UNIVERSIDAD ESCUELA COLOMBIANA DE NGENIERÍA JULIO GARAVITO

DOCUMENTO DE ARQUITECTURA

ECI-Bienestar

Equipo Diamante

AUTORES

Vicente Garzón Ríos

Daniel Alejandro Diaz Camelo

Carlos David Barrero

MODULO

Gestión de Turnos para Servicios de Bienestar Universitario

PROGRAMA

Ingeniería de Sistemas

ASIGNATURA

Ciclos de Vida del Desarrollo de Software (CVDS)

PROFESORES

Ing. Rodrigo Humberto Gualtero Martínez

Ing. Andrés Martín Cantor Urrego

Descripción del Modulo

Este módulo permite a los miembros de la comunidad universitaria (estudiantes, docentes, administrativos y personal de servicios generales) gestionar y visualizar turnos para atención en los servicios de bienestar institucional: medicina general, odontología y psicología.

El sistema contempla la asignación de turnos desde tablets de autoservicio, control administrativo por parte del personal autorizado y seguimiento por parte de los profesionales de la salud.

Perfiles de usuario

Paciente:

- Estudiante.
- Docente.
- Administrativo.
- Servicios generale.

Doctor.

Administrador:

- Secretaria medica

Requisitos funcionales

1. Registro de turnos por parte de los usuarios

- El sistema debe permitir a los usuarios registrarse al llegar a las oficinas de bienestar mediante una interfaz en una tableta.
- El sistema debe permitir al usuario ingresar su nombre completo y número de documento de identidad.
- El sistema debe permitir seleccionar su rol dentro de la institución:
  Estudiante, Docente, Administrativo o Servicios Generales.
- El sistema debe permitir elegir una especialidad disponible: Medicina General, Odontología o Psicología.

- El sistema debe permitir marcar prioridad en caso de condiciones especiales como embarazo o discapacidad.
- El sistema debe generar y mostrar una confirmación visual del turno asignado, en formato "letra-número" (ej. “O-15” para Odontología).
- El sistema debe permitir deshabilitar temporalmente la asignación de nuevos turnos en caso de emergencia.

2. Gestión de disponibilidad por el administrador

- El sistema debe permitir al administrador habilitar o deshabilitar la asignación de turnos para cada especialidad según la disponibilidad del personal.
- El sistema debe permitir habilitar o deshabilitar todos los turnos globalmente en caso necesario.

3. Visualización de turnos y contenido multimedia

- El sistema debe mostrar en pantalla el turno actualmente llamado, incluyendo nombre del usuario.
- El sistema debe mostrar en la misma pantalla contenido multimedia informativo (GIF o MP4) sobre los servicios de bienestar universitario.
- El sistema debe permitir que solo personal autorizado pueda subir, modificar o eliminar el contenido multimedia.

4. Interfaz para profesionales de la salud

- El sistema debe permitir a los profesionales de salud ver la lista de turnos pendientes por atender.
- El sistema debe permitir al profesional llamar al siguiente turno disponible.
- El sistema debe permitir al profesional seleccionar un turno específico si lo considera necesario.

5. Actualización dinámica de turnos

- El sistema debe actualizar automáticamente la pantalla de visualización al momento de que un usuario sea atendido, eliminando su turno de la lista.
- El sistema debe mostrar la lista actualizada de los siguientes turnos programados.

6. Generación de informes

- El sistema debe permitir al administrador generar reportes detallados sobre los turnos atendidos.
- El sistema debe permitir filtrar los reportes por rango de fechas y por rol del usuario (Estudiante, Docente, etc.).
- El sistema debe mostrar estadísticas como el número de turnos por especialidad y el nivel promedio de atención brindado.

Tecnologías a usar

Categoría	Tecnología	Justificación
Lenguaje	Java 17	Estabilidad, soporte LTS (Long Term Support).
Framework principal	Spring Boot	Estandarizado para microservicios RESTful.
Dase de datos	PostgreSQL	Robusto, relacional y gratuito. Ideal para consistencia transaccional.
Mensajería / Bus de datos	Apache Kafka + Spring Cloud Bus	Comunicación asíncrona y eventos de emergencia / actualización de estados.
Seguridad	JWT (JSON Web Token)	Autenticación segura, stateless.
ORM / Utilidades	Lombok	Para reducir código boilerplate (constructores, getters/setters
		automáticos).
Gestor de proyecto	Maven	Construcción de proyectos Java estandarizada.

