Organización de citas y consultas

Nombre del autor: Andy Choque Cabrera
Fecha:03/05/2024

Introducción:

Esta es una aplicación de hospital donde organizamos consultas para concretar cita médica o consulta con algún médico en específico usando un código personal que te genera la aplicación.

Herramientas y Métodos:

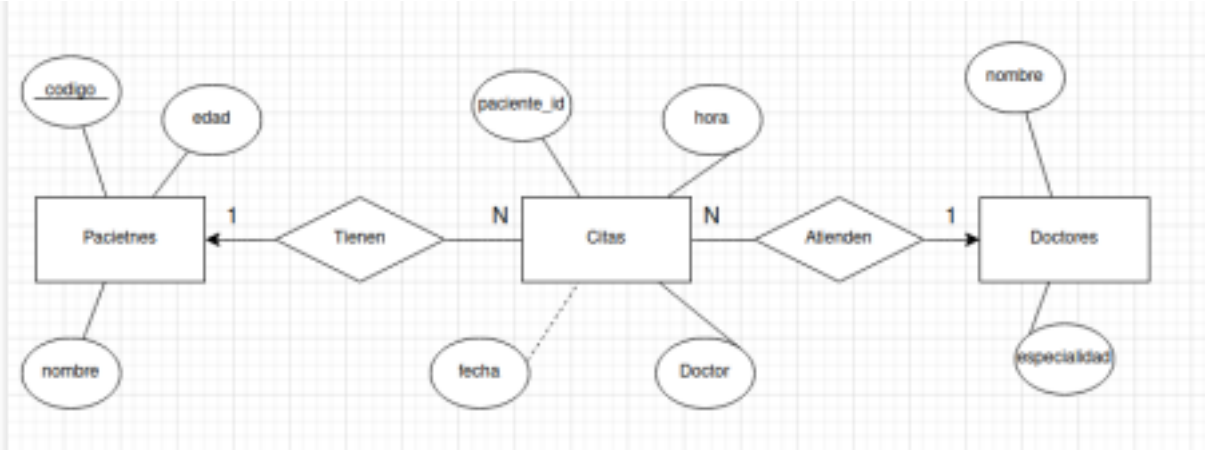
Se ha llevado a cabo el proyecto usando
- Python
- Visual Studio Code
- Chat GPT
- Youtube
- MongoDB (Server)
- pymongo

Explicación de los métodos empleados para llevar a cabo el proyecto:

He usado el MongoDB Server para usarlo de servidor en una maquina virtual y hacer su respectiva configuración. Luego en el cliente he instalado el Visual Studio Code para programar el fichero python ahi, tambien desdes google he descargado python y tambien por comandos el pymongo, luego he importado el mongoClient para simular que es el cliente desde otra maquina virtual y conectarme al servidor.
He usado tanto chat GPT y Youtube para hacer consultas especificas como la configuración de mongoDB Server o la de python.

Perspectiva Estática

Diagramas que representan la estructura de las entidades y sus relaciones:



Pasos a Tablas:

Detalles sobre cómo se transformaron los diagramas E/R en tablas de base de datos. En lo personal yo hize primero las tablas entonces lo que hize fue de la idea establecer las áreas que desarrolaria en mi proyecto es decir los médicos, los pacientes y las citas médicas.

DDL (Data Definition Language)

Ejemplos de código que definen la estructura de la base de datos. Pondré unas pocas líneas:

```
# Crear colecciones si no existen
if 'pacientes' not in db.list_collection_names():
    db.create_collection('pacientes')
if 'doctores' not in db.list_collection_names():
    db.create_collection('doctores')
```

DML (Data Manipulation Language)

Ejemplos de código que manipulan los datos en la base de datos

Pondré unas pocas líneas:

```
# Insertar un nuevo paciente en la colección 'pacientes'
db.pacientes.insert_one({"codigo": codigo, "nombre": nombre, "edad": edad})

# Insertar una nueva cita en la colección 'citas'
db.citas.insert_one({"paciente_id": paciente_id, "fecha": fecha, "hora": hora, "doctor":
doctor})

# Borrar una cita específica de la colección 'citas'
db.citas.delete_one({"paciente_id": paciente_id, "fecha": fecha, "hora": hora})
```

DQL (Data Query Language)

Ejemplos de código que consultan la base de datos.

```
# Buscar un paciente en la colección 'pacientes' basado en su código
paciente = db.pacientes.find_one({"codigo": codigo})

# Obtener todos los doctores disponibles en la colección 'doctores'
doctores = db.doctores.find()

# Buscar todas las citas de un paciente en la colección 'citas' basado en su ID de paciente
citas = db.citas.find({"paciente_id": paciente_id})
```

DCL (Data Control Language)

