

Base de Datos y Programación Web – Descripción del Proyecto

Escuela de Ingeniería Civil Informática

Universidad de Valparaíso

Alejandro Dinamarca - Lucas Rojas

PLANTEAMIENTO Y DESARROLLO DE HISTORIAS DE USUARIOS

1 CONTEXTO GENERAL

ConsultorioDigital es la plataforma que facilitará el acceso a los centros asistenciales primarios - tales como consultorios o Centros de Salud Familiar (CESFAM) - de salud en Chile, a partir de medios digitales.

Normalmente, el primer punto de contacto de los pacientes con el sistema de salud es a partir de los consultorios, CESFAM, postas rurales, etc. No obstante, el acceso a ellos es siempre en formato presencial, dado su característica de atender, con mayor frecuencia, urgencias, accidentes, entre otros.

No obstante, en la pandemia del COVID-19, se dejó en evidencia la falta de cobertura y acceso a estos centros, como, por ejemplo, en la implementación de vacunatorios móviles para acercar la vacunación a las personas, o inclusive, en los PCR Móvil para tener mayor muestra de los infectados en pandemia. Esto, dado que los centros de salud eran vistos como factor de riesgo para contagiarse con el virus.

Una publicación oficial del Colegio Médico de Chile declara que, a partir de la primera ola de la pandemia, la población estaba "...temerosa de asistir a los centros asistenciales (...)", y además que existía la necesidad de "...considerar procesos asistenciales que permitan mantener una menor presencia física en los centros" [1].

Es aquí donde ConsultorioDigital permite aliviar las afluencias a los servicios de salud, a partir de la gestión de citas, solicitando hora directamente en el sitio web donde el sistema internamente coordina los horarios con los consultorios comunales.

2 TECNOLOGÍAS

- **Django:** Un framework web de alto nivel en Python que será el núcleo del backend, gestionando la lógica, la autenticación, y la interacción con la base de datos.
- **Python:** El lenguaje de programación principal del proyecto, utilizado para desarrollar el backend en Django.
- **JavaScript:** Utilizado en el frontend para mejorar la interactividad y la experiencia del usuario en la plataforma.
- **SQLite3:** Una base de datos ligera que se utilizará para almacenar la información relacionada con usuarios, centros asistenciales, citas y demás datos críticos del sistema.

3 COMPONENTES WEB

3.1 REQUISITOS FUNCIONALES Y NO FUNCIONALES

Con el objetivo de definir y detallar lo que el sistema debe hacer a partir de las necesidades previamente indicadas, los siguientes requisitos funcionales y no funcionales son descritos

Los requisitos funcionales son servicios que el sistema debe ser capaz de proveer al usuario, entre los cuales encontramos:

1. Reserva, Cancelación y Reprogramación de Citas y Turnos:

- El sistema debe permitir a los usuarios reservar citas médicas a través de la plataforma.
- El sistema debe permitir a los usuarios cancelar citas previamente reservadas.
- El sistema debe permitir a los usuarios reprogramar citas, cambiando la fecha y hora según disponibilidad.

2. Búsqueda de Centros Asistenciales:

- El sistema debe permitir a los usuarios buscar centros asistenciales según su ubicación actual o una dirección específica.
- El sistema debe permitir a los usuarios filtrar los centros asistenciales por el tipo de servicio requerido (e.g., especialidades médicas).
- El sistema debe mostrar los centros asistenciales con el menor tiempo de espera disponible.
- El sistema debe permitir a los usuarios buscar y seleccionar centros asistenciales utilizando criterios relevantes, como la distancia, calificación de otros usuarios, y horarios de atención.

3. Reportes, Estadísticas y Gestión de Usuarios para Administradores:

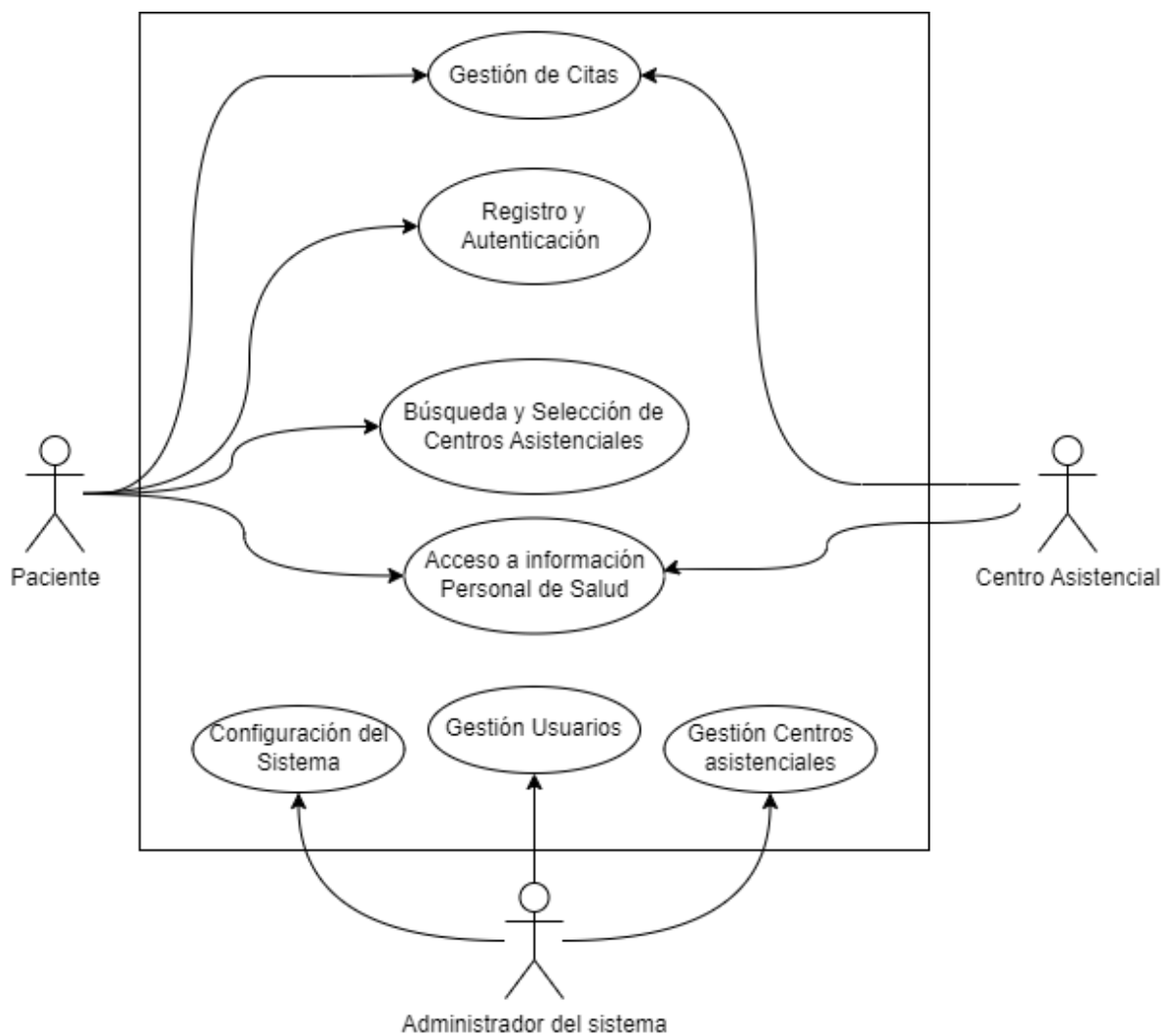
- El sistema debe proporcionar a los administradores reportes y estadísticas sobre la utilización de servicios, número de citas reservadas, canceladas y reprogramadas.
- El sistema debe permitir a los administradores gestionar la información de los usuarios, incluyendo la creación, modificación y eliminación de cuentas.
- El sistema debe permitir a los administradores gestionar y actualizar la información de los centros asistenciales, incluyendo especialidades y horarios de atención.

