**PROYECTO DE SERVICIOS API.**

Autores:

**JOSE LUIS GARCIA VASQUEZ**

Entregado a:

**Totto**

Departamento

Ciudad

24 de enero de 2024

**Tabla de contenido.**

[Introducción 3](#_heading=h.1pxezwc)

[Arquitectura de la API 4](#_heading=h.tqghrk3vk0xr)

[Tecnologías a utilizar 4](#_heading=h.hduymll7evyp)

[Optimización del Backend 5](#_heading=h.z5hdvgbv8nh3)

[Desarrollo del Frontend 6](#_heading=h.t7td49nq7ktd)

[Proceso de Despliegue de la Aplicación 6](#_heading=h.oge626uizco)

[Actividades Generales a Realizar 6](#_heading=h.1hs4clziirby)

[Cronograma de Actividades 8](#_heading=h.y1k0nbkx2q4y)

[Semana 1: 8](#_heading=h.coo2bsspr4uv)

[Semana 2: 8](#_heading=h.6ff5jj33t9dn)

[Semana 3: 8](#_heading=h.1hys24ehn1l2)

[Semana 4: 8](#_heading=h.7qqvajq242dc)

[Semana 5: 8](#_heading=h.m9tj82c6dib3)

[Semana 6: 9](#_heading=h.lkrikyp9ocl7)

[Semana 7: 9](#_heading=h.3jzmoaatsh6t)

[Semana 8: 9](#_heading=h.flvbmhw1lngw)

[Conclusiones 10](#_heading=h.f7sp9541ltg)

[Posibles trabajos Futuros 11](#_heading=h.t4gi3a7za7ib)

# 

# 

# 

# 

# **Introducción**

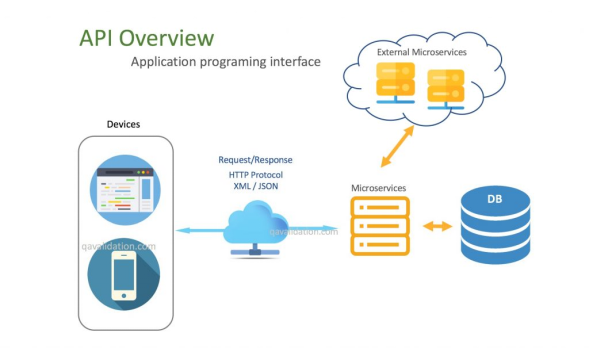
El presente informe aborda con detalle el proceso de desarrollo de una aplicación web, empleando tecnologías que incluye **Flask, React, SQLite, Oracle, Postgres.** El objetivo principal de este proyecto es la creación de un sistema integral que abarca desde la construcción de la interfaz de usuario y la lógica del servidor, la implementación de una base de datos, hasta su posterior despliegue en un entorno de producción.

A lo largo de este documento, se detallan los pasos seguidos para perfeccionar un API ya existente, diseñar y desarrollar el frontend de la aplicación, establecer una conexión eficiente entre ambas partes, llevar a cabo el proceso de despliegue en un entorno de producción para que esté disponible para los usuarios, y finalmente, se analizarán las mejoras implementadas en el sistema para optimizar su rendimiento y funcionalidad.

Este proyecto constituye un esfuerzo multidisciplinario que abarca desde la planificación inicial hasta la puesta en marcha de una aplicación web completa y funcional. A lo largo del informe, se detallarán los desafíos encontrados, las soluciones propuestas y las lecciones aprendidas en cada etapa del proceso de desarrollo, brindando una visión integral de la creación de una aplicación web de alto nivel técnico y de calidad.

