UF2404 - Análisis Funcional y Diseño Diagrama UML de Clases para Java

Anthony Torres

A continuación, se presenta un conjunto de ejercicios que deberán ser desarrollados en grupos de 2 o 3 personas. Cada grupo será responsable de realizar el análisis funcional y el diseño detallado del diagrama UML de clases para cada uno de los problemas planteados. El objetivo de estos ejercicios es aplicar los conceptos de diseño de diagramas UML de clases para java, identificando las clases, relaciones, atributos, métodos y otros elementos clave que permitirán desarrollar el programa en cuestión de manera eficiente y efectiva. Estos ejercicios también buscan fomentar el trabajo en equipo y la colaboración para resolver problemas complejos de diseño de sistemas.

Ejercicio 1:

La empresa XYZ busca desarrollar una aplicación web para la gestión integral de inventarios. Esta aplicación deberá permitir a los usuarios realizar el registro de nuevos productos, con detalles que incluyen el **ID de producto**, **nombre del producto**, **descripción**, **categoría**, **precio unitario**, **cantidad en stock**, **proveedor**, **fecha de adquisición** y **fecha de caducidad**.

Además, el sistema permitirá la búsqueda avanzada de productos utilizando filtros como **nombre**, **categoría**, **rango de precio**, **disponibilidad** (en stock/no en stock) y **fecha de caducidad**. La aplicación deberá generar reportes periódicos sobre el estado del inventario, con opción de descarga en formato PDF y Excel. Estos reportes incluirán información como **productos con baja cantidad**, **productos próximos a caducar**, y **valor total del inventario**.

El sistema también enviará **notificaciones automáticas** por correo electrónico cuando un producto esté cerca de su fecha de caducidad o cuando las existencias sean bajas (definiendo un umbral específico por producto). Para asegurar la protección de la información, el sistema implementará un control de acceso con roles de usuario, como **Administrador**, **Gestor de Inventario** y **Usuario Regular**, cada uno con permisos específicos.

En cuanto a la interfaz de usuario, deberá ser responsiva y accesible desde computadoras de escritorio, tabletas y teléfonos móviles. El sistema debe ser escalable para soportar un crecimiento en el número de productos y usuarios, sin degradación del rendimiento.

Consideraciones para el diseño de la base de datos:

- **Tablas sugeridas:** Productos, Categorías, Proveedores, Usuarios, Roles, Notificaciones, Reportes.
- **Relaciones:** Un producto pertenece a una categoría, un producto es suministrado por un proveedor, un reporte es generado por un usuario.
- **Atributos clave:** ID de producto, ID de categoría, ID de proveedor, ID de usuario, fecha de reporte.

1. Análisis Funcional

Objetivos del Sistema:

- **Gestión de Productos**: Registro, búsqueda, actualización y eliminación de productos.
- **Reportes de Inventario**: Generación de reportes sobre inventario bajo, productos por caducar, valor total del inventario.
- **Notificaciones Automáticas**: Envío de notificaciones cuando productos estén por caducar o existencias sean bajas.
- **Control de Acceso por Roles**: Administrador, Gestor de Inventario, Usuario Regular.
- **Interfaz de Usuario**: Responsiva y accesible en diferentes dispositivos.
- **Escalabilidad**: Soporte a un número creciente de productos y usuarios.

Funcionalidades Clave:

- **Registro de Productos**: ID, nombre, descripción, categoría, precio unitario, cantidad en stock, proveedor, fecha de adquisición, fecha de caducidad.
- **Búsqueda Avanzada**: Filtros de nombre, categoría, rango de precio, disponibilidad, fecha de caducidad.
- **Reportes**: Productos con baja cantidad, próximos a caducar, valor total del inventario.
- **Notificaciones**: Automáticas por correo electrónico para productos con baja existencia o cercanos a caducar.
- **Control de Acceso**: Múltiples roles con permisos específicos.

Entidades del Sistema:

- **Producto**: ID, nombre, descripción, categoría, precio unitario, cantidad en stock, proveedor, fecha de adquisición, fecha de caducidad.
- **Categoría**: ID, nombre, descripción.
- **Proveedor**: ID, nombre, dirección, contacto.
- **Usuario**: ID, nombre, email, rol.
- **Rol**: ID, nombre (Administrador, Gestor de Inventario, Usuario Regular).
- **Notificación**: ID, tipo (caducidad o baja cantidad), fecha, producto asociado.
- **Reporte**: ID, tipo de reporte, fecha de generación, usuario que lo generó.
- **Inventario**: Puede considerarse una entidad virtual que agrupa productos.