Los requisitos no funcionales son aquellos que no se encuentran relacionados directamente a los servicios ofrecidos, pero si pueden relacionarse con propiedades emergentes del sistema:

- Proteger la información de los usuarios
- Se debe tener una disponibilidad 24/7 con un tiempo entre fallas mínimo.
- Capacidad de adaptabilidad y escalabilidad, manteniendo lo antes mencionado.
- El sistema debe mantener su rendimiento óptimo y cumplir con los requisitos de seguridad y disponibilidad durante su escalabilidad.

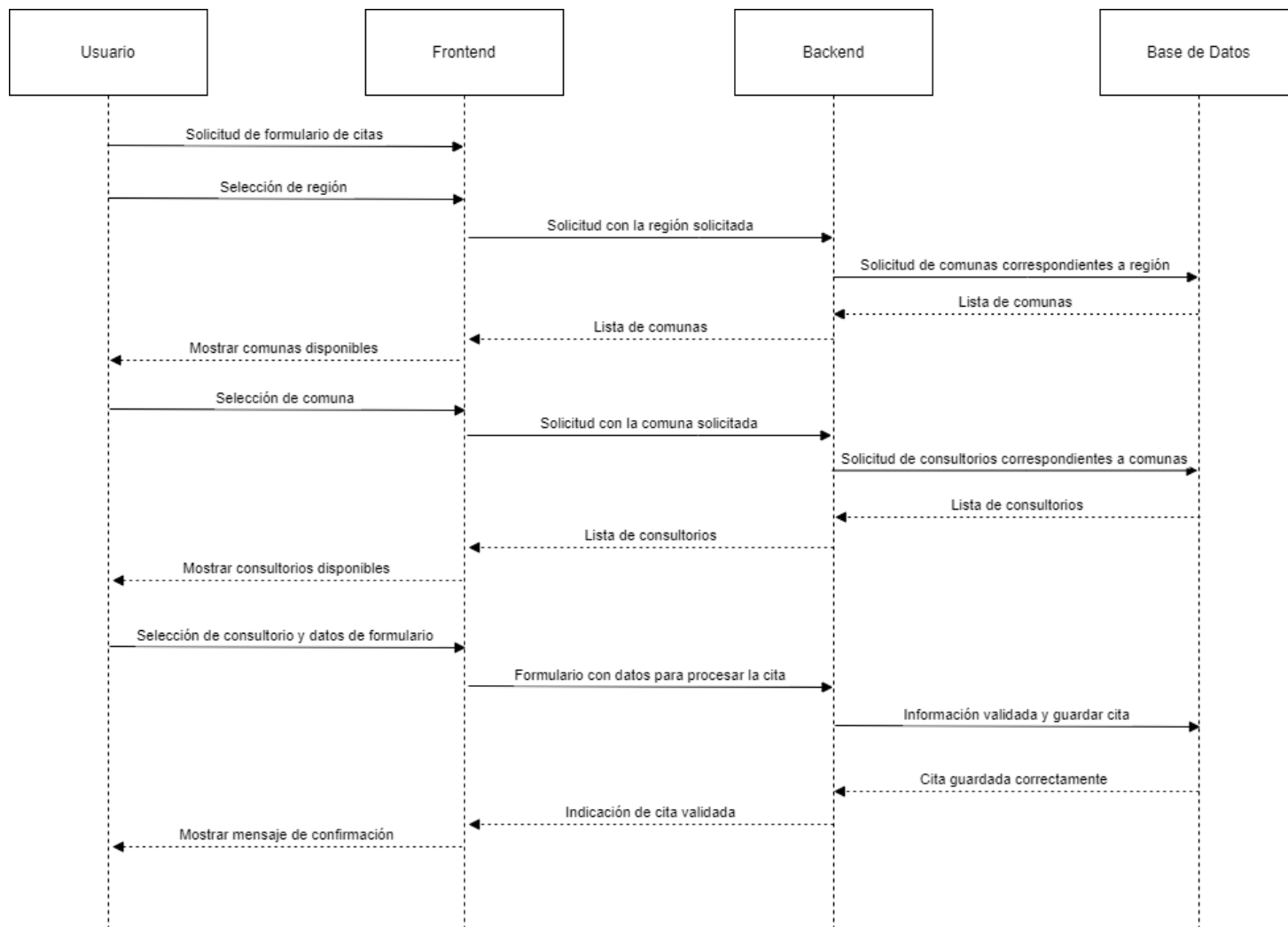
Los componentes web se encuentran en el repositorio GitHub solicitado:
https://github.com/adinamarca/salud_publica_digital/