# **Arquitectura de la API**

A continuación, se podrá observar la arquitectura en la cual se desarrollará la aplicación:



# **Tecnologías a utilizar**

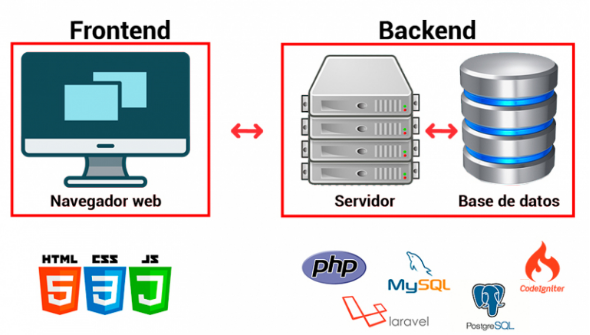
Se utilizarán las siguientes tecnologías para poder abordar el desarrollo:

* Flask: framework web Python para el desarrollo del backend.
* React: biblioteca JS para construir interfaces de usuario en el frontend.
* Bootstrap: framework CSS para diseño responsive.
* SQLite y Oracle: sistemas gestores de bases de datos.
* Gunicorn/XAMPP: servidores para despliegue en producción.

# **Optimización del Backend**

Se partió de un API simple en Flask y se realizarán las siguientes mejoras:

* Refactorización del código para mayor legibilidad.
* Manejo de datos de prueba con diferentes endpoints.
* Base de datos SQLite, Oracle, para almacenar usuarios.
* Modelos SQLAlchemy para mapear la base de datos.
* Endpoints de registro e inicio de sesión.
* Autenticación con Flask-Login.
* Documentación de API con Swagger.

****

# **Desarrollo del Frontend**

El frontend se desarrollará en React o Bootstrap con las siguientes páginas:

* **Página de Inicio**: página principal.
* **Página de Registro**: formulario para crear cuentas.
* **Login de Usuarios**: formulario para iniciar sesión.
* **Tabla**: muestra los datos consumidos desde la API.

Se utilizarán peticiones **Fetch/AJAX** para consumir los endpoints del Backend; los datos se renderizan en componentes de React.

# **Proceso de Despliegue de la Aplicación**

* El Backend se desplegó con Gunicorn.
* El Frontend se sirve desde el mismo servidor Flask.
* Se monitorizó el rendimiento y estabilidad en producción.

