TEC | Tecnológico de Costa Rica

Bases de Datos – (CE-3103)

Grupo 01

Proyecto II

Profesor:

Marco Rivera Meneses

Estudiantes:

Daniel Duarte Cordero

Manuel Emilio Alfaro Mayorga

Luis Felipe Jiménez Ulate

Mauro Andrés Navarro Obando

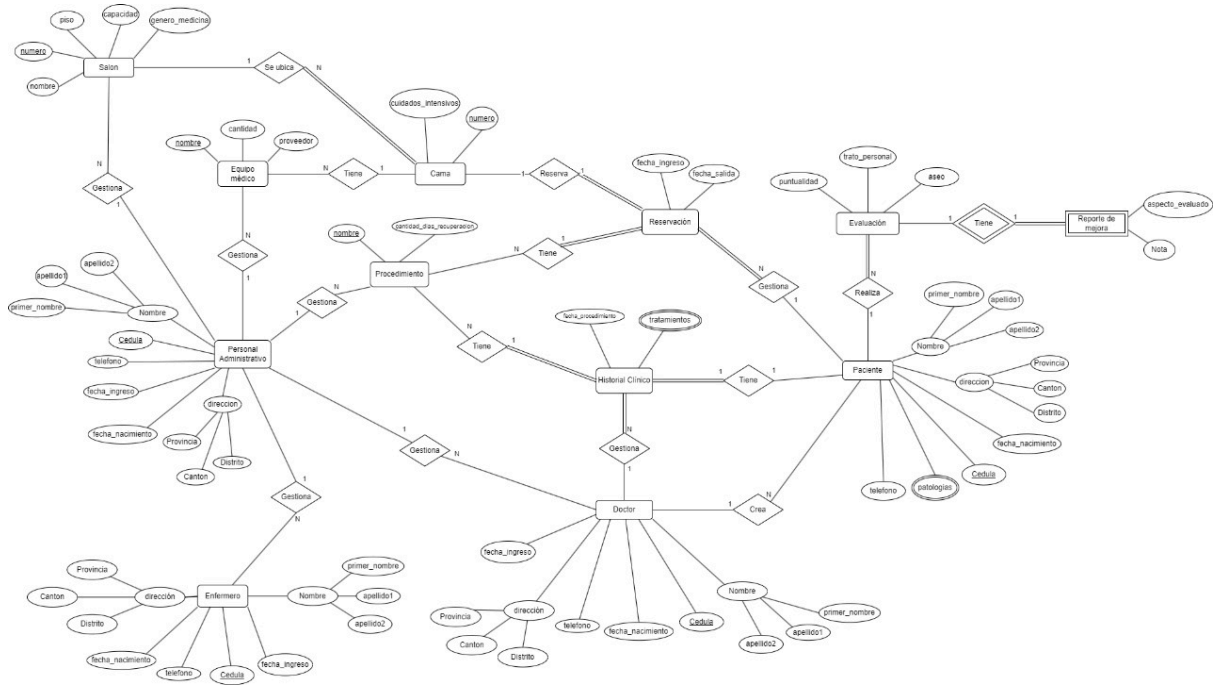
Mauricio Luna Acuña

05 / 06 / 2024

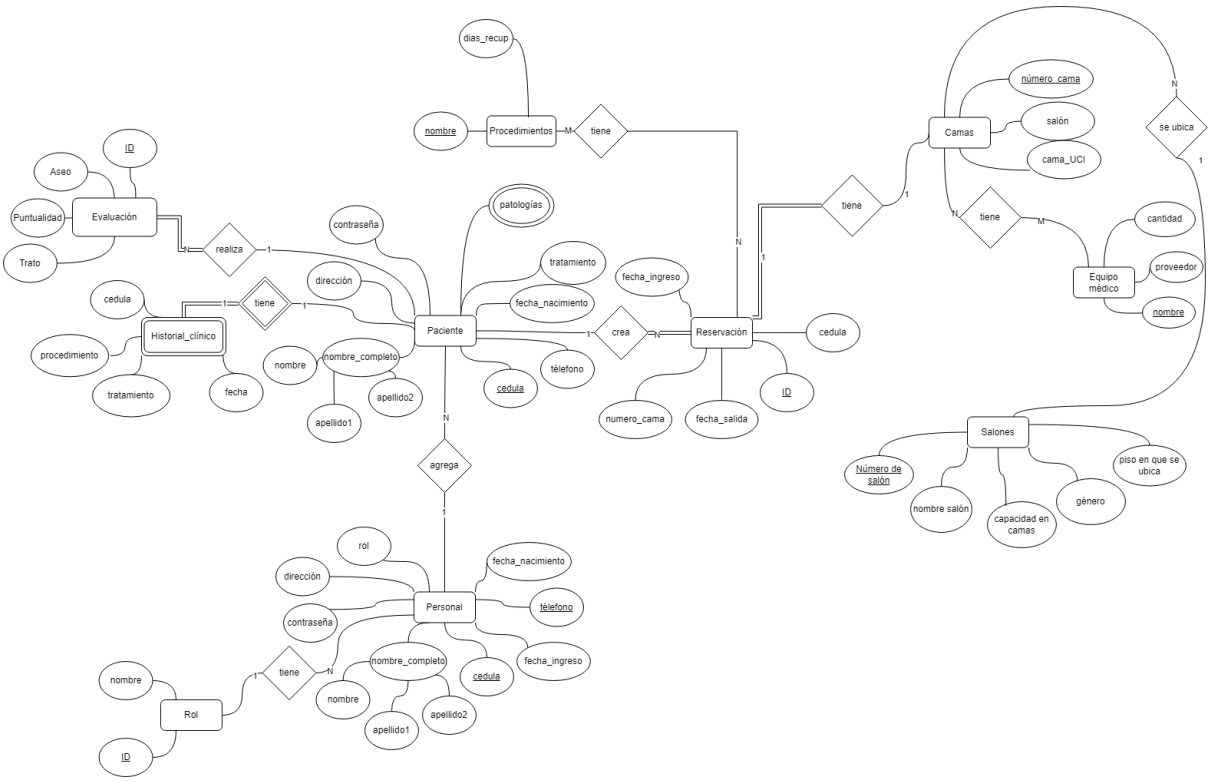
I Semestre

Índice

Modelos conceptuales propuestos con Notacion Chen



Modelo 1



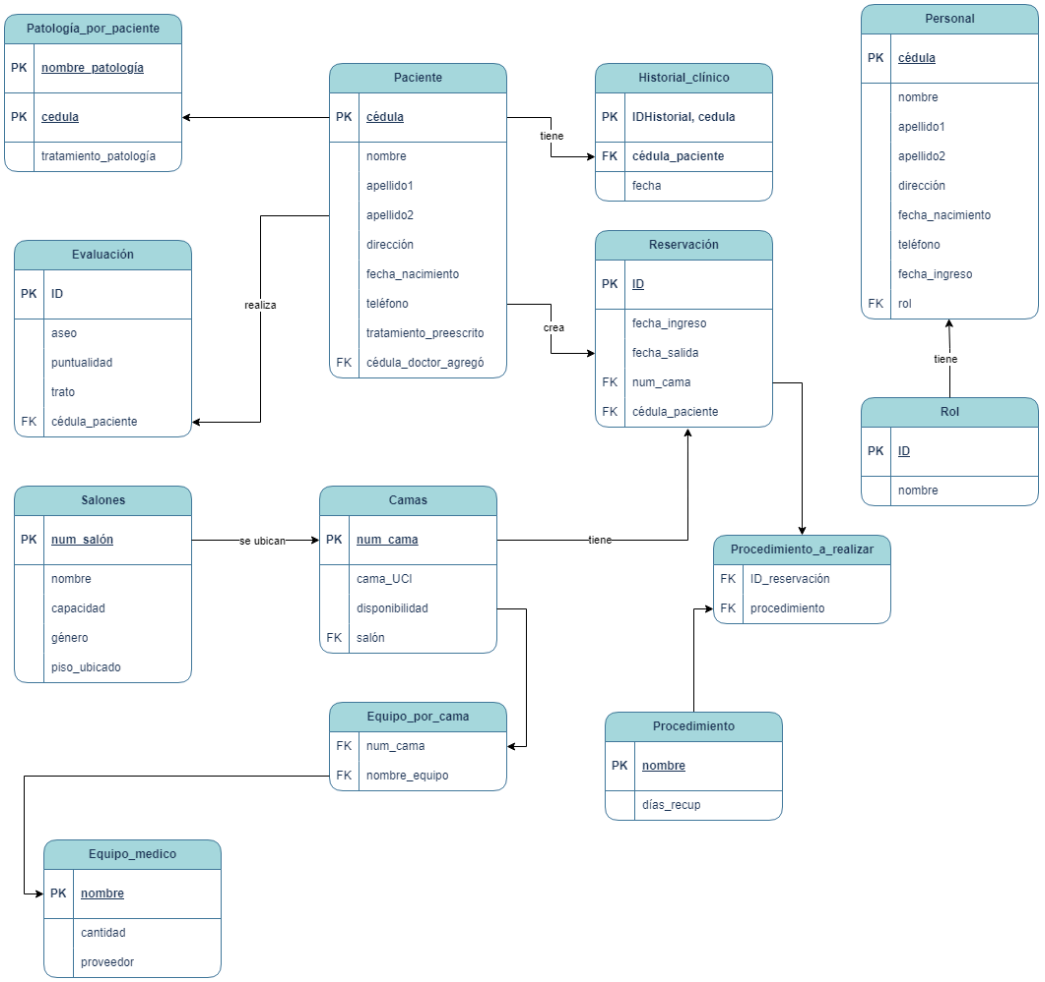
Modelo 2

Se decidió tomar el modelo conceptual 2 para seguir con la realización del modelo relacional ya que el primero contaba con relaciones innecesarias y le hacían falta algunas, por esta razón se pensó que el segundo tenía lo necesario para poder implementarse en la base de datos y que tuviera un funcionamiento correcto.

Para una mayor calidad se puede acceder a:
https://drive.google.com/file/d/1JgiDeCcrFV3bln0bVu2YmnZT3UMnufYg/view?usp=sharing

Modelo Relacional

El mapeo se encuentra en la siguiente dirección:
https://app.diagrams.net/#G1JgiDeCcrFV3bln0bVu2YmnZT3UMnufYg#%7B%22pageId%22%3A%22RWcDjfPl-x5b0UMvAG6r%22%7D



Para una mayor calidad se puede acceder a:
https://app.diagrams.net/#G1JgiDeCcrFV3bln0bVu2YmnZT3UMnufYg#%7B%22pageId%22%3A%22NE_RFxRn3LwBltYWwYsN%22%7D

Descripción de las estructuras de datos desarrolladas (Tablas).

En esta implementación, se utilizó PostgreSQL como sistema de gestión de bases de datos para gestionar la información de un hospital. A continuación se describen las tablas principales y sus relaciones.

Tabla Camas

La tabla Camas almacena información sobre las camas disponibles en el hospital. Cada cama tiene un número único que la identifica (numeroCama), un indicador de si es una cama de cuidados intensivos (camaUCI), y el número del salón en el que se encuentra (salon). El campo salon es una clave foránea que referencia a la tabla Salones.

Tabla EquipoMedico

La tabla EquipoMedico contiene información sobre los equipos médicos disponibles. Cada equipo tiene un nombre (nombre), que actúa como clave primaria, un proveedor (proveedor), y una cantidad disponible (cantidad).

Tabla EquipoPorCama

La tabla EquipoPorCama relaciona las camas con los equipos médicos asignados a ellas. Esta tabla tiene dos campos principales: numCama y equipo, que son claves foráneas que referencian a las tablas Camas y EquipoMedico, respectivamente. La combinación de estos dos campos forma la clave primaria de la tabla.