Actualización	WebSocket	Para actualizar la pantalla de turnos dinámicamente sin recargar.
Herramientas adicionales	JaCoCo, SonarQube	Control de calidad de código y cobertura de pruebas.
Documentación de API	Swagger / Springdoc OpenAPI	Generación automática y visualización interactiva de la documentación de APIs.

Comparativa de Tecnologías Alternativas Evaluadas:

Node.js vs Java

Se optó por Java en lugar de Node.js debido a varias razones clave asociadas al contexto del proyecto:

- Robustez y madurez: Java es un lenguaje consolidado con décadas de evolución y es ampliamente usado en sistemas críticos, especialmente en entornos empresariales, donde se valora la estabilidad a largo plazo.
- Soporte empresarial: El ecosistema Java (especialmente con frameworks como Spring Boot) está diseñado para aplicaciones empresariales escalables, seguras y mantenibles. Cuenta con herramientas integradas para autenticación, manejo de errores, pruebas automatizadas y más.
- Tipado estático y control estricto: A diferencia de Node.js (JavaScript/TypeScript), Java permite detectar más errores en tiempo de compilación, lo que reduce problemas en producción.
- Multihilo y concurrencia: Java ofrece un manejo avanzado de concurrencia, útil en escenarios donde múltiples procesos pueden estar interactuando con el sistema (como la atención simultánea de turnos).

Por estas razones, Java fue la opción más adecuada para una solución sólida, con requerimientos estructurados y alta expectativa de mantenibilidad.

