

Universidad Interamericana de Puerto Rico Recinto de Arecibo

Departamento de Ciencias y Tecnología

Programa de Ciencias de Computadoras

Sistema de Gestión de Citas Médicas

COMP2052 – Web Development: Server Side & Microservices

Mikael Hashem K. Tiba (R00627615)

Janiel O. Rodríguez Velázquez (R00622992)

Descripción general del proyecto:

Este proyecto consiste en un sistema web de gestión de citas médicas desarrollado con Flask y MySQL. Permite a los usuarios registrarse e iniciar sesión, asignar roles como Administrador, Médico o Paciente, y realizar operaciones como agendar, cancelar y marcar citas como realizadas. Además, se implementaron pruebas de los endpoints con archivos .rest para validar el funcionamiento del CRUD completo.

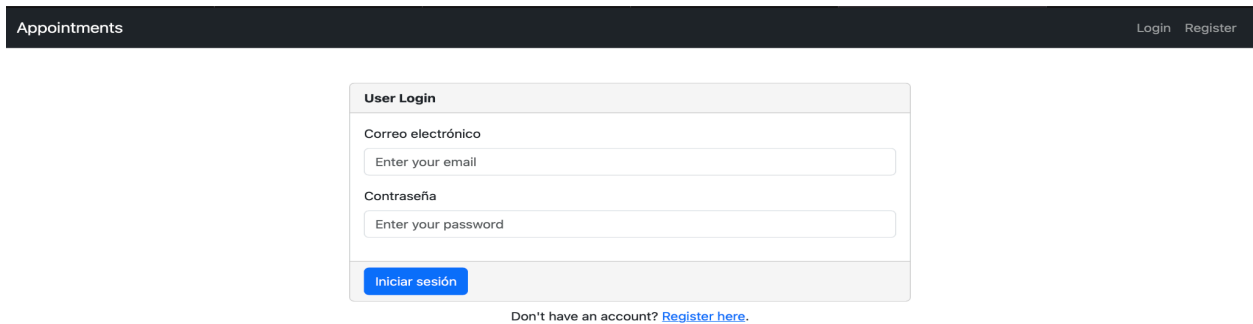
Funcionalidades:

- Registro y autenticación de usuarios con Flask-Login.
- Asignación de roles: Administrador, Médico y Paciente.
- Agendamiento de citas por parte de pacientes, seleccionando fecha y médico.
- Cancelación de citas por pacientes, médicos o administradores.
- Marcado de citas como realizadas por médicos o administradores.
- Panel de control (dashboard) personalizado según el rol del usuario.
- Listado de usuarios (solo accesible para administradores).
- Cambio de contraseña seguro para cualquier usuario.
- Pruebas CRUD con endpoints REST (create, read, update, delete) usando archivos .rest.

Capturas de Pantalla:

Captura del login:

Permite a los usuarios acceder al sistema autenticándose con su correo electrónico y contraseña previamente registrados.



The screenshot displays a web interface for a system named "Appointments". At the top, a dark navigation bar contains the title "Appointments" on the left and links for "Login" and "Register" on the right. The main content area features a "User Login" form. This form includes two input fields: "Correo electrónico" (Email) with the placeholder text "Enter your email", and "Contraseña" (Password) with the placeholder text "Enter your password". Below these fields is a blue button labeled "Iniciar sesión" (Log in). At the bottom of the form, there is a link that reads "Don't have an account? [Register here.](#)".

Dashboard de admin:

Vista exclusiva para administradores. Muestra todas las citas agendadas del sistema y acceso al listado completo de usuarios.

Appointments						Dashboard	Users	Change Password	Logout
Gestión de Citas Médicas									
Fecha y Hora		Médico	Paciente	Motivo	Estado	Acciones			
12/05/2025 21:09		Dra Ana Torres	Carlos Pérez	Dolor de cabeza	Cancelada	🗑			

Listado de usuarios:

Desde esta vista, el administrador puede visualizar todos los usuarios registrados, sus roles y correos electrónicos.