Tabla HistorialClinico

La tabla HistorialClinico almacena el historial clínico de los pacientes, incluyendo tratamientos y procedimientos realizados. Los campos principales son cedulaPaciente, fechaProcedimiento, tratamiento, y procedimiento. cedulaPaciente y procedimiento forman la clave primaria de la tabla y son claves foráneas que referencian a las tablas Paciente y Procedimientos, respectivamente.

Tabla Paciente

La tabla Paciente contiene información personal y médica de los pacientes. Cada paciente está identificado por su número de cédula (cedula), que es la clave primaria. Otros campos incluyen telefono, nombre, apellido1, apellido2, direccion, patologias, tratPatologia, y fechaNacimiento.

Tabla Personal

La tabla Personal almacena información sobre el personal del hospital. Similar a la tabla de pacientes, cada miembro del personal tiene un número de cédula (cedula) como clave primaria, y otros datos personales como telefono, nombre, apellido1, apellido2, direccion, rol, fechaNacimiento, y fechaIngreso. El campo rol es una clave foránea que referencia a la tabla Rol.

Tabla Procedimientos

La tabla Procedimientos contiene información sobre los procedimientos médicos realizados en el hospital. Cada procedimiento tiene un nombre (nombre), que es la clave primaria, y un número de días de recuperación (diasRecuperacion).

Tabla Reservacion

La tabla Reservacion gestiona las reservaciones de camas para los pacientes. Cada reservación tiene un identificador único (IDReservacion), y los campos cedulaPaciente, numCama, fechaIngreso, nombrePaciente, y procedimiento. Los atributos cedulaPaciente, numCama, y procedimiento son claves foráneas que referencian a las tablas Paciente, Camas, y Procedimientos, respectivamente.

Tabla ProcedimientoPorReservacion

Esta tabla relaciona las reservaciones con los procedimientos realizados. Tiene dos campos principales: nombreProcedimiento y IDReservacion, que son claves foráneas que referencian a las tablas Procedimientos y Reservacion, respectivamente. La combinación de estos campos forma la clave primaria de la tabla.

Tabla Rol

La tabla Rol define los diferentes roles del personal del hospital. Cada rol tiene un identificador (ID) como clave primaria y un nombre (nombre).