3.2 DIAGRAMA DE CASO DE USO

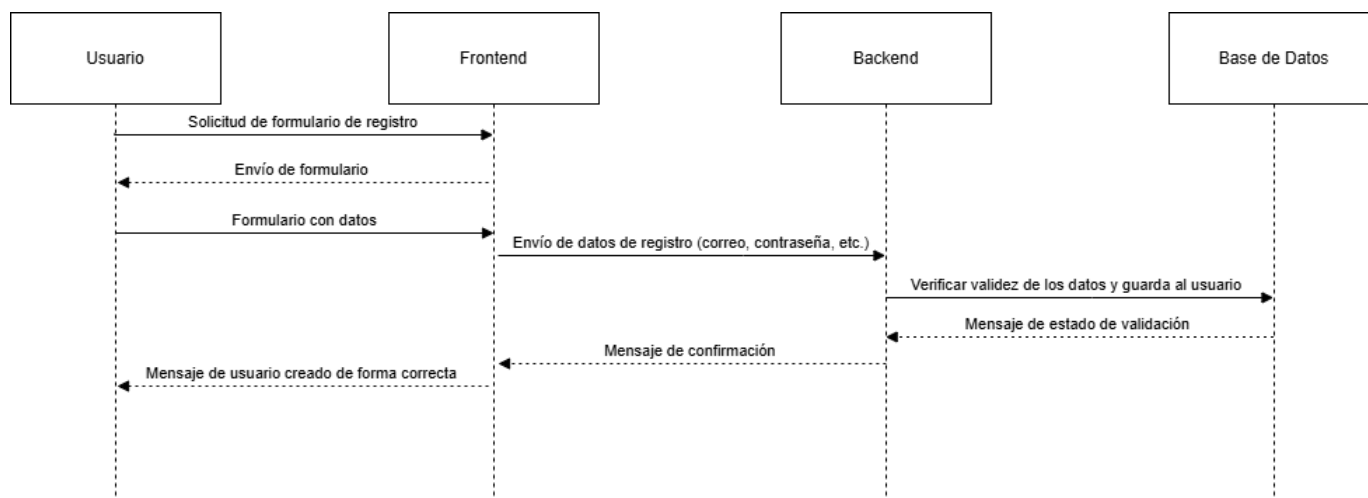


3.3 DIAGRAMAS DE SECUENCIA

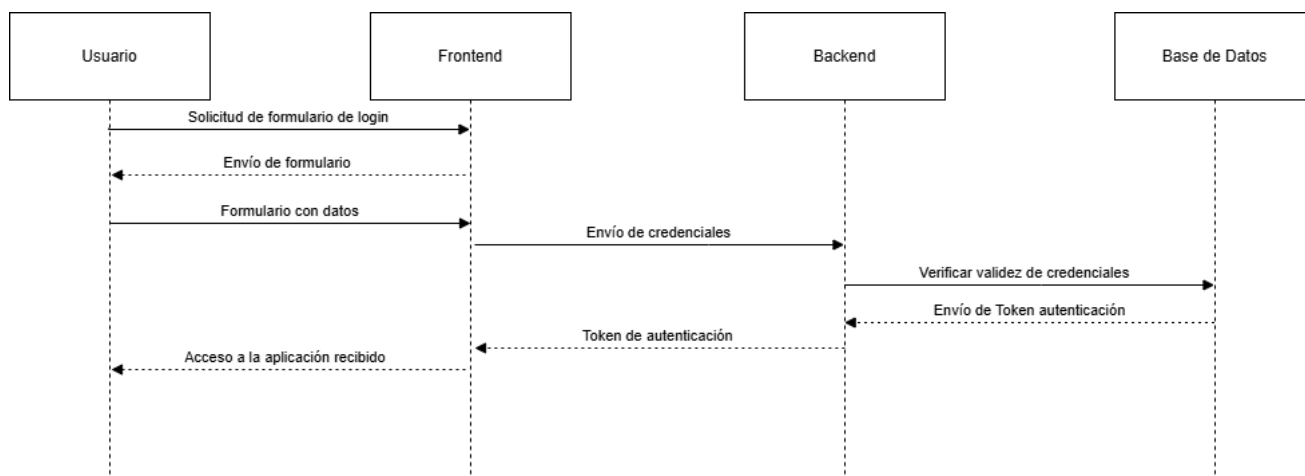
-Agendar una cita:



-Registro de pacientes:

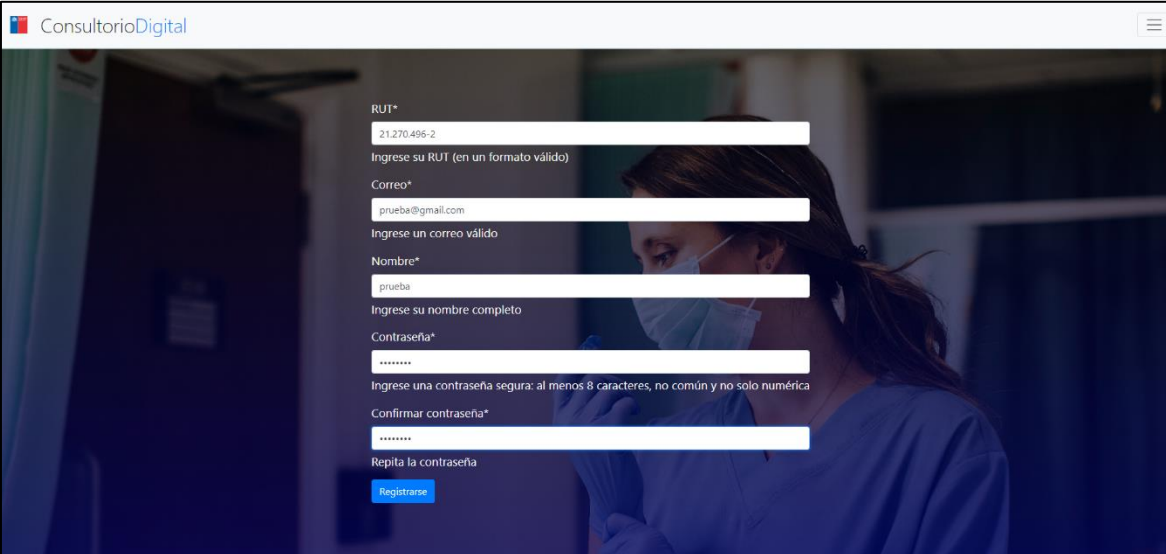


-Inicio de sesión:



3.4 PROTOTIPOS DE INTERFAZ GRÁFICA

Prototipo de interfaz gráfica que permite ilustrar las funcionalidades a las que el usuario podrá acceder.



ConsultorioDigital

RUT*

21.270.496-2

Ingrese su RUT (en un formato válido)

Correo*

prueba@gmail.com

Ingrese un correo válido

Nombre*

prueba

Ingrese su nombre completo

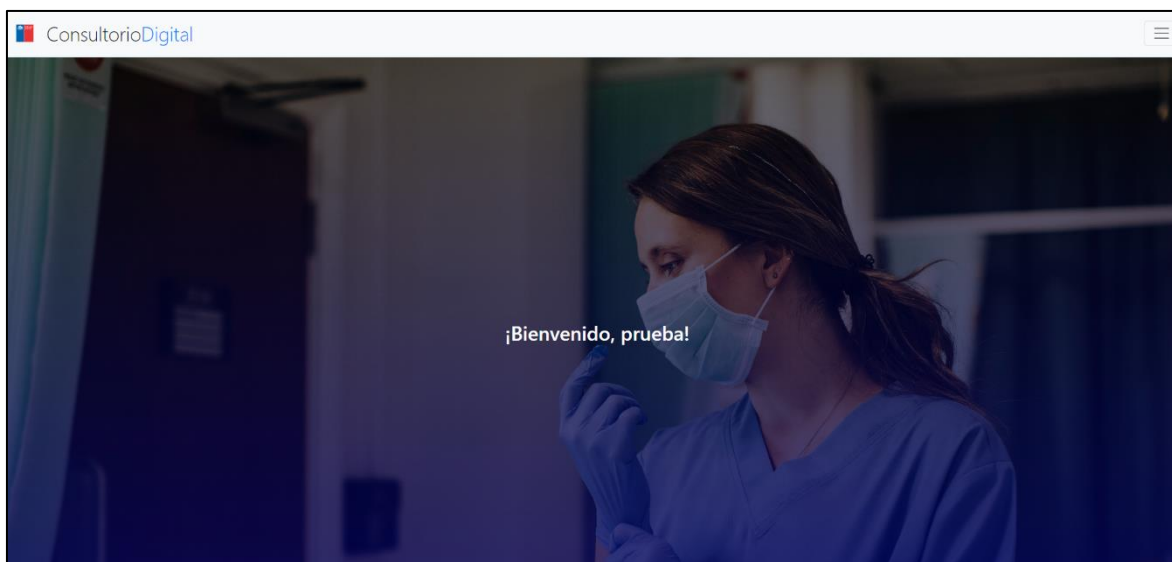
Contraseña*


Ingrese una contraseña segura: al menos 8 caracteres, no común y no solo numérica


Confirmar contraseña*

Repita la contraseña

Registrarse



 ConsultorioDigital



Agenda tu cita

Región:

Seleccione una región

Comuna:

Seleccione una comuna

Consultorio:

Seleccione un consultorio


Motivo de Consulta:


Ingrese el motivo de la consulta

Fecha de Consulta:

mm/dd/yyyy

Enviar

 ConsultorioDigital




Agenda tu cita

Región:

