Escenarios de prueba – ProyectoEPSANDES

Grupo 6

-Juan José Buitrago

-Samuel Andrés Molina

-Emilio Bottagisio

* 1. **Clases de java Spring para los RF y RFC. Entrega 1**

ESCENARIOS DE PRUEBA

1. RF1 – REGISTRAR IPS

Verificar el registro de forma correcta de una ips en el sistema.

Datos de entrada:

* + Nombre: “clínica santa fe”
  + Dirección: “Calle 123 #45 - 67” - teléfono: 3003919306 Resultado esperado:

Se muestra el mensaje "IPS registrada exitosamente".

La IPS aparece en la lista de IPS registradas.

1. RF2 - REGISTRAR UN SERVICIO DE SALUD

Verificar que el sistema permite registrar servicios de salud. Datos de entrada:

* + Código: "001"
  + Nombre: "Consulta General"
  + Descripción: "Atención médica general para diagnóstico y tratamiento" Resultado esperado:

Se muestra el mensaje: "Servicio de salud registrado correctamente". Servicio aparece en la lista de servicios disponibles.

1. RF3 - ASIGNAR UN SERVICIO DE SALUD A UNA IPS

Validar que una IPS puede ofrecer servicios de salud correctamente.

Datos de entrada:

* + IPS: "Clínica San José"
  + Servicio: "Consulta general"

Resultado esperado:

Mensaje de confirmación: "Servicio asignado exitosamente".

La IPS muestra el servicio en su lista de servicios disponibles.

1. RF4 - REGISTRAR MÉDICO

Verificar que se pueden registrar médicos correctamente.

Datos de entrada:

* Nombre: "Dr. Juan"
* Especialidad: "Medicina General"
* IPS: "Clínica San José"

Resultado esperado:

Mensaje de éxito: "Médico registrado correctamente".

Médico aparece en la lista de médicos de la IPS.

1. RF5 - REGISTRAR AFILIADO

Asegurar que los afiliados se registren con todos los datos requeridos.

Datos de entrada:

* + Nombre: "Carlos López"
  + Tipo: "Contribuyente"
  + Beneficiarios: "Hija"

Resultado esperado:

Mensaje: "Afiliado registrado exitosamente".

El afiliado y sus beneficiarios aparecen en la base de datos.

1. RF6 - REGISTRAR UNA ORDEN DE SERVICIO DE SALUD PARA UN AFILIADO POR PARTE DE UN MÉDICO

Validar que un médico pueda emitir una orden de servicio.

Datos de entrada:

Médico: "Dr. Juan"

Afiliado: "Carlos"

Servicio: "Consulta General"

Resultado esperado:

Orden registrada con número único.

Afiliado puede agendar una cita con la orden.

1. RF7 - AGENDAR UN SERVICIO DE SALUD POR PARTE DE UN AFILIADO Asegurar que un afiliado pueda agendar una cita correctamente.

Datos de entrada:

Afiliado: "Carlos "

Servicio: "Consulta General"

Fecha y hora: "2025-02-20 10:00 AM"

Resultado esperado:

Mensaje de confirmación: "Cita agendada exitosamente".

Cita registrada en la agenda de la IPS.

1. RF8 - REGISTRAR LA PRESTACIÓN DE UN SERVICIO DE SALUD Asegurar que la prestación de un servicio se registre correctamente.

Datos de entrada:

Afiliado: "Carlos"

Médico: "Dr. Juan"

Fecha: "2025-02-20"

Resultado esperado:

Registro de la prestación del servicio.

Afiliado puede consultar su historial médico.

1. RFC1 - CONSULTAR LA AGENDA DE DISPONIBILIDAD DE UN SERVICIO Verificar que se puede consultar la disponibilidad de un servicio.

Datos de entrada:

Código de servicio: "001"

Resultado esperado:

Se muestra la disponibilidad de fechas y horas.