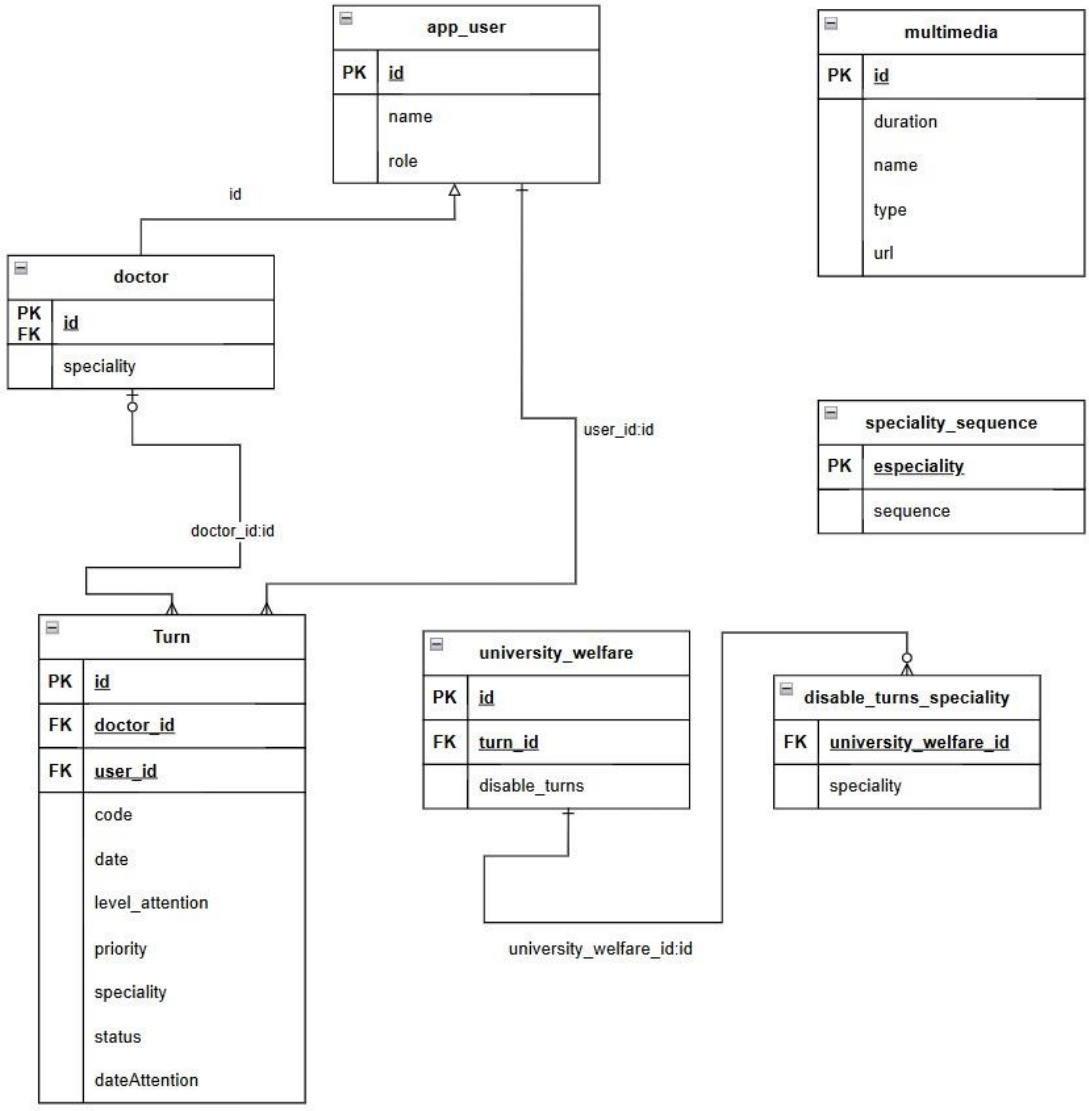
MongoDB vs PostgreSQL

Aunque el sistema actual gestiona un conjunto relativamente pequeño de entidades (usuarios, turnos y multimedia), se optó por PostgreSQL debido a que:

- Modelo relacional estructurado: A pesar del número limitado de tablas, las entidades tienen estructuras definidas y relaciones directas entre sí (por ejemplo, un turno asociado a un usuario). PostgreSQL permite modelar estas relaciones de manera clara, utilizando claves foráneas y restricciones que aseguran la consistencia de los datos.
- Integridad de datos garantizada: PostgreSQL ofrece mecanismos nativos para validar la integridad (como unicidad de usuarios, o restricciones de formato en los turnos), lo cual es fundamental en sistemas donde no puede haber ambigüedad ni duplicidad, especialmente en procesos secuenciales como la gestión de turnos.
- Escalabilidad estructurada: Aunque actualmente son pocas tablas, el modelo relacional facilita una evolución controlada del sistema en el futuro (por ejemplo, agregando especialidades, reportes o historiales de atención) sin perder coherencia.
- Consultas analíticas y filtros: PostgreSQL brinda capacidades avanzadas de consultas SQL, útiles para reportes y análisis (fechas, conteos por usuario, estados de turnos), que pueden ser más complejas de optimizar en un modelo NoSQL como MongoDB.

En resumen, se eligió PostgreSQL no por la cantidad de entidades, sino por la necesidad de mantener consistencia, validación y claridad en la lógica de datos, fundamentales incluso en sistemas con pocos modelos.

Diagrama de datos



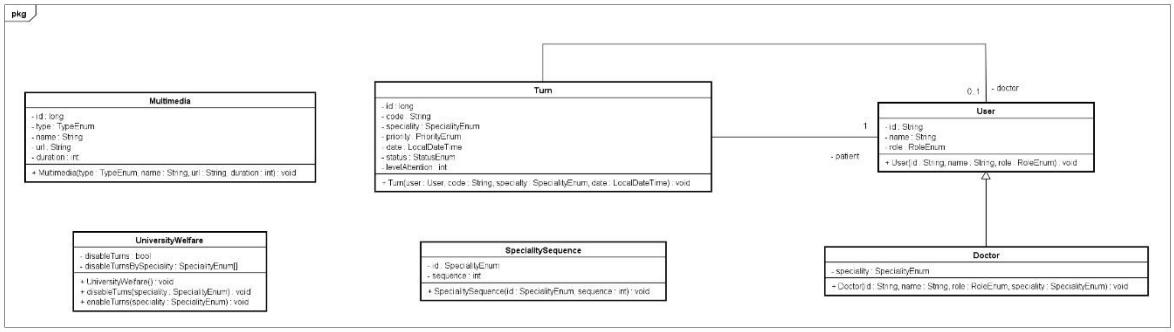
El sistema utiliza una base de datos relacional (PostgreSQL) cuyo modelo representa las entidades clave involucradas en la gestión de turnos dentro del contexto de bienestar universitario. A continuación, se describen las tablas principales y su propósito:

- app_user: Representa a los usuarios del sistema (como estudiantes o personal administrativo). Aunque el sistema general cuenta con un módulo centralizado de autenticación, este microservicio almacena localmente la tabla app_user para evitar

una dependencia directa, garantizando la autonomía y resiliencia del servicio en conformidad con los principios de diseño de microservicios.