Seleccione una región

- Región De Tarapacá
- Región De Antofagasta
- Región De Atacama
- Región De Coquimbo
- Región De Valparaíso
- Región Del Libertador Gral. B. O'Higgins
- Región Del Maule
- Región Del Biobío
- Región De La Araucanía
- Región De Los Lagos
- Región De Aysén del General Carlos Ibañez del Campo
- Región De Magallanes y de la Antártica Chilena**
- Región Metropolitana de Santiago
- Región De Los Ríos
- Región De Los Ríos
- Región De Arica Parinacota
- Región De Ñuble


ConsultorioDigital

Página principal


Mis horas

Reservar hora
Cancelar hora
Cerrar sesión

Bienvenido, Alejandro Dinamarca

Mis horas agendadas

Nombre del Consultorio	Hora Agendada	Razón de Agendar	Estado
Centro de Salud Familiar	2024-07-15 09:00	Consulta Dental	Confirmado
Centro de Salud Familiar	2024-07-16 11:00	Control Médico	Confirmado
Centro de Salud Familiar	2024-07-17 14:00	Consulta General	Confirmado
Centro de Salud Familiar	2024-07-18 10:00	Consulta General	En espera de profesional
Centro de Salud Familiar	2024-07-19 08:30	Consulta General	En espera de profesional


ConsultorioDigital

Cancela tu cita

Tu cita solo puede ser cancelada con 24 horas de anticipación. Si no puedes asistir, por favor cancela tu cita para que otro paciente pueda ocupar tu hora.

Código de confirmación:

Motivo por el que cancela:

[Enviar](#)

Django administration

WELCOME, ALEJANDRO DINAMARCA

Home > Authentication and Authorization > Users

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

Select user to change

[Search](#)

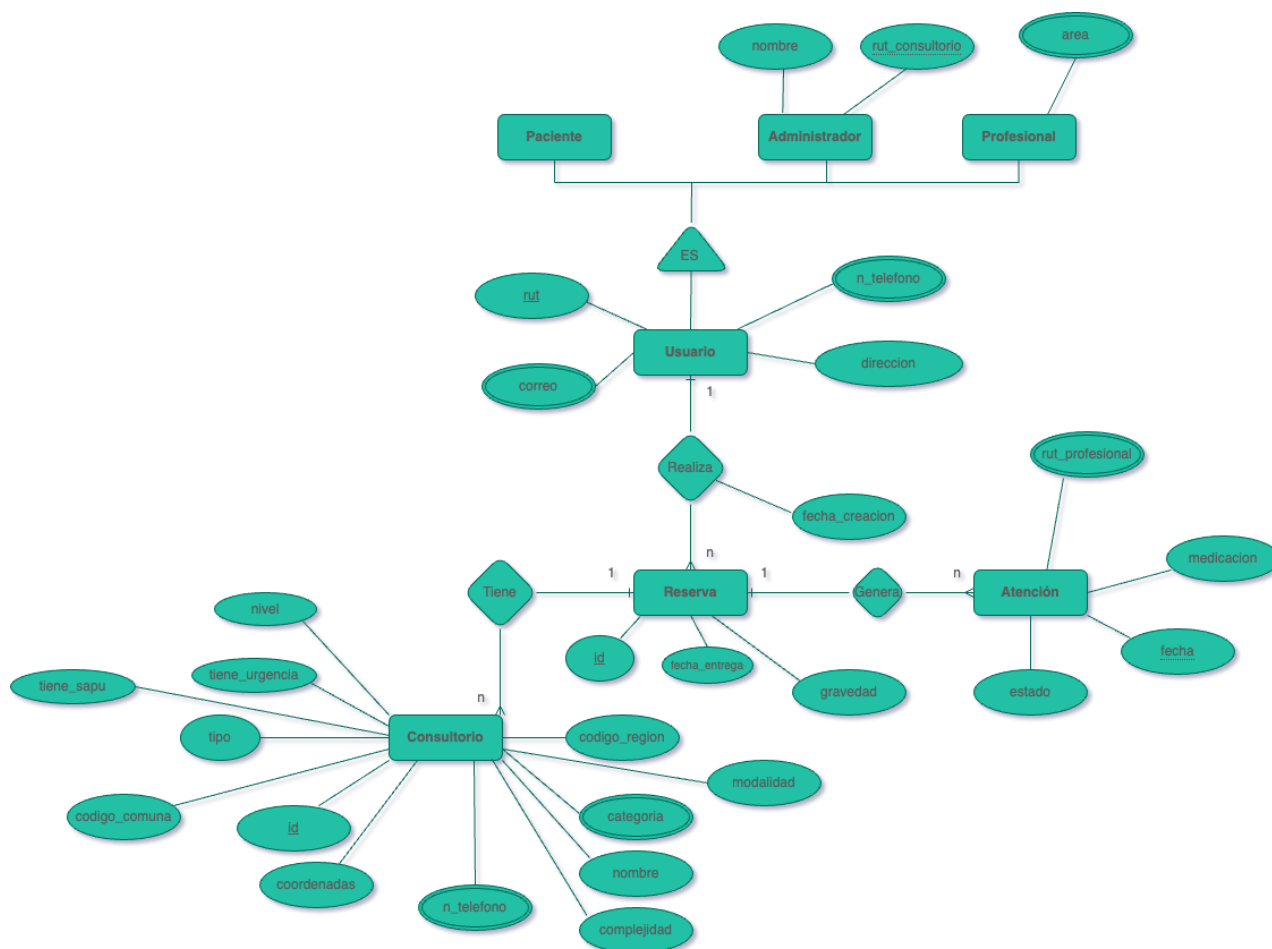
Action: [-----](#) Go 0 of 6 selected

	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	108814659	prueba@gmail.com	prueba		
<input type="checkbox"/>	202478816	dinamarcaalejandro@gmail.com	Alejandro Dinamarca		
<input type="checkbox"/>	212704962	prueba@gmail.com	prueba		
<input type="checkbox"/>	71403068	violeta_parra@gmail.com	Violeta Parra		
<input type="checkbox"/>	74396232	bdoarq@gmail.com	Bernardo Dinamarca		
<input type="checkbox"/>	97090408	fer.rossana@gmail.com	Rossana Cáceres		