Appointments						Dashboard	Users	Change Password	Logout
List of Registered Users									
Username		Email			Role				
Dra Ana Torres		medico@clinica.com			Médico				
Carlos Pérez		carlos@correo.com			Paciente				
Administrador General		admin@clinica.com			Admin				

Dashboard de paciente, aquí nos permite agendar citas y escribir el motivo:

El paciente puede ver sus citas, agendar nuevas, cancelar las existentes y consultar el motivo de cada una.

Appointments

DashboardChange PasswordLogout

Agendar Nueva Cita

Médico

Dra Ana Torres

Fecha y Hora

05/17/2025, 08:00 AM

Motivo

Dolor de cabeza recurrente.

Agendar

Formulario que permite seleccionar un médico, una fecha y un motivo. Solo accesible para usuarios con rol de paciente. Después de enviar el formulario, la cita se muestra en el dashboard como confirmada con estado “Agendada”. También se habilita la opción para cancelarla.

Appointments

DashboardChange PasswordLogout

Cita agendada exitosamente.

Gestión de Citas Médicas

⊕ Agendar nueva cita

Fecha y Hora	Médico	Paciente	Motivo	Estado	Acciones
17/05/2025 08:00	Dra Ana Torres	Carlos Pérez	Dolor de cabeza recurrente.	Agendada	<div>⊖ Cancelar</div>

Dashboard del médico:

El médico puede ver las citas que le han sido asignadas. Tiene la opción de marcarlas como “Realizada” o “Cancelada” según corresponda.

Appointments

DashboardChange PasswordLogout

Gestión de Citas Médicas

Fecha y Hora	Médico	Paciente	Motivo	Estado	Acciones
17/05/2025 08:00	Dra Ana Torres	Carlos Pérez	Dolor de cabeza recurrente.	Agendada	<div><div>⌛ Cancelar</div><div>✅ Realizar</div></div>

Cuando el médico hace clic en “Marcar como realizada”, el estado de la cita cambia y aparece un mensaje de confirmación.

Gestión de Citas Médicas

Fecha y Hora	Médico	Paciente	Motivo	Estado	Acciones
17/05/2025 08:00	Dra Ana Torres	Carlos Pérez	Dolor de cabeza recurrente.	Realizada	<div>🔒</div>

Pruebas con .rest:

Crear cita:

```
final_project > pruebas > create.rest > POST /test/citas

Send Request
1 POST http://localhost:5001/test/c
2 Content-Type: application/json
3
4 {
5     "fecha_hora": "2025-05-18T10:30
6     "motivo": "Chequeo general",
7     "medico_id": 1,
8     "paciente_id": 5
9 }

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.12.
3 Date: Sat, 17 May 2025 00:35:27 GMT
4 Content-Type: application/json
5 Content-Length: 45
6 Connection: close
7
8 {
9     "message": "Cita creada correctam
10 }
```

Obtener diferentes citas agendadas:

```
EXPLORER  read-a-row.rest M  read.rest M  Response(9ms) X
OPEN EDITORS  _project > pruebas > read.rest > GET /test/citas
1 GET http://localhost

delete.res... M
update.re... M
read-a-ro... M
read.rest... M

GROUP 2
X Response(9ms)
PROJECTOFINAL2052
final_project
__pycache__
app
__pycac...
appointment...
__pycache__
__init__.py
routes.py
auth_service
static
templates
__init__.py M
forms.py
models.py
test_rout... M
database_sche...
pruebas
create.rest M
delete.rest M
read-a-r... M
read.rest M
update.r... M
venv
.prettierrignore
config.py
create_demo_u...
requirements.txt
run.py
README.md

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.12.2
3 Date: Sat, 17 May 2025 00:50:48 GMT
4 Content-Type: application/json
5 Content-Length: 342
6 Connection: close
7
8 [
9     {
10         "estado": "Realizada",
11         "fecha_hora": "2025-05-17T08:00:00",
12         "id": 2,
13         "medico_id": 1,
14         "motivo": "Dolor de cabeza recurrente.",
15         "paciente_id": 5
16     },
17     {
18         "estado": "Agendada",
19         "fecha_hora": "2025-05-18T10:30:00",
20         "id": 7,
21         "medico_id": 1,
22         "motivo": "Chequeo general",
23         "paciente_id": 5
24     }
25 ]
```

