forma modular y escalable. Además, el uso de bibliotecas y APIs especificas per el envio de mensajes ha optimizado el proceso, garantizando una comunicación fiable y consistente.

La implementación exitosa de estas notificaciones establece una base sólida para futuras mejoras y extensiones en el proyecto, abriendo la puerta a nuevas posibilidades de interacción y automatización.

Competencias Desarrolladas

1. Administra proyecto que involucren Tecnologías de la Información y Comunicaciones para el logro de los objetivos organizacionales conforme a requerimientos establecidos.
2. Desarrolla e implementa sistemas de información para la gestión de procesos y apoyo en la toma de decisiones, utilizando metodologías basadas en estándares internacionales.
3. Diseña, desarrolla y gestiona sistemas de bases de datos para garantizar la integridad, disponibilidad y confidencialidad de la información.
4. Integra soluciones de sistemas de comunicación con diferentes tecnologías, plataformas o dispositivos.
5. Integra las diferentes arquitecturas de hardware y administra plataformas de software para incrementar la productividad en las organizaciones
6. Diseña e implementa estrategias para optimizar los procesos en la generación de negocios.
7. Utiliza tecnologías emergentes y herramientas actuales para atender necesidades acordes al entorno.
8. Posee habilidades metodológicas de investigación que fortalezcan el desarrollo cultural, científico y tecnológico en el ámbito de sistemas computacionales y disciplinas afines.
9. Diseña e implementa interfaces graficas de usuario para facilitar la interacción entre el ser humano, los equipos y sistemas electrónicos.

10. Desempeña sus actividades profesionales considerando los aspectos legales, éticos, sociales y de desarrollo sustentable.

Fuentes de información

Python. (2024). Python is powerful... and fast; plays well with others; runs everywhere; is friendly & easy to learn; is Open., https://www.python.org/about/

Django. (2024a). Django makes it easier to build better web apps more quickly and with less code, https://www.djangoproject.com/

Danielle Richardson. (2022). Git vs GitHub: What's the Difference? https://blog.hubspot.com/website/git-vs-github.

Visual Studio Code. (2021). Code editing. Redefined., *https://code.visualstudio.com/*

SQLite. (2023). About SQLite, https://www.sqlite.org/about.html

SQLiteStudio. (2024). About project, https://sqlitestudio.pl/about/

Django. (2024b). Sending Email, https://docs.djangoproject.com/en/5.0/topics/email/

Twilio. (2024). Twilio Docs, https://www.twilio.com/docs

Frizbit. (2021). Notificaciones Web Push: Guía Completa para Todo tipo de Usuarios, https://frizbit.com/es/blog/guia-notificaciones-web-push/

Pusher. (2024). Powering realtime experiences for mobile and web, https://pusher.com/

Firebase. (2023). Mensajería en la nube de Firebase, https://firebase.google.com/docs/cloud-messaging?hl=es

Vonage. (2024). Gartner Again Names Vonage a Leader in CPaaS, https://www.vonage.com/