2. Diagrama UML de Clases

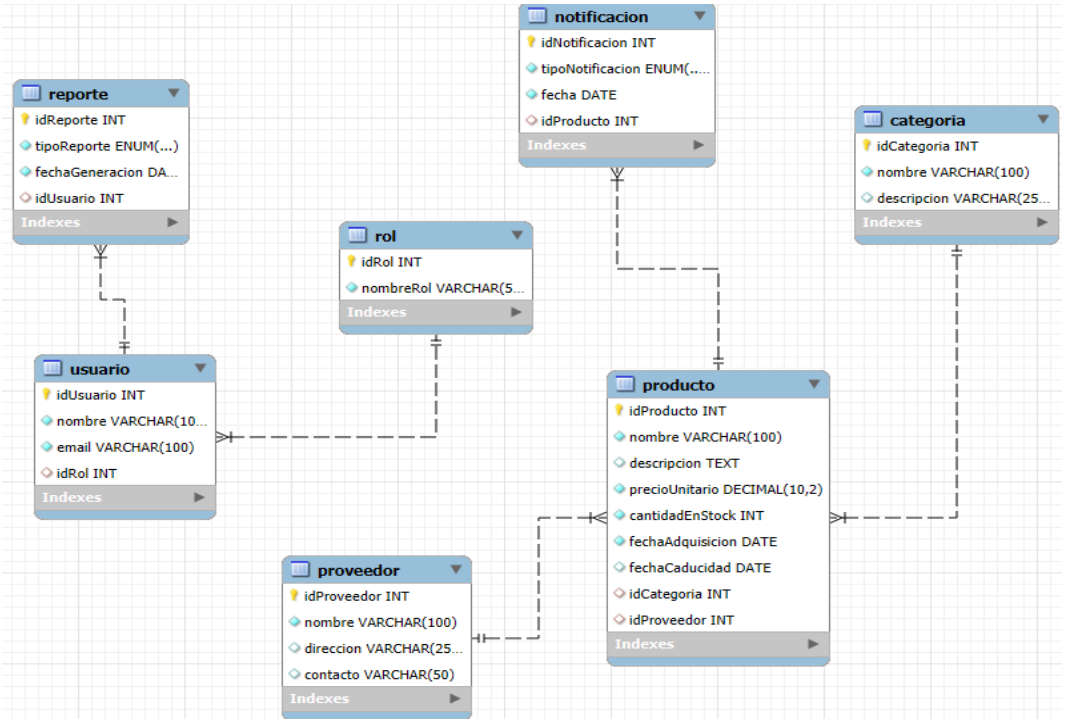
En el diseño del diagrama UML de clases para esta aplicación, identificamos las principales clases, sus atributos, métodos, y las relaciones entre ellas.

Clases principales:

1. **Producto**:
    - Atributos: idProducto, nombre, descripcion, precioUnitario, cantidadEnStock, fechaAdquisicion, fechaCaducidad

- Métodos: registrarProducto(), actualizarProducto(), eliminarProducto(), buscarProducto()

2. **Categoria**:
   - Atributos: idCategoria, nombre, descripcion
   - Métodos: agregarCategoria(), eliminarCategoria(), actualizarCategoria()
3. **Proveedor**:
   - Atributos: idProveedor, nombre, direccion, contacto
   - Métodos: agregarProveedor(), eliminarProveedor(), actualizarProveedor()
4. **Usuario**:
   - Atributos: idUsuario, nombre, email, rol
   - Métodos: crearUsuario(), actualizarUsuario(), eliminarUsuario(), iniciarSesion()
5. **Rol**:
   - Atributos: idRol, nombreRol (Administrador, GestorInventario, UsuarioRegular)
   - Métodos: definirPermisos()
6. **Notificacion**:
   - Atributos: idNotificacion, tipoNotificacion, fecha, producto
   - Métodos: enviarNotificacion()
7. **Reporte**:
   - Atributos: idReporte, tipoReporte (productos bajos, próximos a caducar), fechaGeneracion, usuarioGenerador
   - Métodos: generarReporte(), descargarPDF(), descargarExcel()



Ejercicio 2:

La Universidad ABC está en proceso de desarrollar un sistema de gestión académica en línea que facilitará la administración de cursos, tareas y exámenes. Los estudiantes podrán inscribirse en cursos, los cuales estarán compuestos por **un ID de curso**, **nombre del curso**, **descripción**, **fecha de inicio y fin**, **profesor responsable** y **cantidad máxima de estudiantes**.