- doctor: Representa a los profesionales encargados de atender los turnos. Está asociado directamente a un registro en app_user y contiene información adicional como la especialidad médica.
- turn: Registra los turnos solicitados por los usuarios, incluyendo atributos como código, fecha, nivel de atención, prioridad, especialidad y estado del turno. Cada turno se asocia tanto a un usuario como a un doctor.
- university_welfare: Define la configuración general del servicio de bienestar universitario, incluyendo si los turnos están habilitados o deshabilitados.
- disable_turns_speciality: Relaciona las especialidades con los servicios de bienestar que tienen turnos deshabilitados. Esta tabla permite representar múltiples especialidades deshabilitadas por instancia de bienestar, solventando la limitación de las bases de datos relacionales respecto al almacenamiento de listas.
- multimedia: Almacena contenido informativo como videos o imágenes relacionados con el servicio, incluyendo atributos como duración, nombre, tipo y URL.
- speciality_sequence: Lleva un control de numeración secuencial por especialidad, lo que permite asignar un número de turno ordenado por tipo de atención médica.

Diagrama de Clases



El diagrama de clases representa las principales entidades involucradas en la gestión de turnos en el contexto de bienestar universitario. Cada clase encapsula atributos y relaciones específicas para reflejar el comportamiento y estructura del sistema.

Clases Principales:

- User
    - Representa a los usuarios que pueden solicitar turnos.
    - Atributos: id, name, role
    - Tiene una relación con la clase Turn como paciente.
- Doctor
    - Representa a los profesionales encargados de atender turnos.
    - Atributos: userId, speciality
    - Se relaciona con un User y está asociado a una especialidad médica.
- Turn
    - Contiene la información de los turnos asignados o solicitados. ○ Atributos: id, code, date, levelAttention, priority, speciality, status
    - Asociaciones:
        - Un Doctor que atiende el turno.
        - Un User que solicita el turno.
- UniversityWelfare ○ Contiene configuraciones relacionadas con la disponibilidad de turnos.
    - Atributos: id, disableTurns, disableTurnsBySpeciality (lista de especialidades deshabilitadas)
- Multimedia
    - Gestiona contenido informativo relacionado al servicio. ○ Atributos: id, name, type, url, duration

- SpecialitySequence
    - Lleva el control de la numeración secuencial de turnos por especialidad. ○ Atributos: id, speciality, sequence

Diagrama de componentes

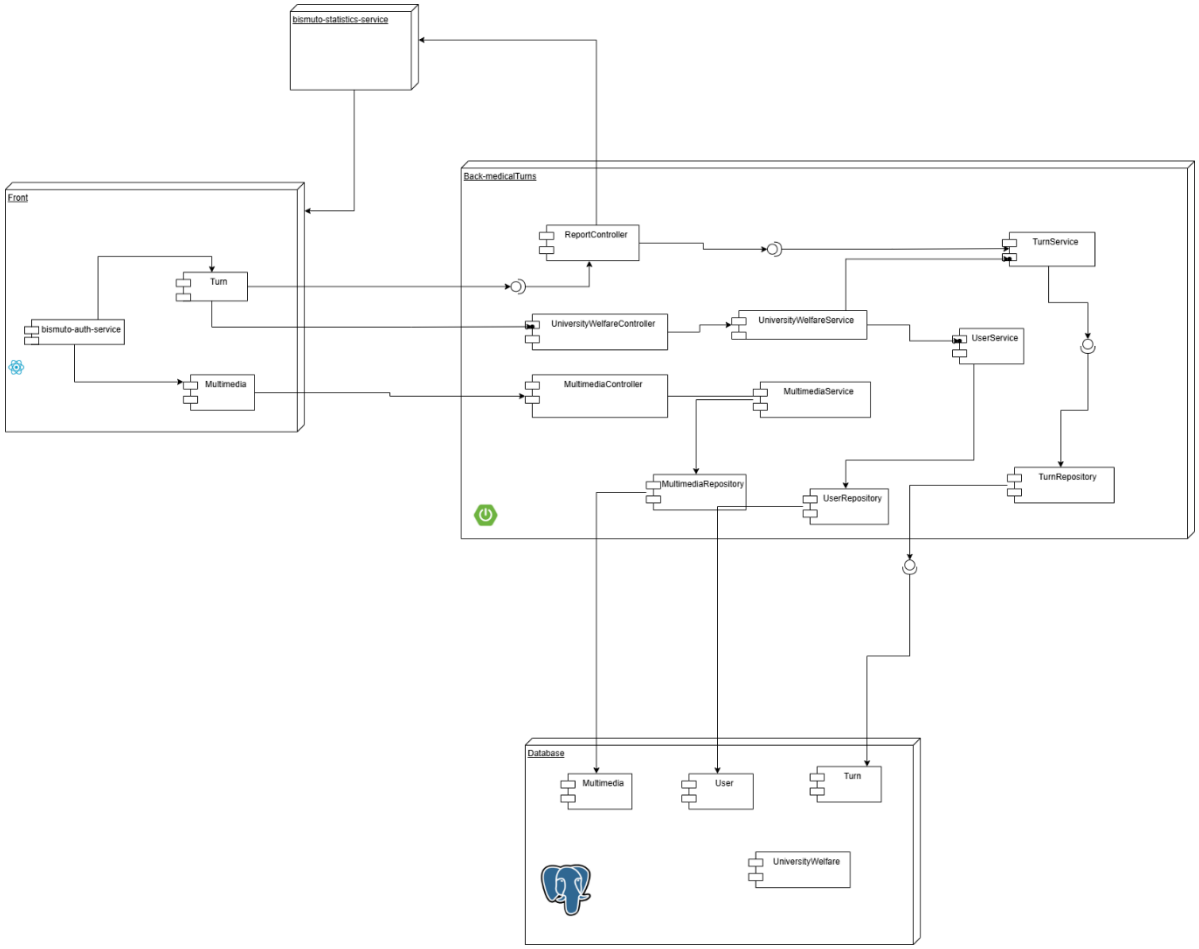
El siguiente diagrama de componentes permite evidenciar el flujo completo y la estructura funcional del sistema MedicalTurns, abarcando desde la interfaz de usuario hasta la integración con servicios externos.