# **Actividades Generales a Realizar**

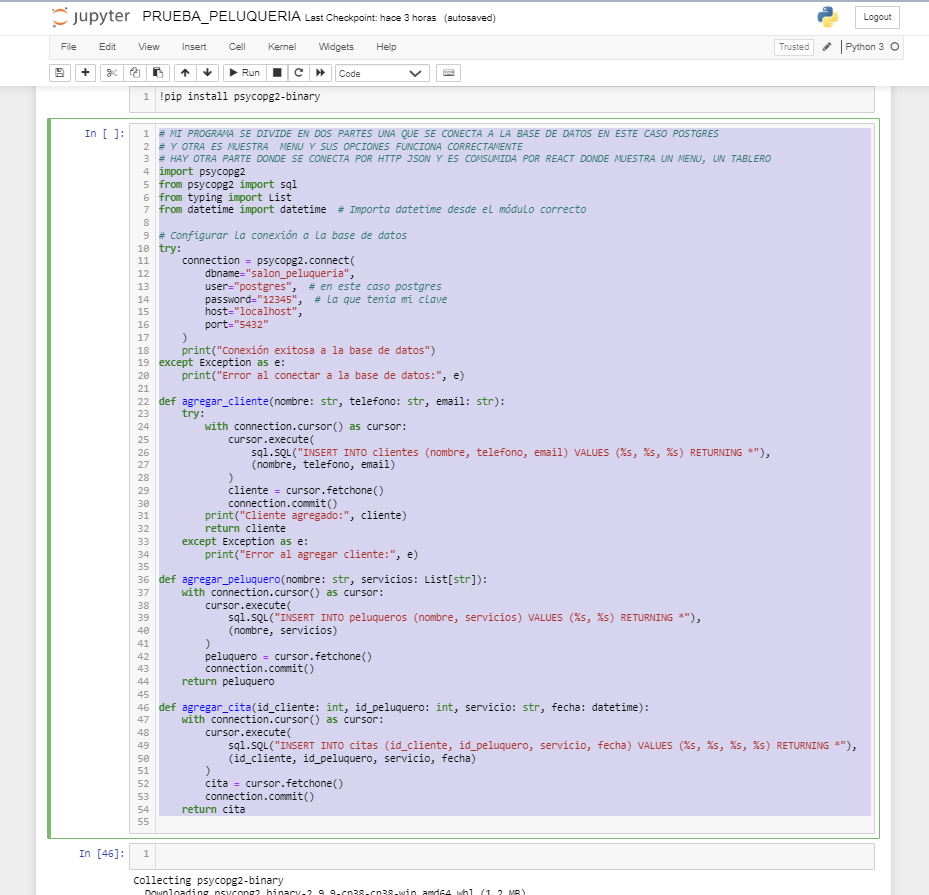
1. Mejorar código de API, ya que este se encuentra plano.
2. Probar datos los datos de la API.
3. Crear todas las vistas necesarias.
4. Validar datos.
5. Crear una base de datos SQLite, Oracle, para almacenar usuarios y contraseñas.
6. Definir modelos con Flask-SQLAlchemy.
7. Agregar endpoint de registro e inicio de sesión.
8. Probar Flask-Login para manejar sesiones. - api gmail - usuario y contraseña definido.
9. Proteger rutas con @login\_required.
10. Probar usuarios autenticados garantizando que pueden acceder a la aplicación.
11. Desarrollar la estructura Frontend con React para las Páginas: Inicio, Registro, Inicio de sesión y la Tabla de datos.
12. Utilizar un framework como Flask, Bootstrap, react para tener una plantilla básica responsive y componentes UI.
13. Crear vistas con Jinja2 para el login, lista de datos, detalle, etc.
14. Agregar control de rutas y autenticación de usuario.
15. Para el Frontend, consumir la API desde Javascript fetch()/AJAX y desplegar los datos en el formato HTML.
16. Consumir API de datos desde React para mostrar la tabla.
17. Probar axios para requests.
18. Desplegar Backend y Frontend.
19. Probar Frontend desde la misma app Flask.
20. Configurar Gunicorn para servir la app Flask en producción.
21. Probar y monitorear el desempeño.
22. Ajustar configuraciones si es necesario.
23. Agregar documentación de la API con Swagger.
24. Mejoras adicionales: paginación, búsquedas, más seguridad, CI/CD.
25. Usa DataTables.js para la representación de tablas con paginación, búsqueda, etc.
26. Agregar gráficos usando Chart.js para visualizaciones.
27. Implementar un servicio CRUD para gestionar los datos mediante la API.
28. Subir el proyecto a un servidor como Heroku o PythonAnywhere para que esté disponible públicamente.
29. Optimizar el Frontend con buen diseño UI/UX. Templates como Bootstrap, react.
30. Agregar documentación con Swagger UI para la API.
31. Utiliza un sistema de control de versiones como Git y repositorios como GitHub.

# **Código api Python**

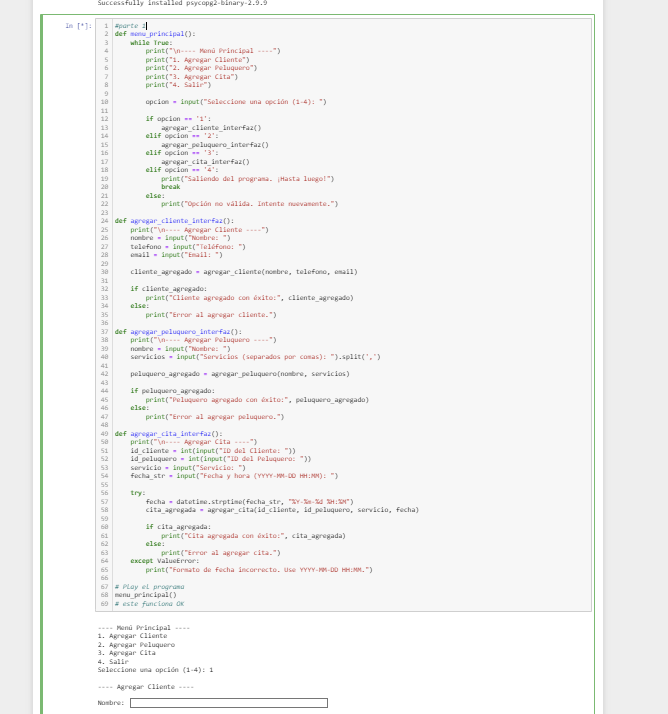
En Jupyter, se introduce el código y se compila en dos secciones distintas. La primera sección se encarga de establecer la conexión con la API, mientras que la segunda presenta un menú interactivo que permite a los usuarios interactuar según sus necesidades.