Obtener una cita en específico:

The screenshot shows a Visual Studio Code editor with a REST client configuration and its response. The Explorer sidebar on the left shows a project structure with folders like 'app', 'auth_service', 'static', 'templates', 'forms.py', 'models.py', 'test_routes.py', 'database_schema', 'pruebas', 'venv', and 'README.md'. The 'pruebas' folder is expanded, showing several REST client files. The 'read.rest' file is selected, showing a GET request to 'http://localhost:5001/test/citas/7' with a 'Content-Type: application/json' header. The 'Response(5ms)' tab on the right shows the response details: HTTP/1.1 200 OK, Server: Werkzeug/3.1.3 Python/3.12.2, Date: Sat, 17 May 2025 01:18:30 GMT, Content-Type: application/json, Content-Length: 146, and Connection: close. The response body is a JSON object: { "estado": "Agendada", "fecha_hora": "2025-05-18T10:30:00", "id": 7, "medico_id": 1, "motivo": "Chequeo general", "paciente_id": 5 }. The bottom panel shows the 'TERMINAL' tab with a log of REST client requests and responses, including PUT, GET, and DELETE requests to various endpoints.

```
1 ## Obtener una cita especifica por ID
2 Send Request
3 GET http://localhost:5001/test/citas/7
4 Content-Type: application/json
```

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.12.2
3 Date: Sat, 17 May 2025 01:18:30 GMT
4 Content-Type: application/json
5 Content-Length: 146
6 Connection: close
7
8 {
9   "estado": "Agendada",
10  "fecha_hora": "2025-05-18T10:30:00",
11  "id": 7,
12  "medico_id": 1,
13  "motivo": "Chequeo general",
14  "paciente_id": 5
15 }
```