1. RFC2 - MOSTRAR LOS 20 SERVICIOS MÁS SOLICITADOS Verificar que se generen estadísticas de los servicios más usados.

Datos de entrada:

Rango de fechas: "2025-01-01 a 2025-02-16"

Resultado esperado:

Lista con los 20 servicios más demandados.

1. RFC3 - MOSTRAR EL ÍNDICE DE USO DE CADA SERVICIO Validar que el índice de uso de servicios se calcule correctamente.

Datos de entrada:

Fecha inicio: "2025-01-01"

Fecha fin: "2025-02-16"

Resultado esperado:

Se muestra el índice de uso (Servicios usados / Servicios disponibles).

1. RFC4 - CONSULTAR UTILIZACIÓN DE SERVICIOS DE UN AFILIADO Validar que un afiliado pueda ver su historial de servicios.

Datos de entrada:

Afiliado: "Carlos"

Rango de fechas: "2025-01-01 a 2025-02-16"

Resultado esperado:

Lista de servicios con fecha, médico y IPS.

* 1. **Clases de java Spring para RF9(versión transaccional), RFC5 y RFC6**

RF1: Registrar IPS

Clase: IPSLogic

Descripción: Gestiona operaciones relacionadas con las Instituciones Prestadoras de Salud (IPS).

Método relevante:

public IPSEntity registrarIPS(IPSInputDTO ipsInputDTO)

Registra una nueva IPS en la base de datos.

Requiere información como nombre, NIT, dirección y teléfono.

RF2: Registrar Servicio de Salud

Clase: ServicioSaludLogic

Descripción: Permite la creación y gestión de los servicios de salud ofrecidos por las EPS.

Método relevante:

public ServicioSaludEntity registrarServicioSalud(ServicioSaludInputDTO servicioSaludInputDTO)

Guarda un nuevo servicio de salud en la base de datos.

Ejemplos de servicios: consulta general, urgencias, terapias.

RF3: Asignar Servicio de Salud a IPS

Clase: IPSServicioSaludLogic

Descripción: Administra la relación entre IPS y los servicios de salud que estas ofrecen.

Método relevante:

public IPSServicioSaludEntity asignarServicioSalud(Long ipsId, Long servicioSaludId)

Asocia un servicio de salud ya registrado a una IPS determinada.

RFC2: Mostrar 20 Servicios Más Solicitados

Clase: ServicioSaludLogic

Descripción: Consulta las estadísticas de uso de los servicios de salud.

Método relevante:

public List<ServicioSaludEntity> obtenerTop20ServiciosMasSolicitados(Date fechaInicio, Date fechaFin)

Retorna los 20 servicios con mayor número de solicitudes en un período de tiempo.

RFC6: Consulta Disponibilidad (READ COMMITTED)

Clase: AgendamientoLogic

Descripción: Permite consultar la disponibilidad de servicios de salud, con concurrencia controlada.

Método relevante:

@Transactional(isolation = Isolation.READ\_COMMITTED)

public List<AgendaDisponibilidadDTO> consultarDisponibilidadReadCommitted(...)

* Nivel de aislamiento: READ COMMITTED.
* Filtro por rango de fechas, médico y servicio de salud.
* Incluye Thread.sleep(30000); para pruebas de concurrencia.

RF9: Agendar un servicio de salud de forma transaccional

Clase: AgendamientoLogic

Descripción: Administra la creación de agendamientos de servicios de salud de manera transaccional para garantizar la integridad de los datos.

Método relevante:

@Transactional(timeout = 30)

public void insertAgendamiento(Long idAfiliado, Long idServicio, String idMedico, Long idIPS, String fecha, String hora, Long ordenId)

Características:

Utiliza manejo de transacciones para asegurar consistencia en el agendamiento.

Valida si es necesario una orden previa dependiendo del tipo de servicio.

Maneja rollback en caso de fallos en la asignación.