Tabla Salones

La tabla Salones almacena información sobre los salones del hospital. Cada salón tiene un número único (numSalon) como clave primaria, un nombre (nombreSalon), una capacidad (capacidad), el piso en el que se encuentra (piso), y el género (genero), que puede indicar si es un salón masculino, femenino, o mixto.

Relaciones entre las Tablas

La tabla Camas referencia a Salones mediante la clave foránea salon.

La tabla EquipoPorCama referencia a Camas mediante la clave foránea numCama y a EquipoMedico mediante la clave foránea equipo.

La tabla HistorialClinico referencia a Paciente mediante la clave foránea cedulaPaciente y a Procedimientos mediante la clave foránea procedimiento.

La tabla Personal referencia a Rol mediante la clave foránea rol.

La tabla Reservacion referencia a Paciente mediante la clave foránea cedulaPaciente, a Camas mediante la clave foránea numCama, y a Procedimientos mediante la clave foránea procedimiento.

La tabla ProcedimientoPorReservacion referencia a Procedimientos mediante la clave foránea nombreProcedimiento y a Reservacion mediante la clave foránea IDReservacion.

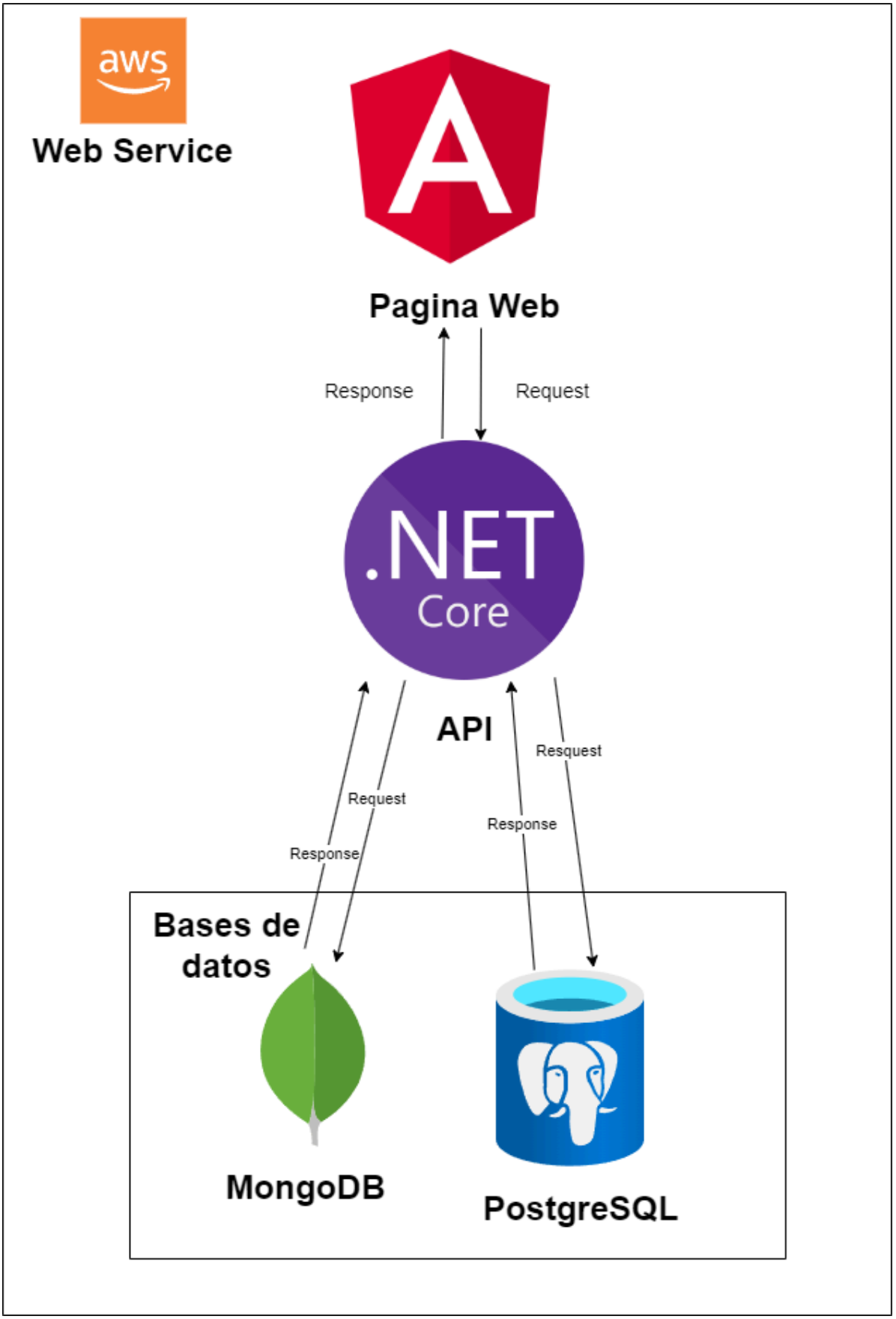
Descripción detallada de la arquitectura desarrollada

El proyecto en cuestión se desarrolló utilizando una arquitectura de capas, un enfoque que se eligió específicamente debido a la necesidad de un modelo de negocio robusto y bien definido. En esta arquitectura, la capa de presentación se encuentra en el cliente de aplicación web. Este cliente interactúa con la API, que es el punto de entrada a la lógica de negocio del sistema.

La API, a su vez, accede a la capa de lógica de negocio, también conocida como Business Logic Layer (BLL). En esta capa se implementa la inyección de dependencias, una técnica que permite que el código sea más flexible y fácil de mantener. La inyección de dependencias es una práctica de programación que ayuda a evitar las dependencias rígidas y difíciles de cambiar entre los componentes del software, lo que facilita las pruebas y el mantenimiento del código.

Desde la capa de lógica de negocio, se accede a la capa de acceso a datos, o Data Access Layer (DAL). Esta capa es la que se encarga de interactuar con la base de datos, realizando consultas a través de funciones y procedimientos almacenados, también conocidos como stored procedures. Los procedimientos almacenados son un conjunto de instrucciones postgreSQL que se almacenan en la base de datos y se pueden llamar desde la aplicación, lo que permite realizar operaciones complejas en la base de datos de manera eficiente y segura. De esta misma capa también se accede a la base de datos MongoDB para realizar escrituras como consultas.

Diagrama de arquitectura



Descripción de los store procedures y los triggers

Añadir Historial Clínico

Este stored procedure permite agregar un nuevo registro al historial clínico de un paciente.
Parámetros de entrada:

- **p_cedulaPaciente:** Cédula del paciente asociado al historial clínico.
- **p_fechaProcedimiento:** Fecha del procedimiento.
- **p_tratamiento:** Arreglo de texto que contiene el tratamiento asociado al procedimiento.
- **p_procedimiento:** Arreglo de texto que contiene el nombre del procedimiento realizado.