El código correspondiente es el siguiente:

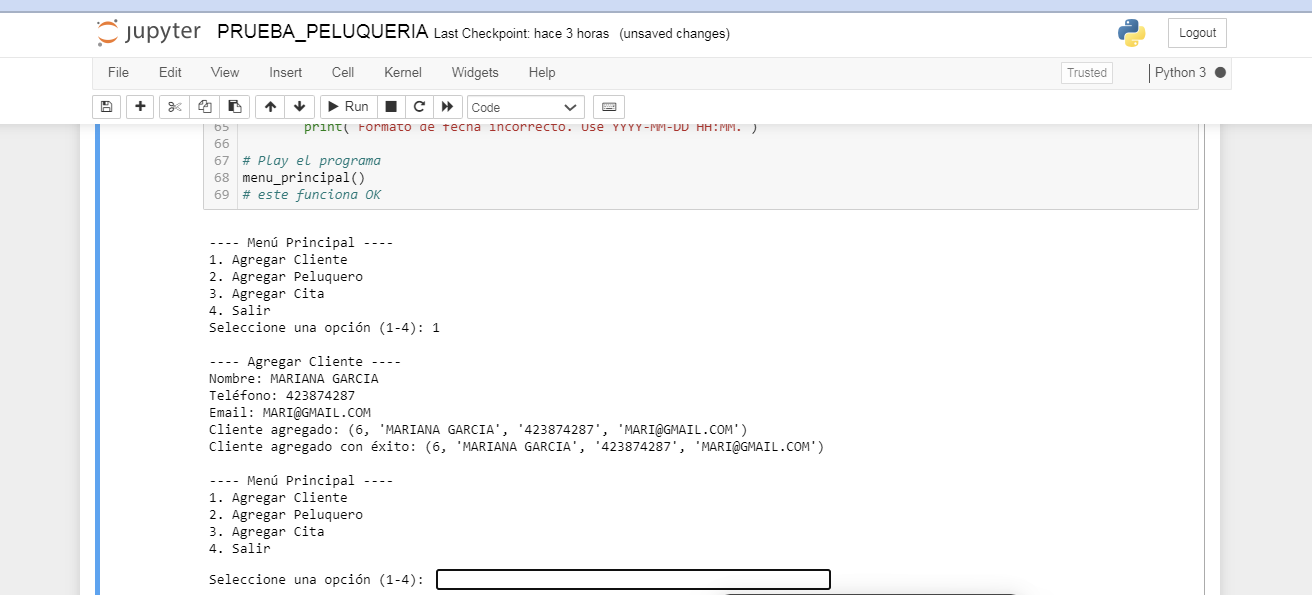
Parte 1.