```
127.0.0.1 - - [16/May/2025 21:09:12] "PUT /test/citas/1 HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2025 21:09:15] "GET /test/citas/1 HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2025 21:09:16] "GET /test/citas/1 HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2025 21:12:09] "GET /test/citas/1 HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2025 21:12:15] "GET /test/citas HTTP/1.1" 200 -
127.0.0.1 - - [16/May/2025 21:12:45] "GET /test/citas HTTP/1.1" 200 -
127.0.0.1 - - [16/May/2025 21:14:21] "GET /test/citas HTTP/1.1" 200 -
127.0.0.1 - - [16/May/2025 21:16:56] "DELETE /test/citas/2 HTTP/1.1" 200 -
127.0.0.1 - - [16/May/2025 21:18:11] "GET /test/citas/2 HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2025 21:18:30] "GET /test/citas/7 HTTP/1.1" 200 -
```


Borrar cita:

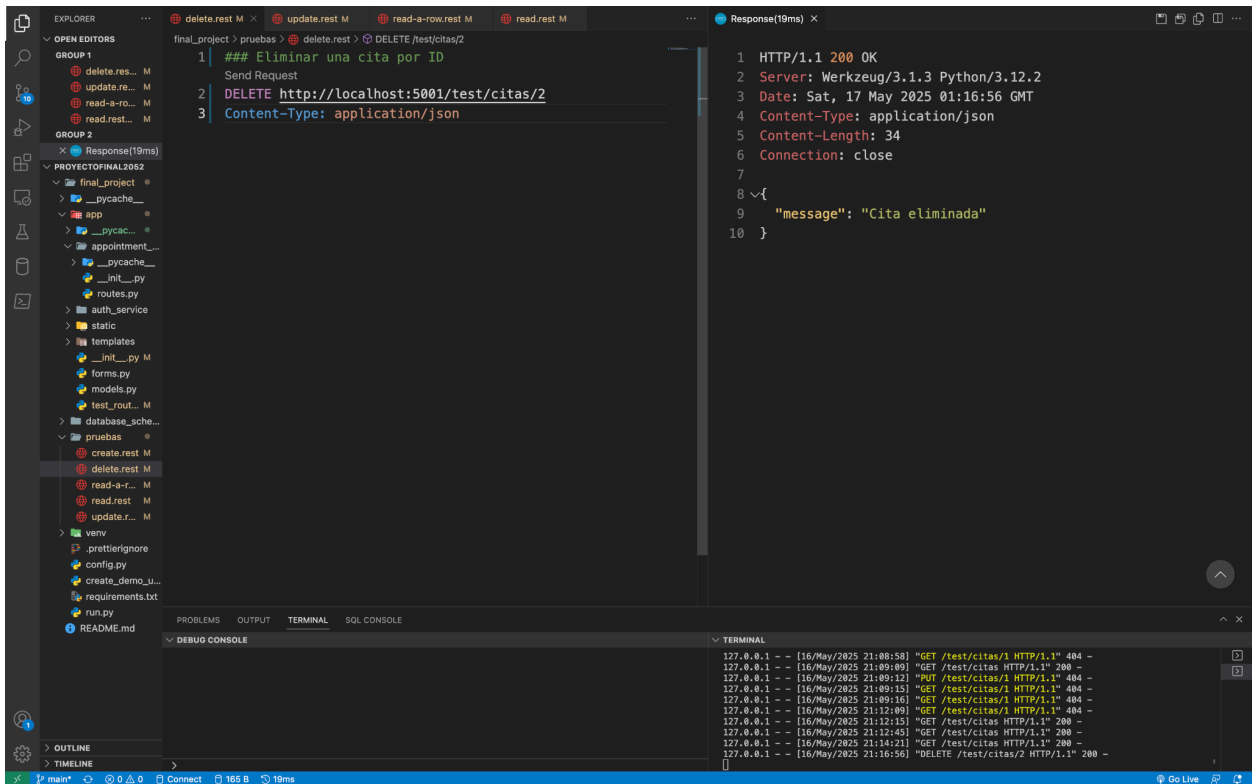


Tabla:

Tipo de prueba	Método HTTP	Ruta	Datos enviados / parámetros	Resultado esperado	Resultado obtenido
Crear cita	POST	/test/citas	JSON con datos válidos	Cita creada con status 200	Cita creada
Leer todas	GET	/test/citas	-	Lista de citas en JSON	Lista obtenida
Leer por ID	GET	/test/citas/2	-	Datos de la cita con ID 2	Cita encontrada
Actualizar cita	PUT	/test/citas/2	JSON con nuevo estado y motivo	Cita actualizada correctamente	Cita actualizada
Eliminar cita	DELETE	/test/citas/2	-	Cita eliminada con status 200	Cita eliminada
Error de foreign	POST	/test/citas	paciente_id no existente	Error 500 (foreign key error)	Error recibido
Ruta no válida	DELETE	/test/cita/999	-	Error 404 (ruta no encontrada)	Error recibido

Endpoints del CRUD principal:

Create:

```
@appointments.route('/agendar', methods=['GET', 'POST'])
```

Solo los usuarios autenticados pueden acceder a este endpoint:

```
@login_required
```

Solo los usuarios que sean pacientes pueden usar esta función. Si otro tipo de usuario intenta acceder (por ejemplo, un médico o admin), se les redirige al dashboard con un mensaje de error.

```
if current_user.role.name != 'Paciente':  
  
    flash('Solo los pacientes pueden agendar citas.')  
  
    return redirect(url_for('appointments.dashboard'))
```

Se crea una instancia del formulario definido en AgendarCitaForm

```
form = AgendarCitaForm()
```

Se consulta la base de datos para obtener todos los usuarios con rol 'Médico'. Estos médicos se cargan como opciones en el campo SelectField del formulario, permitiendo al paciente seleccionar con quién agendar la cita.