Los estudiantes tendrán acceso a materiales de estudio asociados a cada curso, como **documentos**, **videos**, y **enlaces a recursos externos**. Además, podrán enviar tareas, que estarán registradas con **un ID de tarea**, **título**, **descripción**, **fecha de entrega** y **estado de revisión** (pendiente, revisada, aprobada). Los profesores podrán gestionar cursos, asignar tareas, calificar trabajos y publicar resultados. Cada calificación estará vinculada a un **ID de estudiante**, **ID de tarea** y **nota obtenida**.

El sistema también integrará un foro de discusión por curso, donde estudiantes y profesores podrán intercambiar mensajes. Será posible integrar el sistema con bases de datos bibliográficas externas y plataformas de e-learning. La seguridad se garantizará mediante autenticación de dos factores y cifrado de datos sensibles, asegurando la privacidad de los datos de los estudiantes y profesores.

La interfaz de usuario será intuitiva y responsiva, compatible con múltiples navegadores y dispositivos, y el sistema deberá ser escalable para soportar un aumento en el número de usuarios y cursos.

Consideraciones para el diseño de la base de datos:

- **Tablas sugeridas:** Cursos, Estudiantes, Profesores, Tareas, Calificaciones, Materiales, Foros, Mensajes.
- **Relaciones:** Un curso tiene un profesor, un curso tiene múltiples estudiantes inscritos, un estudiante puede enviar múltiples tareas, un profesor califica múltiples tareas.
- **Atributos clave:** ID de curso, ID de profesor, ID de estudiante, ID de tarea, ID de material.

1. Análisis Funcional

El sistema de gestión académica en línea debe cubrir las siguientes funcionalidades clave:

Funcionalidades del sistema:

1. **Gestión de Cursos**:
    - Profesores crean y gestionan cursos con detalles como ID, nombre, descripción, fechas de inicio/fin y límite de estudiantes.
    - Estudiantes pueden inscribirse en los cursos hasta que se alcance la cantidad máxima.
2. **Gestión de Materiales de Estudio**:
    - Los cursos tienen materiales asociados (documentos, videos, enlaces externos) que los estudiantes pueden visualizar.
3. **Gestión de Tareas**:
    - Profesores asignan tareas con ID, título, descripción, fecha de entrega y estado (pendiente, revisada, aprobada).
    - Estudiantes envían tareas y los profesores pueden revisarlas y calificarlas.
4. **Calificaciones**:
    - Cada tarea enviada por un estudiante puede recibir una calificación que asocia el ID de estudiante, ID de tarea y la nota obtenida.
5. **Foros de Discusión**:
    - Cada curso tiene un foro donde los estudiantes y profesores pueden intercambiar mensajes.
6. **Autenticación y Seguridad**:
    - Se implementará autenticación de dos factores para garantizar la seguridad de los datos de los usuarios.
    - Los datos sensibles estarán cifrados.

7. **Escalabilidad y Responsividad**:
    - El sistema debe ser escalable para soportar el aumento de usuarios y cursos.
    - La interfaz será compatible con múltiples dispositivos y navegadores.