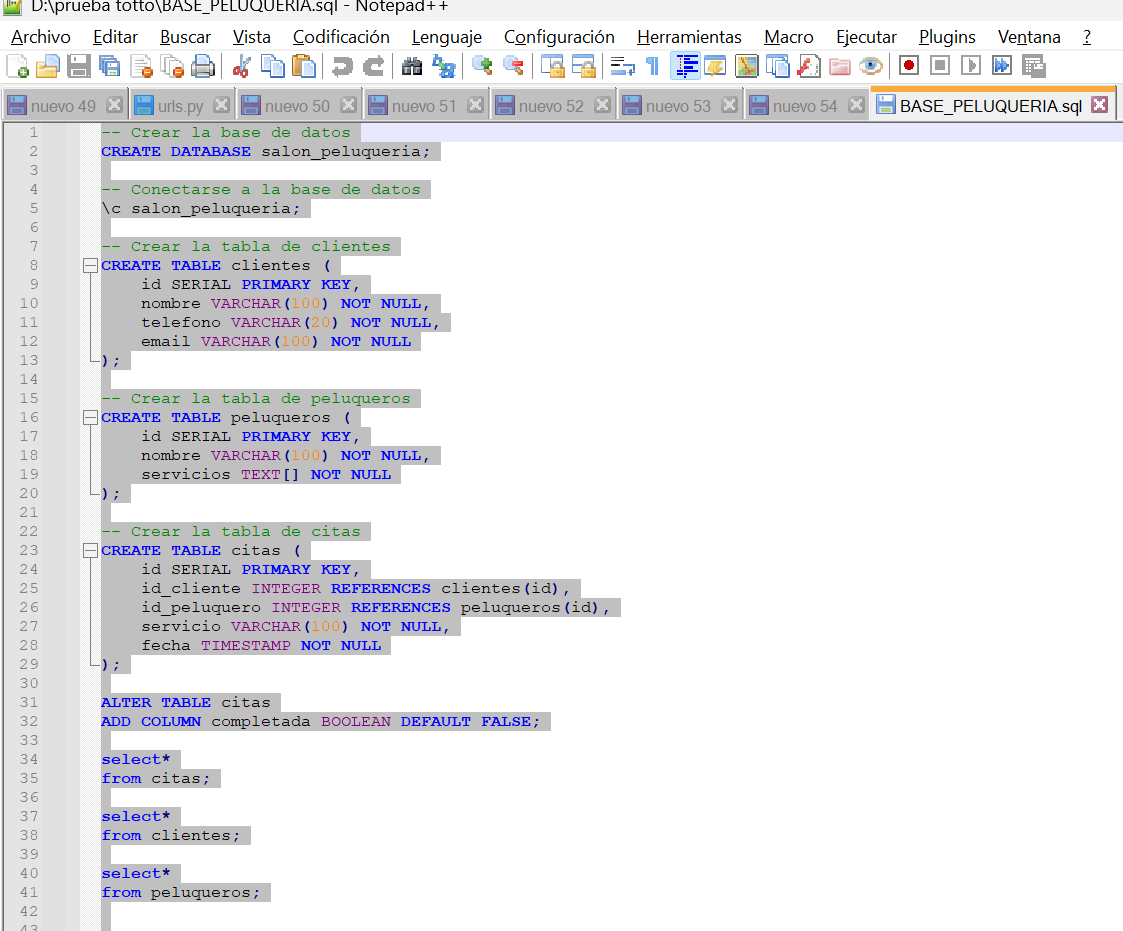
Parte 2.