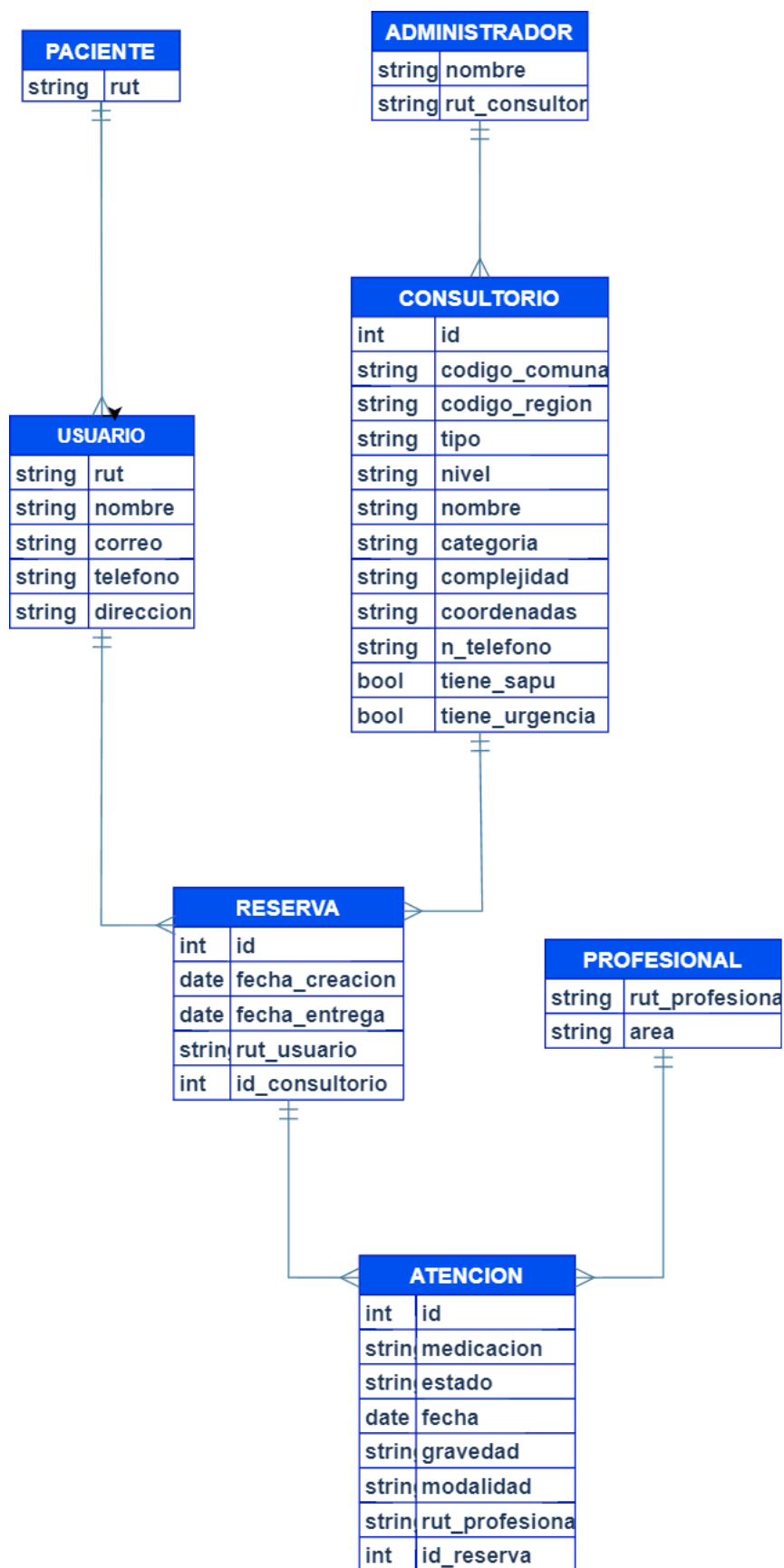
6 users

4 COMPONENTES BD

4.1 MODELO ENTIDAD RELACIÓN



4.2 MODELO RELACIONAL



4.3 DICCIONARIO DE DATOS

Tabla: CONSULTORIO

Nombre Atributo	PK/FK	Tipo de dato	NULL/NOT NULL	Valores por defecto	Descripción
id	PK	int	NOT NULL	AUTO_INCREMENT	Identificador único del consultorio.
codigo_comuna		string	NULL	Ninguno	Código de la comuna del consultorio.
codigo_region		string	NULL	Ninguno	Código de la región del consultorio.
tipo		string	NOT NULL	Ninguno	Tipo de consultorio.
nivel		string	NULL	Ninguno	Nivel de complejidad del consultorio.
nombre		string	NOT NULL	Ninguno	Nombre del consultorio.
categoria		string	NULL	Ninguno	Categoría del consultorio.
complejidad		string	NULL	Ninguno	Complejidad del consultorio.
coordenadas		string	NULL	Ninguno	Coordenadas geográficas del consultorio.
n_telefono		string	NULL	Ninguno	Número de teléfono del consultorio.
tiene_sapu		bool	NULL	Ninguno	Indica si el consultorio tiene servicio de SAPU.
tiene_urgencia		bool	NULL	Ninguno	Indica si el consultorio tiene servicio de urgencia.

Tabla: ATENCION

Nombre Atributo	PK/FK	Tipo de dato	NULL/NOT NULL	Valores por defecto	Descripción
id	PK	int	NOT NULL	AUTO_INCREMENT	Identificador único de la atención.
medicacion		string	NULL	Ninguno	Medicación prescrita durante la atención.
estado		string	NOT NULL	Ninguno	Estado actual de la atención.
fecha		date	NOT NULL	CURRENT_DATE	Fecha en la que se realizó la atención.
gravedad		string	NULL	Ninguno	Gravedad del caso atendido.
modalidad		string	NULL	Ninguno	Modalidad de atención.
rut_profesional	FK	string	NOT NULL	Ninguno	Profesional que supervisó la atención.
id_reserva	FK	int	NOT NULL	Ninguno	Reserva que generó la atención.

Tabla: RESERVA

Nombre Atributo	PK/FK	Tipo de dato	NULL/NOT NULL	Valores por defecto	Descripción
id	PK	int	NOT NULL	AUTO_INCREMENT	Identificador único de la reserva.
fecha_creacion		date	NOT NULL	CURRENT_DATE	Fecha de creación de la reserva.
fecha_entrega		date	NULL	Ninguno	Fecha en la que se entrega la reserva.
rut_usuario	FK	string	NOT NULL	Ninguno	Usuario que realizó la reserva.
id_consultorio	FK	int	NOT NULL	Ninguno	Consultorio donde se realiza la reserva.

Tabla: Usuario

Nombre Atributo	PK/FK	Tipo de dato	NULL/NOT NULL	Valores por defecto	Descripción
rut	PK	string	NOT NULL	Ninguno	Identificador único del usuario.
correo		string	NULL	Ninguno	Correo electrónico del usuario.
n_telefono		string	NULL	Ninguno	Número de teléfono del usuario.
direccion		string	NULL	Ninguno	Dirección de residencia del usuario.

Tabla: PROFESIONAL

Nombre Atributo	PK/FK	Tipo de dato	NULL/NOT NULL	Valores por defecto	Descripción
rut_profesional	PK	string	NOT NULL	Ninguno	Identificador único del profesional.
area		string	NOT NULL	Ninguno	Área de especialización del profesional.

Tabla: ADMINISTRADOR

Nombre Atributo	PK/FK	Tipo de dato	NULL/NOT NULL	Valores por defecto	Descripción
nombre	PK	string	NOT NULL	Ninguno	Nombre del administrador.
rut_consultorio	FK	string	NOT NULL	Ninguno	Consultorio al que está asociado el administrador.

Tabla: PACIENTE

Nombre Atributo	PK/FK	Tipo de dato	NULL/NOT NULL	Valores por defecto	Descripción
rut	PK	string	NOT NULL	Ninguno	Identificador único del paciente.

4.4 CONSULTAS EN ÁLGEBRA RELACIONAL

1. Listar pacientes por consultorio

```
SELECT p.nombre AS nombre_paciente,  
       c.nombre AS nombre_consultorio  
FROM   paciente p  
       LEFT JOIN reserva r  
         ON p.rut = r.rut  
       LEFT JOIN consultorio c  
         ON r.id_consultorio = c.id;
```

2. Listar los profesionales con citas

```
SELECT pr.nombre AS nombre_profesional,  
       a.fecha   AS fecha_atencion  
FROM   profesional pr  
       LEFT JOIN atencion a  
         ON pr.rut = a.rut;
```

3. Total de reservas en un consultorio

```
SELECT COUNT(*) AS total_reservas  
FROM   reserva  
WHERE  id_consultorio = 1;
```

1. $\pi_{nombre_{paciente}, nombre_{consultorio}} ((paciente \bowtie reserva) \bowtie consultorio)$

2. $\pi_{nombre_{profesional}, fecha_{atencion}} ((profesional \bowtie atencion))$

3. $\pi_{\sigma_{id_{consultorio}=1}} (reserva)$

5 BACKEND – API REST

5.1 INTRODUCCIÓN

El Backend de ConsultorioDigital ya se encuentra implementado haciendo uso de Django, un framework que facilita la creación de aplicaciones web de manera escalable. Esta API REST gestiona la lógica de negocio, autenticación y base de datos asociada al proyecto.