```
medicos = User.query.join(User.role).filter_by(name='Médico').all()

form.medico_id.choices = [(m.id, m.username) for m in medicos]
```

Si el formulario pasa la validación se crea una cita. Se asigna automáticamente el paciente como el usuario actual (current_user.id) y el estado de la cita como 'Agendada'.

```
if form.validate_on_submit():

    cita = Cita(

        fecha_hora=form.fecha_hora.data,

        motivo=form.motivo.data,

        medico_id=form.medico_id.data,

        paciente_id=current_user.id,

        estado='Agendada'

    )

    db.session.add(cita)

    db.session.commit()

    flash('Cita agendada exitosamente.')

    return redirect(url_for('appointments.dashboard'))
```

Si se accede mediante GET o el formulario no es válido, se muestra la plantilla HTML con el formulario de agendar cita.

```
return render_template('agendar_cita.html', form=form)
```

Read:

```
@appointments.route('/usuarios')
```

Solo los usuarios autenticados pueden acceder a este endpoint:

```
@login_required
```

Solo los admins pueden acceder a este endpoint.

```
if current_user.role.name != 'Admin':  
  
    flash("No tienes permiso para acceder a esta página.")  
  
    return redirect(url_for('appointments.dashboard'))
```

Se consulta en la base de datos todos los usuarios con su rol para mostrarlos.

```
usuarios = User.query.join(User.role).all()
```

Se renderiza la plantilla HTML usuarios.html enviando la lista de usuarios consultada para mostrarla en la interfaz.

```
return render_template('usuarios.html', usuarios=usuarios)
```

Update:

```
@appointments.route('/cambiar-password', methods=['GET', 'POST'])
```

Solo los usuarios autenticados pueden acceder a este endpoint:

```
@login_required
```

Se crea una instancia del formulario ChangePasswordForm().

```
form = ChangePasswordForm()
```

Se valida que cumpla con las características requeridas.

```
if form.validate_on_submit():
```

Se compara la contraseña actual introducida por el usuario con la que está guardada en la base de datos. Si no coincide, se muestra un mensaje de error y se vuelve a mostrar el formulario.

```
if not current_user.check_password(form.old_password.data):  
  
    flash('Contraseña actual incorrecta.')  
  
    return render_template('cambiar_password.html',  
form=form)
```

Se actualiza la contraseña, se actualiza en la base de datos, envía un mensaje de retroalimentación y redirige al dashboard.

```
current_user.set_password(form.new_password.data)  
  
db.session.commit()  
  
flash('Contraseña actualizada.')  
  
return redirect(url_for('appointments.dashboard'))
```

Si se accede con GET o si el formulario no es válido, se muestra el formulario para cambiar la contraseña.

```
return render_template('cambiar_password.html', form=form)
```

Delete:

```
@appointments.route('/agendar', methods=['GET', 'POST'])
```

Solo los usuarios autenticados pueden acceder a este endpoint:

```
@login_required
```

Se busca la cita por su ID, si no se encuentra el sistema lanza un error.

```
cita = Cita.query.get_or_404(id)
```

Solo se puede cancelar la cita si el usuario es admin, si el usuario que la agendó es el paciente y si el usuario es el médico asignado a la cita.

```
if current_user.role.name == 'Admin' or \
    cita.paciente_id == current_user.id or \
    cita.medico_id == current_user.id:
```

En vez de eliminar la cita se hace un soft delete, el estado de la cita cambia de agendada a cancelada.

```
cita.estado = 'Cancelada'
```



```
db.session.commit()

flash('Cita cancelada.')
```

Si el usuario no tiene los permisos le saldrá el siguiente mensaje:

```
else:

    flash('No tienes permiso para cancelar esta cita.')
```

Redirige al dashboard.

```
return redirect(url_for('appointments.dashboard'))
```

Github (Mikael): <https://github.com/mikatiba/proyectoFinal2052>

Github (Janiel): <https://github.com/Janielrodz/proyectoFinal2052>