RFC5: Consulta de disponibilidad de servicios de salud (AISLAMIENTO SERIALIZABLE)

Clase: AgendamientoLogic

Descripción: Permite consultar la disponibilidad de agendamientos para un servicio de salud utilizando un nivel de aislamiento SERIALIZABLE, evitando conflictos en accesos concurrentes.

Método relevante:

@Transactional(timeout = 30, isolation = Isolation.SERIALIZABLE)

public Collection<AgendamientoEntity> consultarAgendaConFiltrosSerializable(Long idServicio, String idMedico, Long idAfiliado, LocalDate fechaInicio, LocalDate fechaFin)

Características:

Control estricto de concurrencia con SERIALIZABLE.

Permite filtrar por servicio, médico y afiliado, en un rango de fechas.

* 1. **Escenarios de prueba de concurrencia 1 y 2**

|  |
| --- |
| (Consulta RFC6) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Abrir dos consultas de disponibilidad al mismo tiempo, mientras un tercera agenda el servicio. Mostrar que las lecturas son consistentes y no hay lecturas sucias gracias a READ COMMITTED  (RFC5 SERIALIZABLE vs RF9 Registro Transaccional)  Descripción del escenario: Primero se ejecuta el RFC5: Consulta de la agenda de disponibilidad de un servicio de salud, utilizando aislamiento SERIALIZABLE y simulando 30 segundos de espera (Thread.sleep(30000)). Antes de que pasen los 30 segundos, se ejecuta de manera concurrente el RF9, que es el registro de un nuevo agendamiento (orden de servicio de salud) para un afiliado.   | **Tiempo** | **Acción** | | --- | --- |  |  |  | | --- | --- | | t0 | - Usuario A envía petición GET a /agendamientos/filtros/serializable (RFC5). |  |  |  | | --- | --- | | t5 | - Usuario B, en paralelo, envía una petición POST a /agendamientos/crear (RF9). |  |  |  | | --- | --- | | t30 | - Finaliza la consulta RFC5 (después de dormir 30s). |  |  |  | | --- | --- | | t32 | - Se completa la inserción de la nueva orden de servicio en RF9. | |  |  | |

Descripción de lo sucedido: ¿acaso el componente que implementa RF9 debió esperar a que terminara la ejecución de la consulta RFC5 para poder registrar la orden de servicio?

El registro de la orden de servicio (RF9) no debió esperar a que la consulta RFC5 terminara su ejecución. Al revisar nuevamente la tabla de agendamientos, vemos que no se agrego el nuevo servicio solicitado por RF9. Lo que nos da a entender que el uso de serializable bloquea las filas involucradas por lo que no permite ninguna escritura concurrente hasta que la lectura termine.

(RFC6 READ COMMITTED vs RF9 Registro Transaccional)

Descripción del escenario:

Primero se ejecuta el RFC6: Consulta de la agenda de disponibilidad de un servicio de salud, utilizando aislamiento READ COMMITTED y simulando 30 segundos de espera (Thread.sleep(30000)).

Antes de que pasen los 30 segundos, se ejecuta de manera concurrente el RF9, que registra un nuevo agendamiento para un afiliado.

Línea de tiempo:

Tiempo Acción

t0 - Usuario A envía petición GET a /agendamientos/filtros (RFC6).

t5 - Usuario B, en paralelo, envía una petición POST a /agendamientos/crear (RF9).

t10 - El registro de RF9 se completa exitosamente.

t30 - Finaliza la ejecución de RFC6.

Descripción de lo sucedido: ¿acaso el componente que implementa RF9 debió esperar a que terminara la ejecución de la consulta RFC6 para poder registrar la orden de servicio?

El registro de la orden de servicio (RF9) no tuvo que esperar la finalización de RFC6. Lo que nos da a entender que el nivel de aislamiento Read Commited permite que nuevas escrituras se realicen mientras una consulta está leyendo los datos previamente confirmados. Al final RFC6 no incluye el nuevo agendamiento registrado por RF9.