5.2 ENDPOINTS Y ARQUITECTURA

La API ofrece endpoints que permiten operaciones CRUD sobre las entidades ya definidas como usuarios, citas y centros asistenciales. Los endpoints facilitan el acceso a nuestros recursos por medio de métodos HTTP de uso estándar (POST, GET, DELETE Y PUT).

Pasando al aspecto de la implementación de los **puntos finales, métodos HTTP admitidos, parámetros de entrada, respuestas esperadas y códigos de estado** presentes en la solución desarrollada podemos encontrar:

5.2.1 Obtener Token de Acceso

- **Endpoint:** `/api/v1/token/`
- **Método:** POST
- **Descripción:** Devuelve un Token de acceso.
- **Parámetros:** `codigo_region`
 - `username:` (string) Nombre de usuario del usuario.
 - `password:` (string) Contraseña del usuario.
- **Respuesta:**
 - **Código de estado:** 200 OK
 - **Estructura de respuesta:**

```
[
  {
    "token": "string"
  },
  ...
]
```

- **Código de estado:** 400 Bad Request

```
{
  "error": "Credenciales inválidas"
},
...
]
```


5.2.2. Obtener Token de Refresco

- **Endpoint:** `/api/v1/token/refresh/`
- **Método:** POST
- **Descripción:** Devuelve un Token de refresco.
- **Parámetros:**
 - `refresh`: (string) Token de refresco.
- **Respuesta:**
 - **Código de estado:** 200 OK
 - **Estructura de respuesta:**

```
[  
  {  
    "access": "string"  
  },  
  ...  
]
```

5.2.3. Listar Regiones

- **Endpoint:** `/api/v1/region/`
- **Método:** GET
- **Descripción:** Devuelve una lista de consultorios para una comuna específica.
- **Parámetros:** Ninguno
- **Respuesta:**
 - **Código de estado:** 200 OK
 - **Estructura de respuesta:**

```
[  
  {  
    "c_reg": 1,  
    "nom_reg": "<Nombre de la Región>",  
    ...  
  },  
  ...  
]
```

5.2.4. Obtener Comunas por Región

- **Endpoint:** `/api/v1/comuna/<int:c_reg>/`
- **Método:** GET
- **Descripción:** Devuelve las comunas presentes en la región.
- **Parámetros:**
 - `c_reg`: (int) Código de la región.
- **Respuesta:**
 - **Código de estado:** 200 OK

- **Estructura de respuesta:**

```
{ [
  {
    "c_com": "Código Comuna",
    "nom_com": "Nombre de la Comuna"
  },
  ...
]
```

- **Código de estado:** 404 Not Found , Si no encuentran comunas.

2.5.5. Obtener Consultorios por Comuna

- **Endpoint:** /api/v1/consultorio/<str:c_com>/
- **Método:** GET
- **Parámetros:**
 - c_com: (int) Código de la comuna.
- **Respuesta:**
 - **Código de estado:** 201 Created si la reserva es exitosa.
 - **Estructura de respuesta:**

```
{
  {
    "c_com": "Código Comuna",
    "nombre": "Nombre del Consultorio",
    ...
  },
  ...
}
```

- **Código de estado:** 404 Not Found si no se encuentran consultorios.

2.5.6. Registro

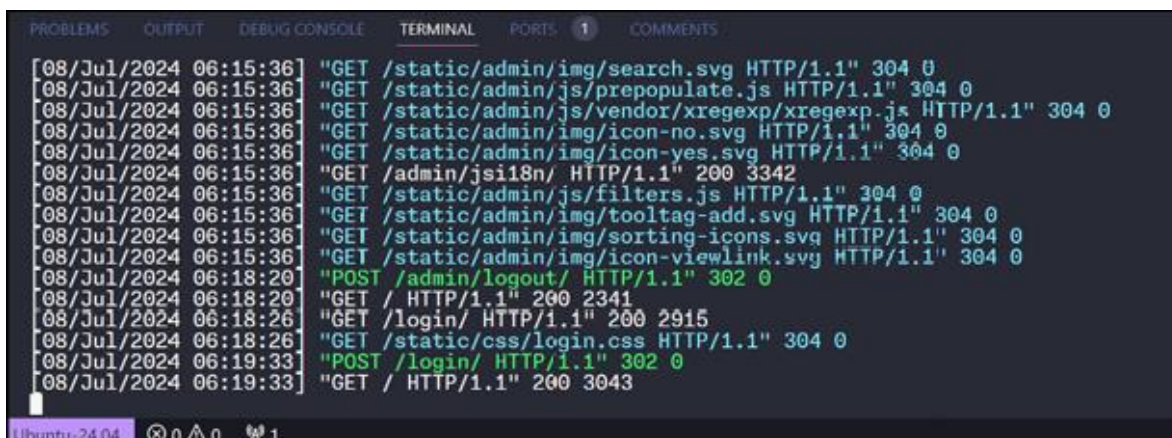
- **Endpoint:** /registro/
- **Método:** POST
- **Parámetros:**
 - c_com: (int) Código de la comuna.
- **Respuesta:**
 - **Código de estado:** 200 Created Respuesta HTML o JSON.
 - **Código de estado:** 400 Bad Request Mensaje de error si el registro falla.

2.5.7. Autenticación

- **Endpoint:** /accounts/login/
- **Método:** GET y POST
- **Parámetros:**
 - Al hacer POST:
 - username: (string) Nombre de usuario.
 - password: (string) Contraseña.
- **Respuesta:**
 - **Código de estado:** 200 Created Redirección a la página deseada tras inicio de sesión.
 - **Código de estado:** 401 Unauthorized Mensaje de error si las credenciales son incorrectas

Algunos ejemplos de cómo se ven distintos tipos de consultas:

- solicitud de crear usuario y, además, al ir a la página de inicio, cargando correctamente los datos de los usuarios con el código 302 de "Encontrado" o "Found" (*POST /registro/ HTTP/1.1" 302 0*):