Crear Cama

Este stored procedure permite crear una nueva cama en un salón específico, con la opción de asignar equipos médicos.
Parámetros de entrada:

- **p_numeroCama:** Número de la cama.
- **p_equiposMedicos:** Arreglo de texto que contiene los nombres de los equipos médicos asignados a la cama.
- **p_salon:** Número del salón al que pertenece la cama.
- **p_camaUCI:** Indica si la cama es de cuidados intensivos (true/false).
- **p_disponible:** Indica si la cama está disponible para su uso (true/false).

Crear Equipo Médico

Este stored procedure permite crear un nuevo equipo médico en el sistema.
Parámetros de entrada:

- **p_nombre:** Nombre del equipo médico.
- **p_proveedor:** Proveedor del equipo médico.
- **p_cantidad:** Cantidad disponible del equipo médico.

Crear Paciente

Este stored procedure permite registrar un nuevo paciente en el sistema.
Parámetros de entrada:

- **p_cedula:** Cédula del paciente.
- **p_telefono:** Número de teléfono del paciente.
- **p_nombre:** Nombre del paciente.
- **p_apellido1:** Primer apellido del paciente.
- **p_apellido2:** Segundo apellido del paciente.
- **p_direccion:** Dirección del paciente.
- **p_patologias:** Arreglo de texto que contiene las patologías del paciente.
- **p_tratPatologia:** Arreglo de texto que contiene los tratamientos asociados a las patologías del paciente.
- **p_fechaNacimiento:** Fecha de nacimiento del paciente.
- **p_contraseña:** Contraseña para acceder al registro del paciente.

Crear Personal

Este stored procedure permite agregar un nuevo miembro al personal del centro médico.
Parámetros de entrada:

- **p_cedula:** Cédula del miembro del personal.
- **p_telefono:** Número de teléfono del miembro del personal.
- **p_nombre:** Nombre del miembro del personal.
- **p_apellido1:** Primer apellido del miembro del personal.
- **p_apellido2:** Segundo apellido del miembro del personal.
- **p_direccion:** Dirección del miembro del personal.
- **p_rol:** Rol del miembro del personal.
- **p_fechaNacimiento:** Fecha de nacimiento del miembro del personal.
- **p_fechaIngreso:** Fecha de ingreso del miembro del personal.
- **p_contraseña:** Contraseña para acceder al registro del miembro del personal.

Crear Procedimiento

Este stored procedure permite registrar un nuevo procedimiento médico en el sistema.
Parámetros de entrada:

- **p_nombre:** Nombre del procedimiento médico.
- **p_diasRecuperacion:** Número de días estimados de recuperación para el procedimiento.

Crear Reservación

Este stored procedure permite crear una nueva reservación para un paciente, asignando una cama y procedimientos.
Parámetros de entrada:

- **p_cedulaPaciente:** Cédula del paciente para la reservación.
- **p_numCama:** Número de la cama asignada.
- **p_fechaIngreso:** Fecha de ingreso del paciente.
- **p_nombrePaciente:** Nombre del paciente.
- **p_procedimientos:** Arreglo de texto que contiene los nombres de los procedimientos asociados a la reservación.

Crear Salón

Este stored procedure permite crear un nuevo salón en el centro médico.
Parámetros de entrada:

- **p_numSalon:** Número del salón.
- **p_nombreSalon:** Nombre del salón.
- **p_capacidad:** Capacidad máxima del salón.
- **p_piso:** Piso en el que se encuentra el salón.
- **p_genero:** Género para el cual está designado el salón (masculino, femenino, mixto, etc.).

Modificar Reservación

Este stored procedure permite modificar los detalles de una reservación existente, incluyendo la cédula del paciente, el número de cama, la fecha de ingreso, el nombre del paciente y los procedimientos asociados.
Argumentos de entrada:

- **p_IDReservacion (INTEGER)**: El ID de la reservación que se desea modificar.
- **p_cedulaPaciente (INTEGER):** La cédula del paciente asociado a la reservación.
- **p_numCama (INTEGER):** El número de la cama asociada a la reservación.
- **p_fechaIngreso (DATE):** La fecha de ingreso para la reservación.
- **p_nombrePaciente (TEXT):** El nombre del paciente asociado a la reservación.
- **p_procedimientos (TEXT[]):** Una lista de los procedimientos asociados a la reservación.

Modificar Cama

Este stored procedure permite modificar los detalles de una cama existente, incluyendo los equipos médicos asociados, el salón al que pertenece y si es una cama de cuidados intensivos.
Argumentos de entrada:

- **p_numeroCama (INTEGER):** El número de la cama que se desea modificar.
- **p_equiposMedicos (TEXT[]):** Una lista de los equipos médicos asociados a la cama.
- **p_salon (INTEGER):** El número del salón al que pertenece la cama.
- **p_camaUCI (BOOLEAN):** Un indicador que especifica si la cama es de cuidados intensivos.

Modificar Equipo Médico

Este stored procedure permite modificar los detalles de un equipo médico existente, incluyendo el proveedor y la cantidad disponible.
Argumentos de entrada:

- **p_nombre (TEXT):** El nombre del equipo médico que se desea modificar.
- **p_proveedor (TEXT):** El proveedor del equipo médico.
- **p_cantidad (INTEGER):** La cantidad disponible del equipo médico.

Modificar Personal

Este stored procedure permite modificar los detalles de un miembro del personal existente, incluyendo el número de teléfono, nombre, apellidos, dirección, rol, fecha de nacimiento, fecha de ingreso y contraseña.
Argumentos de entrada:

- **p_cedula (INTEGER):** La cédula del miembro del personal que se desea modificar.
- **p_telefono (TEXT):** El número de teléfono del miembro del personal.
- **p_nombre (TEXT):** El nombre del miembro del personal.
- **p_apellido1 (TEXT):** El primer apellido del miembro del personal.
- **p_apellido2 (TEXT):** El segundo apellido del miembro del personal.
- **p_direccion (TEXT):** La dirección del miembro del personal.
- **p_rol (INTEGER):** El rol del miembro del personal.
- **p_fechaNacimiento (DATE):** La fecha de nacimiento del miembro del personal.
- **p_fechaIngreso (DATE):** La fecha de ingreso del miembro del personal.
- **p_contraseña (TEXT):** La contraseña del miembro del personal.

Modificar Procedimiento

Este stored procedure permite modificar los detalles de un procedimiento médico existente, incluyendo el nombre y el número de días estimados de recuperación.
Argumentos de entrada:

- **p_nombre (TEXT):** El nombre del procedimiento médico que se desea modificar.
- **p_diasRecuperacion (INTEGER):** El número de días estimados de recuperación del procedimiento médico.

Modificar Salón

Este stored procedure permite modificar los detalles de un salón existente, incluyendo el nombre, la capacidad, el piso y el género para el cual está designado.
Argumentos de entrada:

- **p_numSalon (INTEGER):** El número del salón que se desea modificar.
- **p_nombreSalon (TEXT):** El nombre del salón.
- **p_capacidad (INTEGER):** La capacidad máxima de personas del salón.
- **p_piso (TEXT):** El piso en el que se encuentra el salón.