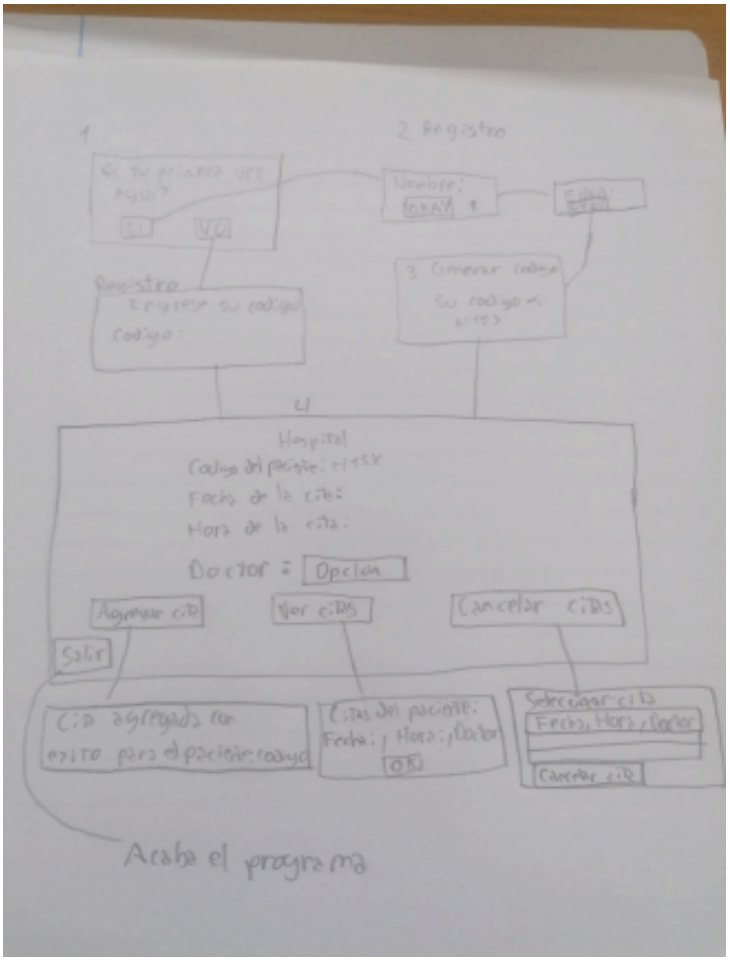
Ejemplos de código que controlan los permisos y la seguridad de la base de datos. Pondré unas pocas líneas

Crear un nuevo usuario con el rol de lectura/escritura en la base de datos

db.command("createUser", "nuevo_usuario", pwd="contraseña", roles=["readWrite"])