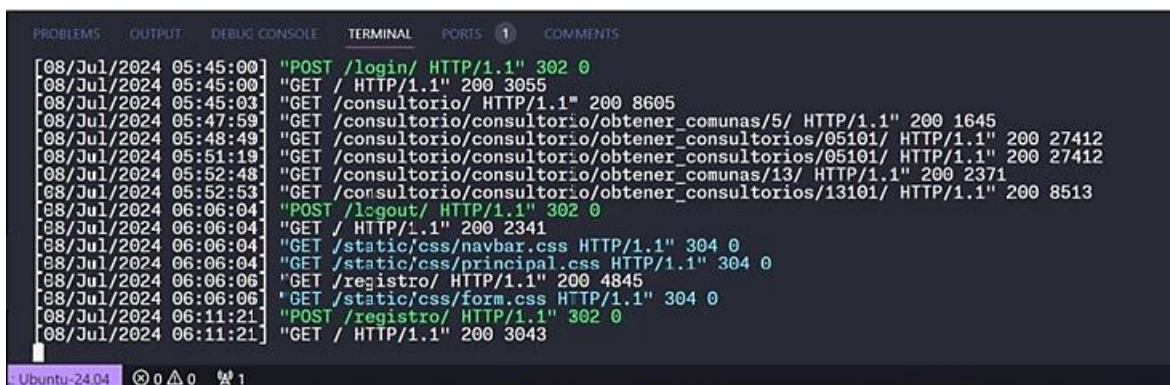


```

[08/Jul/2024 06:15:36] "GET /static/admin/img/search.svg HTTP/1.1" 304 0
[08/Jul/2024 06:15:36] "GET /static/admin/js/prepopulate.js HTTP/1.1" 304 0
[08/Jul/2024 06:15:36] "GET /static/admin/js/vendor/xregexp/xregexp.js HTTP/1.1" 304 0
[08/Jul/2024 06:15:36] "GET /static/admin/img/icon-no.svg HTTP/1.1" 304 0
[08/Jul/2024 06:15:36] "GET /static/admin/img/icon-yes.svg HTTP/1.1" 304 0
[08/Jul/2024 06:15:36] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[08/Jul/2024 06:15:36] "GET /static/admin/js/filters.js HTTP/1.1" 304 0
[08/Jul/2024 06:15:36] "GET /static/admin/img/tooltag-add.svg HTTP/1.1" 304 0
[08/Jul/2024 06:15:36] "GET /static/admin/img/sorting-icons.svg HTTP/1.1" 304 0
[08/Jul/2024 06:15:36] "GET /static/admin/img/icon-viewlink.svg HTTP/1.1" 304 0
[08/Jul/2024 06:18:20] "POST /admin/logout/ HTTP/1.1" 302 0
[08/Jul/2024 06:18:20] "GET / HTTP/1.1" 200 2341
[08/Jul/2024 06:18:26] "GET /login/ HTTP/1.1" 200 2915
[08/Jul/2024 06:18:26] "GET /static/css/login.css HTTP/1.1" 304 0
[08/Jul/2024 06:19:33] "POST /login/ HTTP/1.1" 302 0
[08/Jul/2024 06:19:33] "GET / HTTP/1.1" 200 3043
  
```

Figura 1

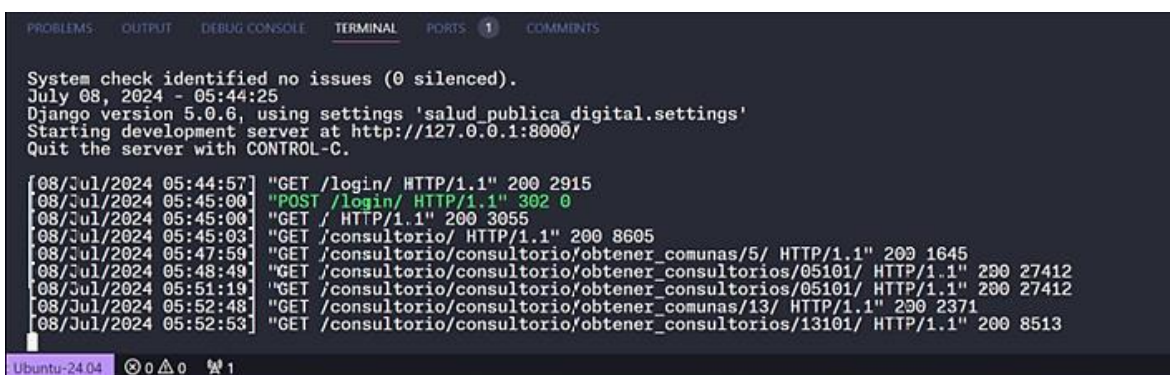
- Inicio de sesión con la solicitud "POST login" en la consola de Django con código 200 (OK):



```
[08/Jul/2024 05:45:00] "POST /login/ HTTP/1.1" 302 0
[08/Jul/2024 05:45:00] "GET / HTTP/1.1" 200 3055
[08/Jul/2024 05:45:03] "GET /consultorio/ HTTP/1.1" 200 8605
[08/Jul/2024 05:47:59] "GET /consultorio/consultorio/obtener_comunas/5/ HTTP/1.1" 200 1645
[08/Jul/2024 05:48:49] "GET /consultorio/consultorio/obtener_consultorios/05101/ HTTP/1.1" 200 27412
[08/Jul/2024 05:51:19] "GET /consultorio/consultorio/obtener_consultorios/05101/ HTTP/1.1" 200 27412
[08/Jul/2024 05:52:48] "GET /consultorio/consultorio/obtener_comunas/13/ HTTP/1.1" 200 2371
[08/Jul/2024 05:52:53] "GET /consultorio/consultorio/obtener_consultorios/13101/ HTTP/1.1" 200 8513
[08/Jul/2024 06:06:04] "POST /logout/ HTTP/1.1" 302 0
[08/Jul/2024 06:06:04] "GET / HTTP/1.1" 200 2341
[08/Jul/2024 06:06:04] "GET /static/css/navbar.css HTTP/1.1" 304 0
[08/Jul/2024 06:06:04] "GET /static/css/principal.css HTTP/1.1" 304 0
[08/Jul/2024 06:06:06] "GET /registro/ HTTP/1.1" 200 4845
[08/Jul/2024 06:06:06] "GET /static/css/form.css HTTP/1.1" 304 0
[08/Jul/2024 06:11:21] "POST /registro/ HTTP/1.1" 302 0
[08/Jul/2024 06:11:21] "GET / HTTP/1.1" 200 3043
```

Figura 2

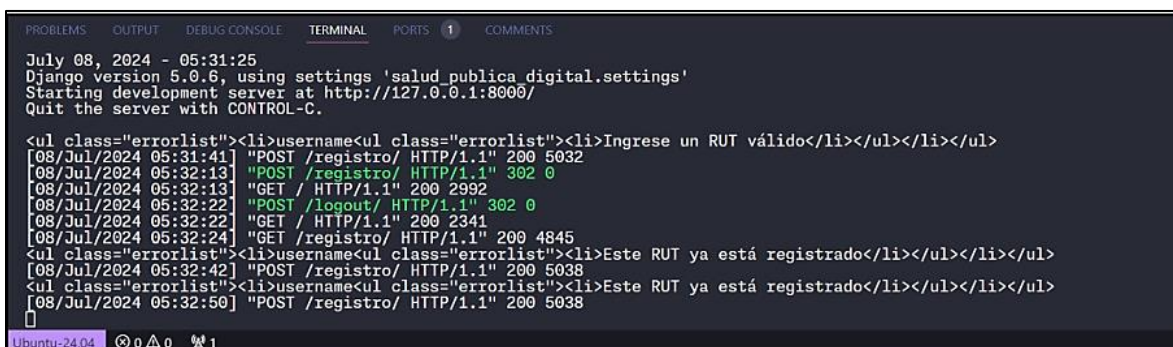
- Peticiones internas al momento de buscar consultorios disponibles, notar que, de igual forma es posible acceder a los datos mediante una petición en el navegador, que retorna un JSON con los datos de la comuna en cuestión:



```
System check identified no issues (0 silenced).
July 08, 2024 - 05:44:25
Django version 5.0.6, using settings 'salud_publica.digital.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

[08/Jul/2024 05:44:57] "GET /login/ HTTP/1.1" 200 2915
[08/Jul/2024 05:45:00] "POST /login/ HTTP/1.1" 302 0
[08/Jul/2024 05:45:00] "GET / HTTP/1.1" 200 3055
[08/Jul/2024 05:45:03] "GET /consultorio/ HTTP/1.1" 200 8605
[08/Jul/2024 05:47:59] "GET /consultorio/consultorio/obtener_comunas/5/ HTTP/1.1" 200 1645
[08/Jul/2024 05:48:49] "GET /consultorio/consultorio/obtener_consultorios/05101/ HTTP/1.1" 200 27412
[08/Jul/2024 05:51:19] "GET /consultorio/consultorio/obtener_consultorios/05101/ HTTP/1.1" 200 27412
[08/Jul/2024 05:52:48] "GET /consultorio/consultorio/obtener_comunas/13/ HTTP/1.1" 200 2371
[08/Jul/2024 05:52:53] "GET /consultorio/consultorio/obtener_consultorios/13101/ HTTP/1.1" 200 8513
```