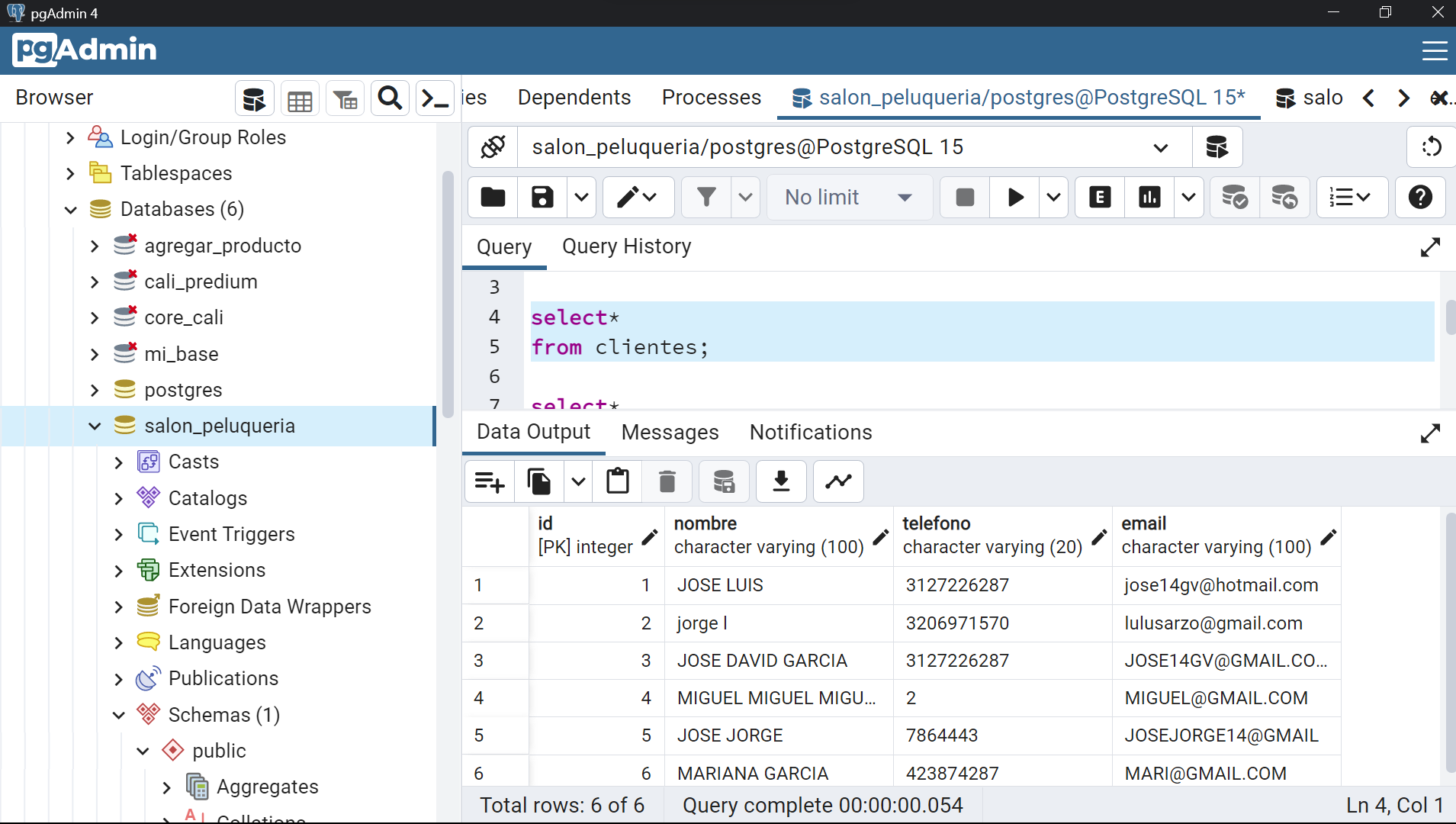
Para acceder al menú, se procede a compilar la segunda parte del código. Una vez en el menú, se selecciona la opción deseada, siguiendo así con la ejecución del programa de acuerdo con la elección realizada.