Perspectiva Dinámica

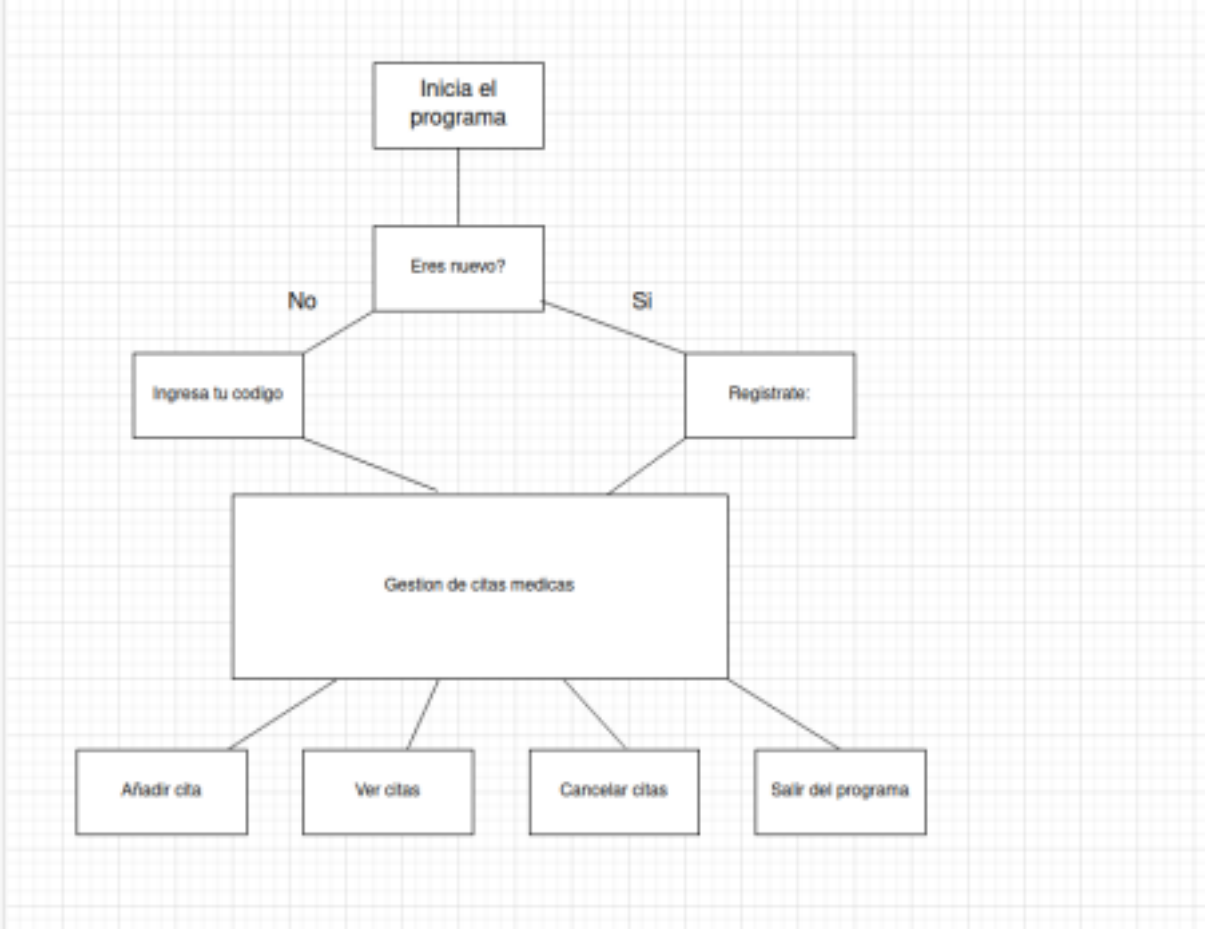
Sketch



La aplicacion basicamente cuando entras pregunta si es tu primera vez luego dependiendo de tu repuesta te pedira tu codigo o tus datos para registrarte luego

entraras al menu de gestion de citas con tu codigo de paciente y luego puedes elegir si quieres salir de la aplicación

Bosquejos o diagramas que representan la funcionalidad del proyecto.



Casos de Uso (Métodos)

Descripción de los casos de uso del proyecto y cómo se implementan.

Agregar paciente: Este caso de uso permite a los usuarios agregar un nuevo paciente al sistema. Se implementa mediante la función agregar_paciente(nombre, edad).

```
def agregar_paciente(nombre, edad):
    codigo = generar_codigo()
    db.pacientes.insert_one({"codigo": codigo, "nombre": nombre, "edad": edad})
    messagebox.showinfo("Éxito", "Paciente agregado con éxito. Su código personal es: " + codigo)
    return codigo  # Devolver el código generado
```

Agregar cita: Permite a los usuarios agregar una nueva cita para un paciente existente. Se implementa con la función agregar_cita(código, fecha, hora, doctor).

```
def agregar_cita(codigo, fecha, hora, doctor):
    paciente_id = verificar_paciente(codigo)
    if paciente_id:
        db.citas.insert_one({"paciente_id": paciente_id, "fecha": fecha, "hora": hora, "doctor": doctor})
        messagebox.showinfo("Éxito", "Cita agregada con éxito para el paciente con código: " + codigo)
    else:
        messagebox.showerror("Error", "El paciente con el código proporcionado no existe.")
```

Ver citas: Permite a los usuarios ver todas las citas de un paciente dado. Se implementa mediante la función ver citas(código).

```
def ver_citas(codigo):
    paciente_id = verificar_paciente(codigo)
    if paciente_id:
        citas = db.citas.find({"paciente_id": paciente_id})
        citas_list = list(citas)
        if citas_list:
            messagebox.showinfo("Citas del paciente", "\n".join([f"Fecha: {cita['fecha']}, Hora: {cita['hora']}
        else:
            messagebox.showinfo("Citas del paciente", "El paciente no tiene citas registradas.")
    else:
        messagebox.showerror("Error", "El paciente con el código proporcionado no existe.")
```

Cancelar cita: Permite a los usuarios cancelar una cita para un paciente dado. Se implementa con la función cancelar_cita(código, fecha, hora).

```
def cancelar_cita(codigo, fecha, hora):
    paciente_id = verificar_paciente(codigo)
    if paciente_id:
        db.citas.delete_one({"paciente_id": paciente_id, "fecha": fecha, "hora": hora})
        messagebox.showinfo("Éxito", "Cita cancelada con éxito para el paciente con código: " + codigo)
        # Actualizar la vista de citas después de la cancelación
        ver_citas(codigo)
    else:
        messagebox.showerror("Error", "El paciente con el código proporcionado no existe.")
```

Conclusiones

Resumen de los resultados obtenidos:

Realmente estoy satisfecho ya que tenía bajas expectativas acerca de lo que podía lograr pero quitando la conexión entre la base de datos y el código no ha sido muy complicado, creo que es una aplicación interesante y bastante completa.

Reflexiones sobre el proceso y posibles mejoras futuras:

El proceso ha sido bastante agobiante porque han habido unos cuantos errores difíciles de arreglar pero al final salió bien.

Como futuras mejoras implementaria las fechas actuales y que no pueden haber 2 fechas y mismas horas.

REPOSITORIO GITHUB:

[yes1221/BaseDatos_proyecto (github.com)]