- **p_genero (TEXT):** El género para el cual está designado el salón.

Problemas Conocidos

No tenemos problemas conocidos.

Problemas Encontrados

A la hora de integrar bootstrap con Angular se presentaron bastantes inconvenientes, precisamente con los componentes que poseían algún tipo de animación. Se realizó la instalación mediante la terminal y luego se modificó el Angular.json para añadir los archivos necesarios de Bootstrap en la sección de styles y scripts. Sin embargo, se seguían presentando limitaciones para ciertos componentes. Se decidió acudir a ngx-boostrap, ya que es una biblioteca de Bootstrap que se usa específicamente para angular. Sin embargo, el problema de que ciertos componentes no se comportaban de la manera deseada persistía.

Por lo que finalmente se hizo uso de la librería ng-bootstrap. Esta también corresponde a una librería diseñada específicamente para Angular. Esta, si bien no cuenta con un gran repositorio de componentes a utilizar está bastante completo, además de que en el proyecto se complementa estos complementos con los de Bootstrap que sí se lograban integrar.

Un problema que nos ocurría mucho a la hora de hacer queries en la base de datos era que en PL/pgSQL, a diferencia de SQL, los identificadores deben ponerse entre comillas dobles para mantener su capitalización original. Mientras que en SQL los identificadores no son sensibles a mayúsculas y minúsculas y se convierten automáticamente a minúsculas si no se encierran entre comillas, en PL/pgSQL es crucial usar comillas dobles para evitar errores. Esto garantiza que los nombres de tablas y columnas con capitalización específica se reconozcan correctamente en las consultas y operaciones dentro de las funciones y procedimientos almacenados.

Conclusiones

Ya que este proyecto fue similar al proyecto 1 del curso de bases de datos, se notó un mayor dominio y conocimiento general, no solo en las secciones asignadas a cada integrante sino también sobre el funcionamiento general de una página web y la importancia de la correcta estructuración de las bases de datos.

Trabajar con tecnologías en la nube requieren una mayor organización al inicio del proyecto pero brindan mucha facilidad para trabajar en etapas medias y tardías del proyecto. Realmente el flujo de trabajo es más eficiente y los errores de una parte de la aplicación pueden ser fácilmente solventados por el equipo a cargo.

Es importante identificar los integrantes del equipo que tiene ciertas fortalezas, de esta manera pueden guiar a los otros integrantes y el avance es continuo.

El proyecto 2 se trabajó bien, el equipo se complementó muy bien durante todo el transcurso del mismo. Se puede notar que cuando existe comunicación continua de parte de todos los miembros del equipo, realmente se puede progresar y concretar un proyecto de calidad.

Recomendaciones

Una de las mayores recomendaciones es la de trabajar con tecnologías en la nube, realmente facilita mucho la implementación del proyecto y permite que cada equipo pueda dar feedback de las otras partes del proyecto desde etapas tempranas y sin las limitantes de hardware que alguno de los integrantes del equipo podría llegar a tener.

Por parte de los desarrolladores de la página web, se puede realizar una comparativa entre los frameworks de Angular y React. Angular a pesar de ser más organizado requiere mucho mayo entendimiento de ciertas estructuras y funciones básicas de html, css y javascript, además de los conceptos de Angular como tal. Por lo que se recomienda utilizar React o al menos darle una oportunidad ya que puede facilitar la comprensión de algunas características que también se utilizan en Angular.

Bibliografía

ProgrammingKnowledge. (2021, June 9). *How to install MongoDB 6 on Windows 10/ Windows 11*. [Video]. YouTube. https://www.youtube.com/watch?v=gB6WLkSrtJk&ab_channel=ProgrammingKnowledge

Geeky Script. (2021, August 19). How to Install Angular 15 on Windows 10/11 [2023 Update] Demo Angular Project | Complete Guide. [Video]. YouTube. https://www.youtube.com/watch?v=-cdk3hJP8uQ&ab_channel=GeekyScript

Geeky Script. (2021, September 7). How to Install PostgreSQL on Windows 10/11 [2024 Update] pgAdmin 4. [Video]. YouTube. https://www.youtube.com/watch?v=uN0AfifH1TA&ab_channel=GeekyScript

Angular. (n.d.). Documentación oficial de Angular. https://docs.angular.lat/docs

Bitácoras

Mauricio Luna Acuña

Fecha	Actividad Realizada
25/05/2024	*Se crea la plantilla de la API Web y se crean las clases algunas entidades entidades*
27/05/2024	*Se crea la base de la capa de acceso y la capa de negocio, además, se crea un controlador para probar conexión con la base postgreSQL*
30/05/2024	*Se termina todas las clases de las entidades correspondientes y se crea algunos endpoints para cada una de estas*
03/06/2024	*Se termina de hacer todos los endpoints necesarios, se corrige algunos bugs de la capa de acceso*
04/06/2024	*Se crea una conexión de la API con la base de datos MongoDB así como sus respectivos endpoints para consultas o escritura de datos*
05/06/2024	*Se corrijen bugs encontrados*

Daniel Duarte Cordero

Fecha	Actividad Realizada
29/05/2024	*Se creó el proyecto de angular, se creó la página home y se investigó cómo integrar bootstrap.*

1/06/2024	*Se intentó realizar la integración con distintas variantes de Bootstrap y se realizaron pruebas para verificar su funcionamiento. Se finalizó la parte de estilos de la página home.*
2/06/2024	*Se comenzó a implementar la vista paciente.* *Se comenzó precisamente con el manejo de las reservaciones de la camilla.*
3/06/2024	*Se continuó con la implementación de la vista paciente, se priorizó la modularidad en el código y se finalizó el manejo de reservaciones.*
4/06/2024	*Se comenzaron a implementar distintos endpoints. Se finalizó con la sección de paciente y se comenzó con la sección de doctor.*
5/06/2024	*Se finalizó la sección de profesor. Y se corrigieron ciertos bugs. Finalmente, se inició la documentación externa.*

Luis Felipe Jiménez Ulate

Fecha	*Actividad Realizada*
18/05/2024	*Se realiza el modelo conceptual con Daniel y Mauro.*
21/05/2024	*Se empieza a descargar las aplicaciones necesarias para poder crear la base de datos en Postgre*
24/05/2024	*Se empieza con la creación de la base de datos en PgAdmin4*
28/05/2024	*Tuve que eliminar la base de datos y volverla a crear en el servicio de AWS para así ya tenerla subida en la*

	nube
2/06/2024	*Se terminan todos los store procedures que se creen necesarios, en caso de que se ocupen más se van a realizar ajustes*
3/06/2024	*Se terminan las funciones y los triggers junto a Emilio para que así la API pueda utilizarlos*
4/06/2024	*Me dediqué a realizar cambios necesarios que pedían Daniel y Mauro en la Web*
5/06/2024	*Hoy me dediqué a terminar todos los documentos de documentación que faltaban*