Figura 3



5.3. AUTENTICACIÓN Y AUTORIZACIÓN

Todos los endpoints que requieren autenticación deben incluir un token de sesión o un mecanismo de autenticación, esto es implementado tal que:

Autenticación

- **Autenticación:** Todos los endpoints que requieren autenticación deben recibir un.
 - **Proceso de Autenticación:**
 1. El usuario envía sus credenciales (usuario y contraseña) al endpoint de login.
 2. Si las credenciales son válidas, el servidor responde con un **token de acceso** (JWT).
 3. El cliente puede acceder a los apartados a los cuales está autorizado.
- **Expiración y Renovación del Token:** Define un tiempo de expiración para el token y utiliza un sistema de renovación para sesiones prolongadas, minimizando el riesgo de acceso no autorizado.
- **Verificación de Permisos:** Antes de procesar cada solicitud, el servidor verifica si el usuario tiene permisos para acceder al recurso solicitado. Esto asegura que un usuario solo pueda ver o modificar datos a los que está autorizado.
- **Ejemplo de Autenticación y Autorización**
 1. **Login:**
 - El cliente envía una solicitud POST /api/login con las credenciales en el cuerpo.
 - El servidor responde con un token JWT.
 2. **Acceso a un Endpoint Protegido:**
 - El cliente realiza una solicitud GET /api/citas e incluye el token en la cabecera.
 - El servidor verifica el token y los permisos del usuario antes de responder.

5.4. ERRORES

Podemos documentar los códigos/mensajes de error que se encuentran dentro de nuestra API REST de manera que se tienen mensajes propios del framework que dirigen a mensajes personalizados que son almacenados dentro de los propios formularios creados, tal que:

- **400 Bad Request:** La solicitud del cliente contiene datos inválidos/faltantes, esto indica la falta de datos en campos obligatorios o datos que no cumplen con validaciones.
 - **Ejemplos de mensajes:**
 - "Ingrese un RUT válido"
 - "Ingrese un correo válido"
 - "Este campo es obligatorio"
- **401 Unauthorized:** El cliente no está autenticado o el token de acceso es inválido, esto debido a credenciales invalidas. Este error se utiliza en endpoints de autorización.
 - **Ejemplos de mensajes:**
 - "Token de acceso no proporcionado"
 - "Token de acceso inválido"
 - "Token de acceso expirado" (error más frecuente)
 - "Autenticación fallida"
- **403 Forbidden:** El cliente no tiene los permisos necesarios para acceder al recurso solicitado, esto al incumplir con los permisos asociados al usuario autenticado, esto al intentar acceder a un recurso al que no posee permisos de acceso.
 - **Ejemplos de mensajes:**
 - "No tiene permiso para acceder a este recurso"
 - "Acceso denegado"
- **404 Not Found:** El recurso solicitado no existe, esto al solicitar un recurso inexistente.
 - **Ejemplos de mensajes:**
 - "El recurso solicitado no fue encontrado"

- **"Consultorio no encontrado"**
- **409 Conflict:** Ocurre al tener conflicto de datos, puede ser útil para manejar los mismos.
 - **Ejemplos de mensajes:**
 - **"Ya existe una reserva para el mismo paciente en esta fecha"**
 - **"El RUT ya está registrado en el sistema"**
 - **"Conflicto en los datos proporcionados"**
- **500 Internal Server Error:** Ocurrió un error inesperado en el servidor, esto al existir un fallo en la base de datos o lógica implementada.
 - **Ejemplos de mensajes:**
 - **"Ha ocurrido un error en el servidor. Inténtelo de nuevo más tarde."**
 - **"Error interno. Por favor, contacte al soporte."**

Dentro del repositorio de GitHub es posible encontrar todos los formularios personalizados que fueron creados, estos funcionan en concurrencia con los mensajes predeterminados y pueden ser vistos en el siguiente link:
https://github.com/adinamarca/salud_publica_digital/blob/main/registro/forms.py

5.5. EJEMPLOS

Los ejemplos de solicitudes y sus respuestas para los endpoints pueden ser encontrados en las figuras 1-5 presentas anteriormente.

5.2. CONSIDERACIONES DE SEGURIDAD

Garantizar la seguridad de los datos manejados y la comunicación es una de las principales preocupaciones a futuro en el desarrollo de ConsultorioDigital, esto debido a las extensas exigencias legales que regulan el uso de información privilegiada de los pacientes a atender, esto genera una necesidad de consideraciones de carácter más avanzado.

Por tanto, el manejo de los datos privados del usuario debe ser seguro y solo ofrecido al personal de salud que atiende al paciente en cuestión. Esto es implementado de la siguiente manera:

1. **Autenticación y Autorización:**

- Implementa autorización basada en roles para limitar el acceso a datos sensibles.

2. **Control de Acceso:**

- Limita los accesos a personal autorizado y registra todas las actividades de los usuarios.

3. **Protección contra Ataques Comunes:**

- Usa validación estricta y sanitización de datos de entrada.

4. **Gestión de Sesiones:**

- Define tiempos de expiración de tokens de autenticación y usa renovación de tokens para mitigar accesos prolongados.
- Implementa la revocación de tokens para usuarios que cambian credenciales o presentan actividades sospechosas.

6 ACTUALIZACIÓN DE CONSULTAS A NoSQL

1. Listar pacientes por consultorio:

```
{db.pacientes.aggregate([
  {
    $lookup: {
      from: "reserva",
      localField: "rut",
      foreignField: "rut",
      as: "reserva"
    }
  },
  {
    $lookup: {
      from: "consultorio",
      localField: "reserva.id_consultorio",
      foreignField: "id",
      as: "consultorio"
    }
  },
  {
    $project: {
      "nombre_paciente": "$nombre",
      "nombre_consultorio": "$consultorio.nombre"
    }
  }
])
```

2. Listar los profesionales con citas:

```
{db.pacientes.aggregate([
  {
    $lookup: {
      from: "atencion",
      localField: "rut",
      foreignField: "rut",
      as: "atencion"
    }
  },
  {
    $project: {
      "nombre_profesional": "$nombre",
      "fecha_atencion": "$atencion.fecha"
    }
  }
])
```

3. Total de reservas en un consultorio:

```
{db.pacientes.aggregate([
  id_consultorio: 1
])
```

7 MODELO NoSQL