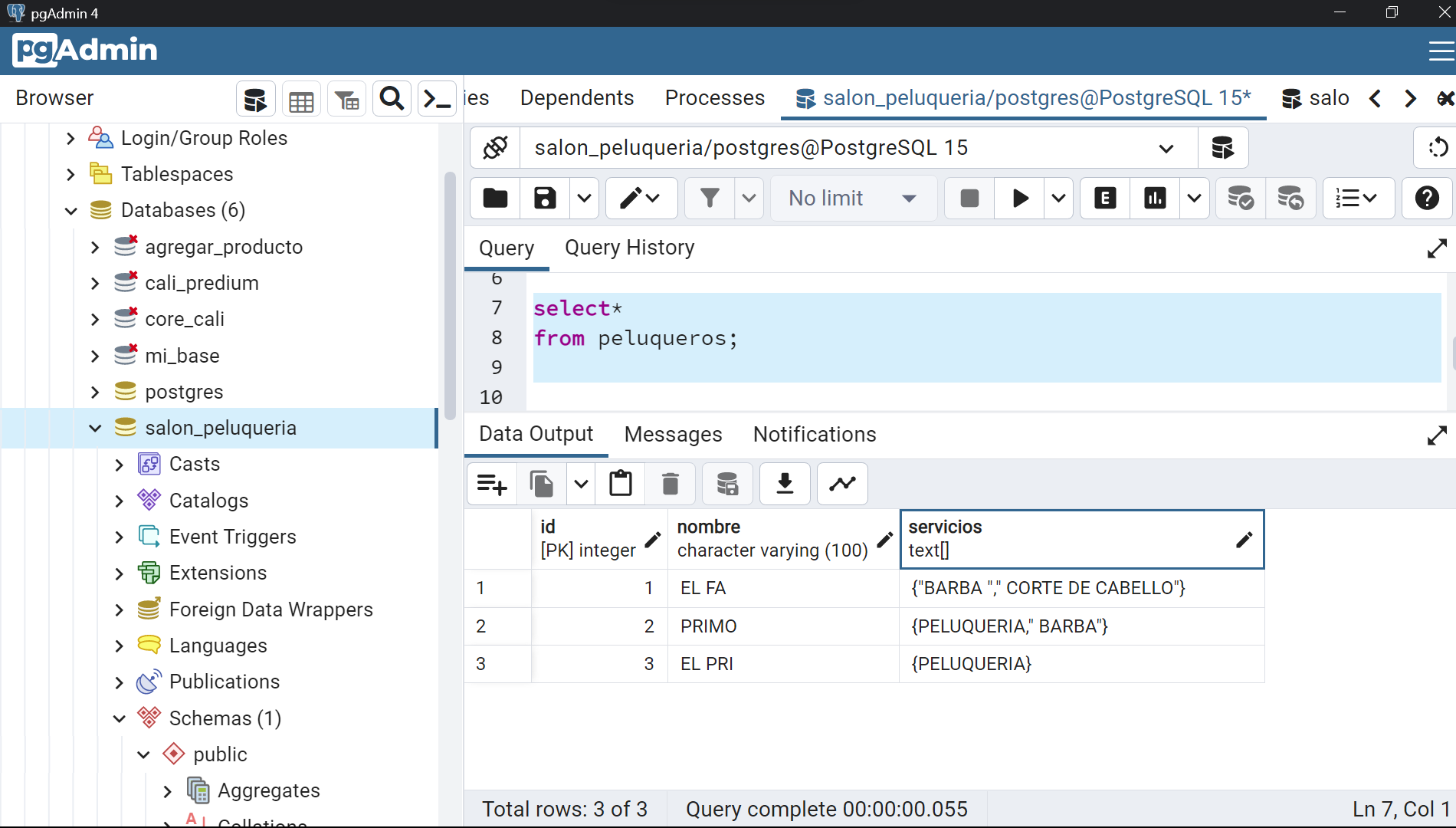
# **Base de datos**



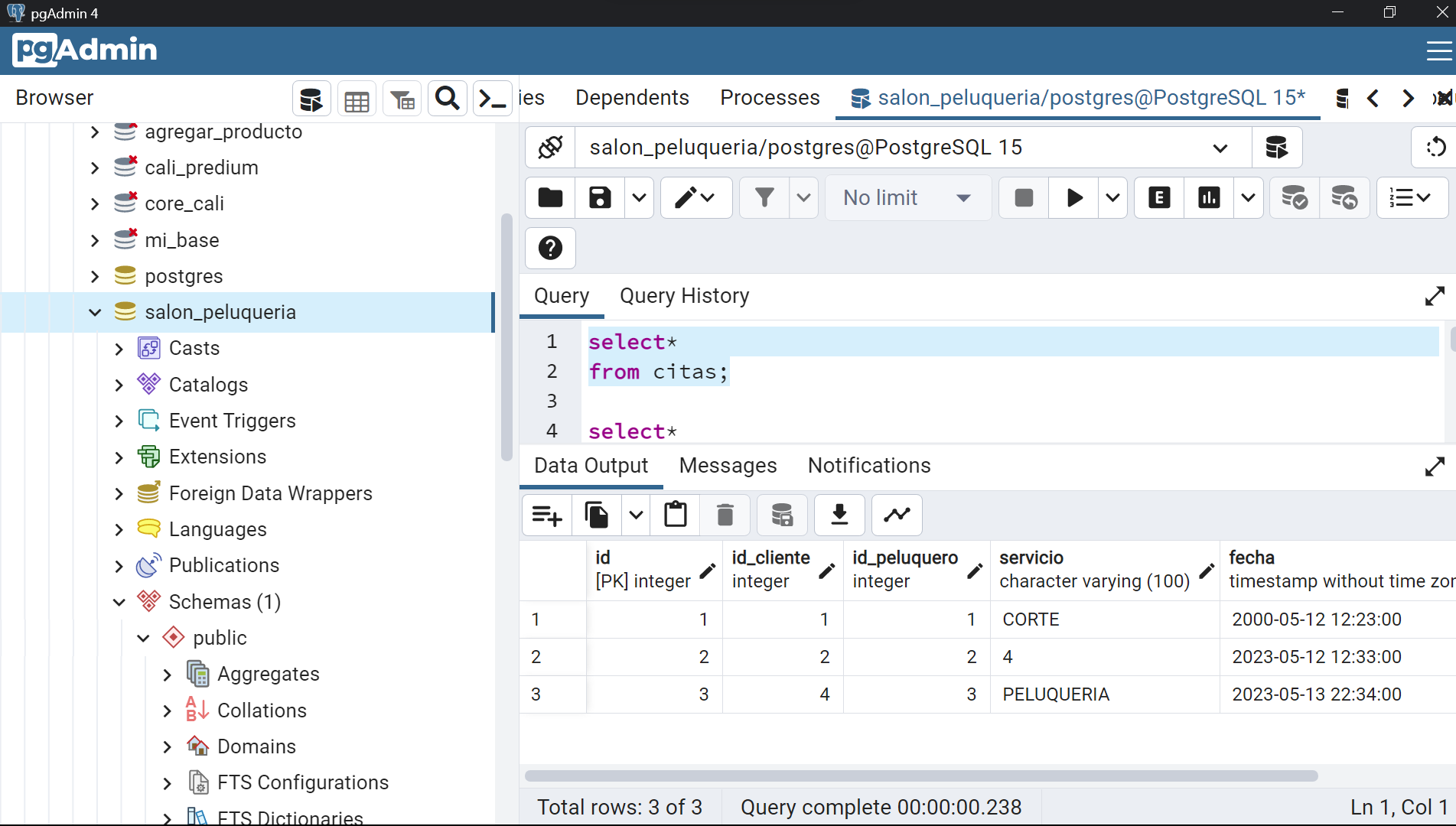
Clientes



Peluqueros



Citas



# **Cronograma de Actividades**

# **Semana 1:**

* Mejorar código API (2 días).
* Probar datos API (1 día).
* Configurar la base de datos de los usuarios (2 días).

# **Semana 2:**

* Definir modelos de datos (1 día).
* Crear endpoints de autenticación (2 días).
* Probar autenticación con Flask-Login (2 días).

# **Semana 3:**

* Crear estructura básica de frontend en React (2 días).
* Crear páginas de Inicio, Login, Registro (3 días).

# **Semana 4:**

* Integrar frontend y backend (2 días).
* Consumir API desde React (2 días).
* Pintar tabla de datos desde API (2 días).

# **Semana 5:**

* Desplegar backend y frontend (2 días).
* Probar despliegue (1 día).
* Configurar monitorización (2 días).

# **Semana 6:**

* Documentar API con Swagger (2 días).
* Mejoras de rendimiento y estabilidad (3 días).

# **Semana 7:**

* Agregar paginación y búsquedas (3 días).
* Más seguridad en APIs y contraseñas (2 días).

# **Semana 8:**

* Mejorar UI/UX en frontend (3 días).
* Gráficos y reportes (2 días).
* Configurar CI/CD (2 días).

**Nota**: La última semana se trabajará domingo, no se tiene en cuenta festivos y días de cuenta de cobro.

# **Conclusiones**

Al cumplirse los objetivos planteados desarrollando un sistema completo con tecnologías web modernas. Se adquirió experiencia en construir backends, frontends, conectarlos e implementar mejoras.

# **Posibles trabajos Futuros**

* Gráficos implementados en React que permitan visualizar los datos de forma clara e interactiva, esto mejora la experiencia de usuario y facilita la toma de decisiones al obtener insights del negocio.
* Como trabajo futuro, se podrían agregar más tipos de gráficos como heatmap, scatterplot.
* También se podría permitir exportar los gráficos a formatos como CSV, PNG, PDF y datos.

# 

# 