Manuel Emilio Alfaro Mayorga

Fecha	*Actividad Realizada*
19/05/2024	*Me reuní con Mauricio para hacer nuestro modelo conceptual y nos salió bien, después realizar el mapeo del modelo relacional con una combinación de los dos modelos conceptuales*
21/05/2024	*Hoy comencé con el mapeo del modelo relacional. En si solo hice los pasos de mapeo ya solo falta unirlo todo para que quede el modelo relacional*
23/05/2024	*Hoy ya se completo el modelo relacional, ahora se lo mandé a Felipe para que lo revisara y que pudiéramos empezar con la creación de las tablas de la base de datos.*
27/05/2024	*Hoy me reuní con Felipe para comenzar con la creación de las tablas de la base de datos, si se*

	pudieron crear todas las tablas entonces todo salió bien.
01/06/2024	*Hoy me puse a crear el script de población de la base de datos y se pudo hacer fácil y sencillo.*
02/06/2024	*Hoy me puse a crear la base en MongoDB y estaba muy fácil, le dije a Mauro que después nos conectaremos para hacer la conexión entre mongo y la API. También me puse a avanzar un poco la documentación del proyecto, específicamente poniendo los modelos conceptuales, poniendo el modelo relacional en la docu, haciendo el diagrama de arquitectura y por último dando estructura al documento.*
03/06/2024	*Hoy nos conectamos todos en discord para que Daniel y Mauro nos dijeran que store procedures y funciones ocupaban de la base de datos, al final del día pudimos darle casi que todos los procedures que nos pedían.*
04/06/2024	*Hoy también nos conectamos todos en discord y pudimos terminar todos los store procedures y funciones que nos pedían, además hicimos los dos triggers que faltaban en la base.*
05/06/2024	*Hoy también nos conectamos todos en discord y se pudo hacer la conexión entre la base en mongo y la API, también se comencé a avanzar el manual de instalación. Además tuvimos que hacer unos arreglos en las funciones y los procedures que daban unos errores.*
06/06/2024	*Hoy por último terminé el manual de*

	instalación, mi parte de la docu y finalmente mandamos el proyecto.

Mauro Andrés Navarro Obando

14/05/24	*Se hace una pequeña distribución de historias de usuario para la página web*
16/05/24	*El grupo se reunió hoy y se plantean las metas para la próxima semana*
19/05/24	*Hoy nos reunimos para completar un taller pero aprovechamos para montar el primer modelo relacional. Los dos compañeros que no asistieron a la reunión realizarán el segundo modelo para compararlo posteriormente*
22/05/24	*Repaso de conceptos básicos de Angular.* *Se crea una página básica con un hola mundo y un router para. Está página también puede servir como base para el proyecto*
24/05/24	*Prototipo básico de las páginas utilizando la herramienta de dibujo de OneNote*
25/05/24	*Se crea una base simple de una página de registro utilizando Angular y bootstrap. Un objetivo importante es que veo que se necesita tener un buen manejo de forms en Angular.*
01/06/24	*Estudio de las tecnologías de AWS. Evaluar cuales son los*

	requerimientos para hacer el deploy de AWS por medio de la página web. Se usará EC2 y Angular.
02/06/24	*Se hace el deploy de la página web usando EC2 de AWS. El compañero Daniel ya tenía una base bastante sólida en Angular. Yo proseguiré trabajando en la vista del administrador.*
03/06/24	*Página base del administrador lista. Se añaden imágenes y todos los botones necesarios.*
04/06/24	*Se actualiza la página correctamente en la instancia de AWS. Se tiene el 50% de la página del administrador. Gestión de salones y salones y equipo con endpoints.*
05/06/26	*Versión final de la página en AWS. Se completa la vista de administrador, gestión de camas, procedimientos y personal.*

Evidencias de trabajo grupal

Reunión # 1

Fecha: 15 de marzo de 2024
Asistentes: Emilio (Bases de Datos MongoDB), Felipe (Bases de Datos PostgreSQL), Mauricio (API con C# y .NET), Daniel (Página web con Angular), Mauro (Página web con Angular)
Objetivo: Planificación inicial del proyecto

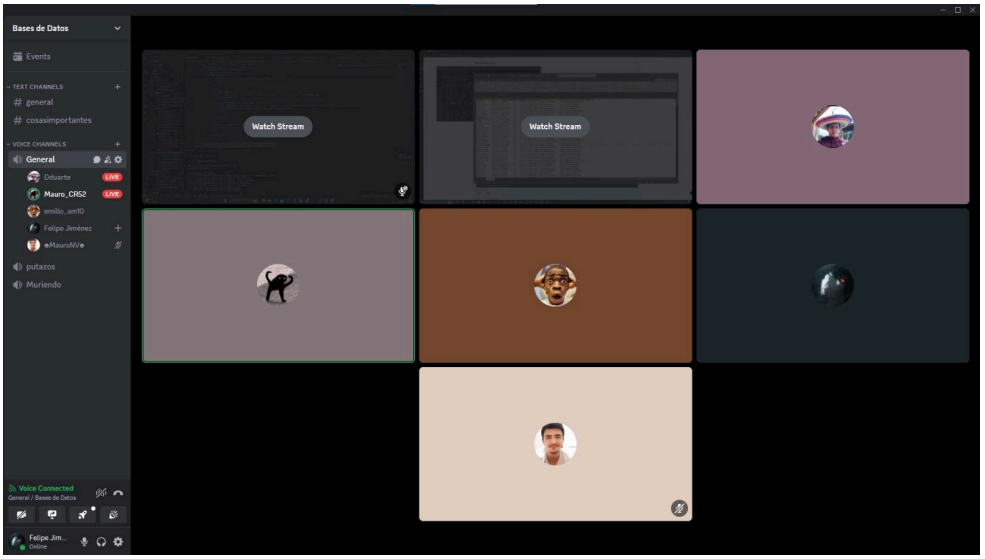
Agenda:

1. Definir los roles y responsabilidades.
2. Establecer la arquitectura del proyecto.
3. Planificar el cronograma de desarrollo.

Notas:

- **Felipe y Emilio** trabajarán en las bases de datos. Emilio se encargará principalmente de MongoDB pero dando apoyo a Felipe que se centrará en la base de datos en PostgreSQL.
- **Mauricio** será responsable del desarrollo del API utilizando C# y .NET, y el despliegue en Azure.
- **Daniel y Mauro** desarrollarán la página web utilizando Angular y la desplegarán en AWS.
- Se discutió y aprobó la arquitectura general del proyecto.
- Se estableció un cronograma preliminar, con hitos semanales y revisiones cada dos semanas.
- Hay que tener buena comunicación y centrarse en los avances semanales.