- UniversityWelfareService: Se encarga de la gestión de turnos, incluyendo su creación y actualización, también es responsable de enviar eventos al Bus de Eventos cuando se requiere notificar a otros servicios (como el módulo de notificaciones).
- ReportController: Encargado de recopilar y enviar los datos necesarios para el análisis estadístico. Esta información es procesada por el servicio externo bismuto-statistics-service, el cual genera las estadísticas cuando son solicitadas.
- MultimediaService: Administra los elementos multimedia del módulo, tales como imágenes y videos. Se apoya en el MultimediaController para dar acceso a esos servicios.

Servicios externos:

- bismuto-statistics-service: Recibe eventos relacionados con los reportes y genera estadísticas.
- bismuto-auth-service: Gestiona la autenticación de los usuarios desde el frontend.

Este diseño modular promueve la separación de responsabilidades, la escalabilidad del sistema y una integración flexible con servicios externos.



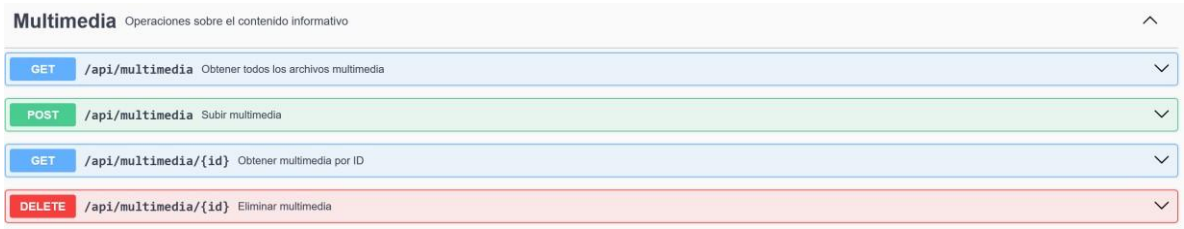
Funcionalidades expuestas

UniversityWelfareController

Bienestar Universitario Operaciones relacionadas con la gestión de bienestar universitario

GET	`/api/turns`	Obtener todos los turnos
POST	`/api/turns`	Crear turno
POST	`/api/turns/enable`	Habilitar turnos
POST	`/api/turns/enable/{speciality}`	Habilitar turnos por especialidad
POST	`/api/turns/disable`	Deshabilitar turnos
POST	`/api/turns/disable/{speciality}`	Deshabilitar turnos por especialidad
POST	`/api/turns/call`	Llamar turno específico
POST	`/api/turns/call-next`	Llamar siguiente turno automáticamente
GET	`/api/turns/{speciality}`	Obtener turnos por especialidad
GET	`/api/turns/current-turn`	Obtener el ultimo turno llamado
GET	`/api/turns/current-turn/{speciality}`	Obtener el ultimo turno llamado dad una especialidad

MultimediaController



ReportController

Manejo de errores

Código HTTP	Mensaje de error	Causa probable
400	"Datos de entrada inválidos"	Validaciones fallidas en el formulario
401	"Usuario no autenticado"	Token inválido o ausente
404	"Turnos no disponibles"	Los turnos están deshabilitados
404	“Especialidad no disponible”	Especialidad deshabilitada
500	"Error interno del servidor"	Fallo inesperado