Imagen de la reunión



Reunión#2

Fecha: 22 de marzo de 2024
Asistentes: Felipe, Emilio, Mauro, Daniel, Mauricio
Objetivo: Revisión de avances iniciales

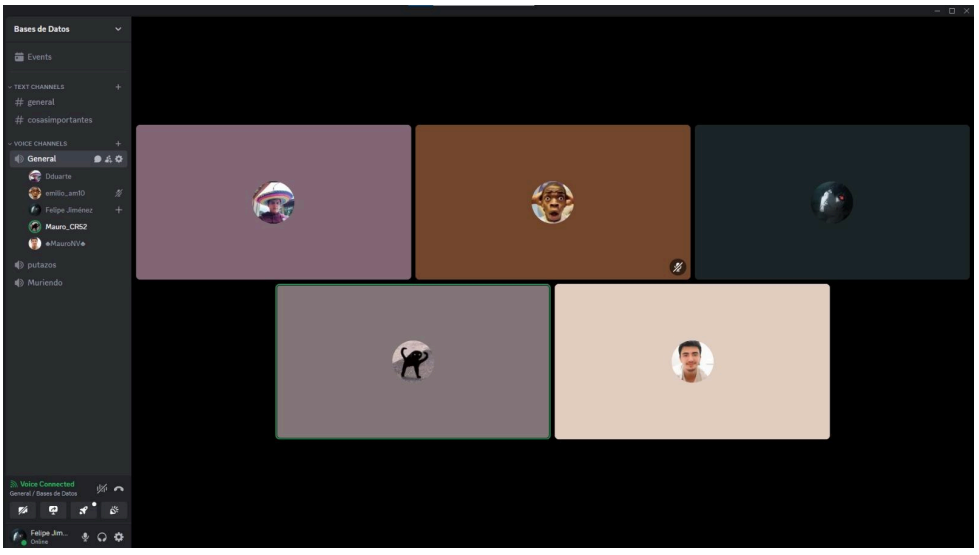
Agenda:

1. Revisar el progreso de la configuración de entornos.
2. Discutir problemas encontrados.
3. Ajustar el cronograma según sea necesario.
4. Creación de modelos conceptuales.

Notas:

- **Emilio** reportó la instalación y configuración de MongoDB en AWS. Configuración básica completada.
- **Felipe** instaló PostgreSQL en AWS y configuró los esquemas iniciales.
- **Mauricio** configuró el entorno de desarrollo de C# y .NET en Azure.
- **Mauro** comenzaron a repasar algunos conceptos de Angular y generó una estructura básica de un proyecto Angular.
- **Daniel** ha estado con exámenes por lo que no ha podido centrarse en el proyecto.
- Se identificó la necesidad de mayor integración entre los equipos de bases de datos y API.
- Se ajustó el cronograma para incluir reuniones diarias breves de sincronización.
- Dos integrantes tuvieron que retirarse temprano, por lo que solo se pudo realizar un modelo conceptua.

Imagen de la reunión



Reunión #3

Fecha: 29 de marzo de 2024
Asistentes: Felipe, Daniel, Emilio
Objetivo: Revisión del primer sprint

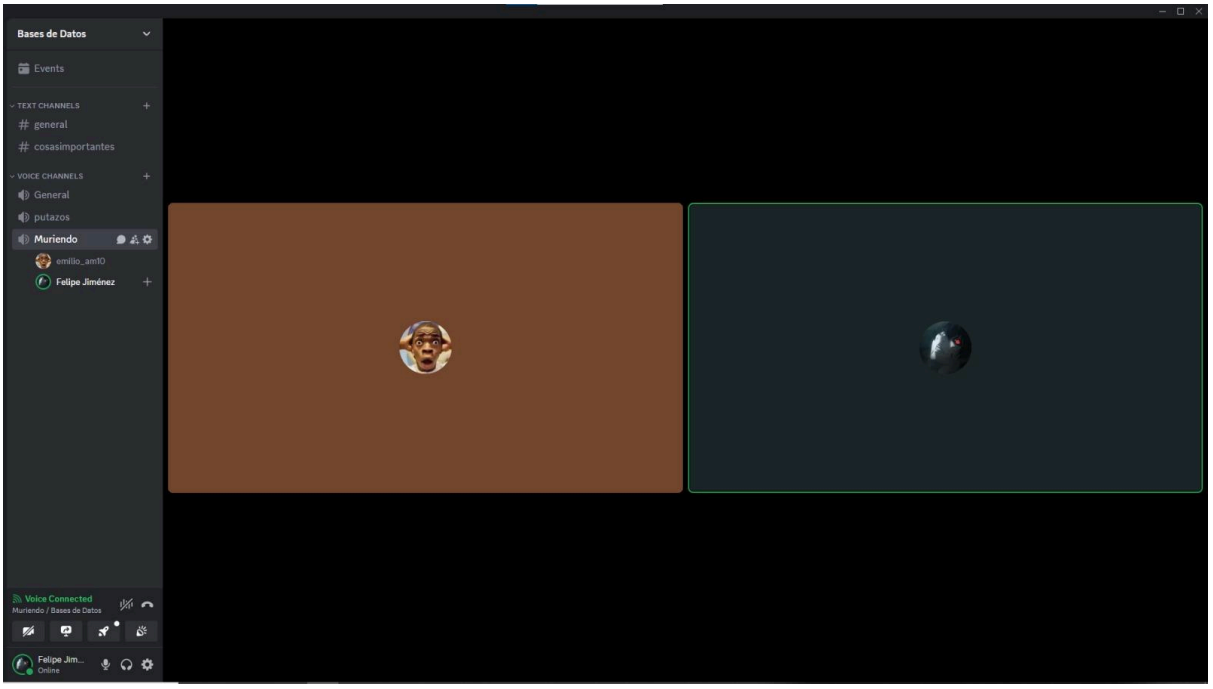
Agenda:

1. Presentación de avances por equipo.
2. Identificación de obstáculos.
3. Planificación del siguiente sprint.
4. Creación del segundo modelo conceptual.

Notas:

- **Emilio** completó la creación de colecciones y estructuras de documentos en MongoDB.
- **Felipe** terminó la estructura de tablas y relaciones en PostgreSQL
- **Juan y Laura** crearon las primeras vistas y componentes del frontend.
- **Mauricio** no logró asistir a la reunión por un tema laboral, pero reportó que avanzó en la creación de endpoints básicos para el API, sobre todo los de la vista Administrador y Paciente.
- **Mauro** no logró asistir debido a un examen, pero ha comunicado que sigue repasando Angular y se va a dedicar al deploy de la página web en AWS.
- Se discutieron algunos problemas de compatibilidad entre Mongo y Postgre y se acordaron soluciones.
- Planificación de tareas para el próximo sprint: conexiones API a bases de datos y desarrollo de funcionalidades avanzadas del frontend.

Imagen de la reunión



Reunión #4

Fecha: 01 de Junio de 2024
Asistentes: Felipe, Daniel, Emilio, Mauricio y Mauro
Objetivo: Sincronización de datos y pruebas iniciales

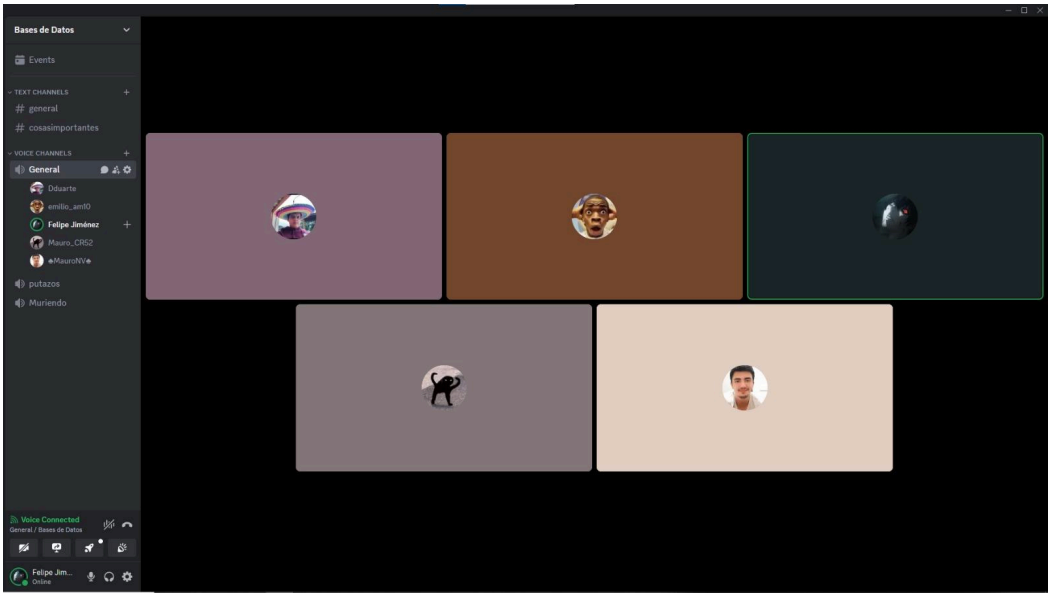
Agenda:

1. Revisión de la integración entre API y bases de datos.
2. Planificación de pruebas unitarias.
3. Discusión de la autenticación y deploy en la nube de las partes del proyecto.

Notas:

- **Emilio** y **Felipe** ya trabajaron en la sincronización de datos entre MongoDB y PostgreSQL.
- **Mauricio** integró las bases de datos con el API y comenzó a realizar pruebas sobre los endpoints que ya están activos.
- **Daniel** inició el desarrollo de la página web en Angular, ya tiene una base sólida para la página inicial, login y sign-in.
- **Mauro** ya ha completado el despliegue de la página web en AWS.
- Se identificaron problemas menores de latencia en la comunicación entre el API y las bases de datos.
- Acordaron realizar una sesión de pruebas conjunta la próxima semana.
- Se mencionó preocupación sobre los avances acerca de la documentación del proyecto, por lo que **Felipe** se comprometió a enviar lo que cada uno debe ir avanzando en la documentación.

Imagen de la reunión



Reunión #5

Fecha: 03 de junio de 2024
Asistentes: Emilio, Mauro, Daniel, Felipe, Muaricio
Objetivo: Pruebas y ajustes

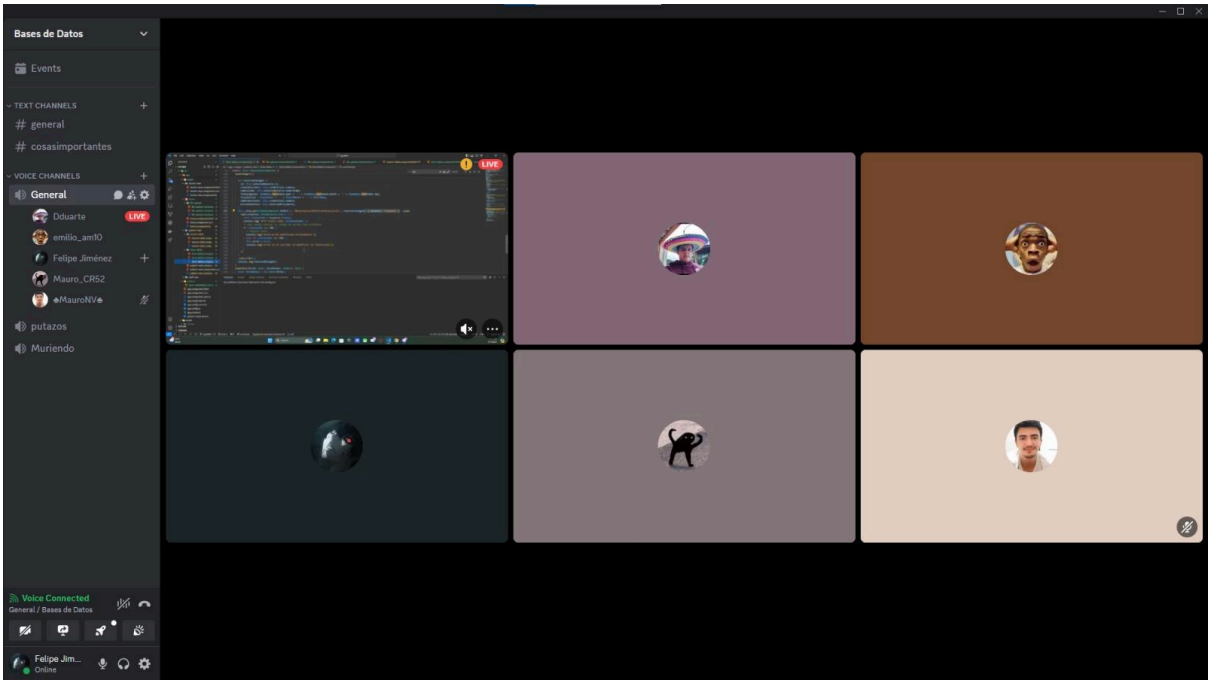
Agenda:

1. Resultados de las pruebas conjuntas.
2. Identificación y corrección de errores.
3. Revisión de la interfaz de usuario.

Notas:

- **Felipe y Emilio** solucionaron los problemas de comunicación entre las bases de Datos, se modificaron algunos stored procedures. Todos los stored procedures están listos, falta terminar de probarlos.
- **Mauricio** reportó la finalización de las pruebas de los endpoints que hay actualmente en el API.
- **Daniel** ha avanzado bastante bien en el desarrollo del frontend, pero aún faltan algunas funcionalidades de la vista doctor.
- **Mauro** ha completado la mitad de la vista del administrador.
- Se discutieron ajustes en la interfaz de usuario para mejorar la experiencia del usuario.
- Se planificó una demo interna para la siguiente reunión.

Imagen de la reunión



Reunión #6

Fecha: 05 de Junio de 2024
Asistentes: Emilio, Mauro, Daniel, Felipe, Muaricio
Objetivo: Demo interna y revisión final

Agenda:

1. Presentación de la demo interna.
2. Feedback y ajustes finales.
3. Planificación de la entrega al cliente.

Notas:

- La demo interna resaltó algunos errores existentes con algunos procedures y endpoints; la app web también tenía algunos errores de actualización.
- Se finaliza la documentación interna y técnica.
- Se realizaron algunos ajustes necesarios finales para el correcto funcionamiento del proyecto.
- Se realiza el despliegue final de la app web en AWS.
- Los miembros expresaron satisfacción con el progreso y la colaboración.

Imagen de la reunión