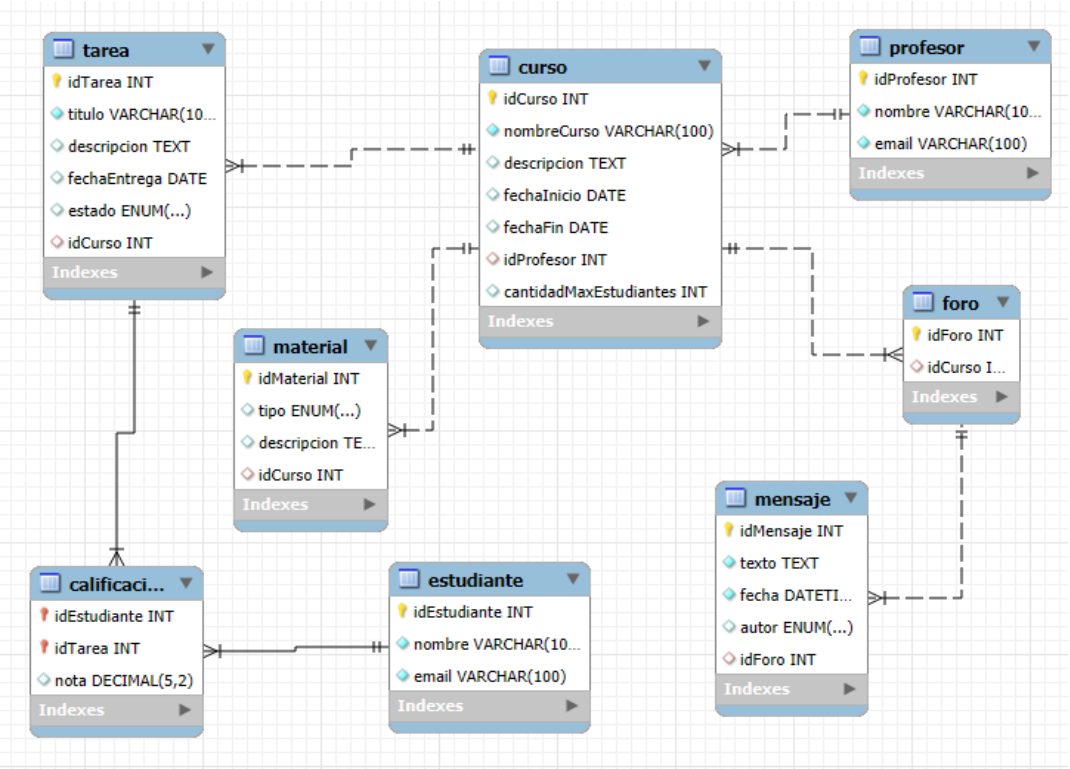
Entidades del sistema:

- **Curso**: ID, nombre, descripción, fechas de inicio y fin, profesor responsable, cantidad máxima de estudiantes.
- **Estudiante**: ID, nombre, email, cursos inscritos.
- **Profesor**: ID, nombre, email, cursos que gestiona.
- **Tarea**: ID, título, descripción, fecha de entrega, estado (pendiente, revisada, aprobada).
- **Calificación**: ID de estudiante, ID de tarea, nota.
- **Material**: ID, tipo (documento, video, enlace), descripción, curso asociado.
- **Foro**: ID de curso, mensajes.
- **Mensaje**: ID de mensaje, texto, fecha, autor (estudiante o profesor).

2. Diagrama UML de Clases

Clases principales:

1. **Curso**:
    - Atributos: idCurso, nombreCurso, descripcion, fechaInicio, fechaFin, profesorResponsable, cantidadMaxEstudiantes.
    - Métodos: inscribirEstudiante(), asignarMaterial(), crearForo().
2. **Estudiante**:
    - Atributos: idEstudiante, nombre, email.
    - Métodos: inscribirseCurso(), enviarTarea().
3. **Profesor**:
    - Atributos: idProfesor, nombre, email.
    - Métodos: crearCurso(), asignarTarea(), calificarTarea().
4. **Tarea**:
    - Atributos: idTarea, titulo, descripcion, fechaEntrega, estado.
    - Métodos: entregarTarea(), actualizarEstado(), calificarTarea().
5. **Calificacion**:
    - Atributos: idEstudiante, idTarea, nota.
    - Métodos: asignarNota().
6. **Material**:
    - Atributos: idMaterial, tipo, descripcion, idCurso.
    - Métodos: agregarMaterial().
7. **Foro**:
    - Atributos: idForo, idCurso.
    - Métodos: agregarMensaje().
8. **Mensaje**:
    - Atributos: idMensaje, texto, fecha, autor.
    - Métodos: publicarMensaje().



Ejercicio 3:

La cadena de supermercados SuperMart desea implementar una aplicación móvil para compras en línea y gestión de entregas a domicilio. Los clientes deberán poder buscar productos utilizando criterios como **nombre**, **categoría**, **marca**, y **precio**. Cada producto tendrá un **ID de producto**, **nombre**, **descripción**, **categoría**, **precio**, **cantidad disponible** y **ID de promoción** (si aplica).

Los clientes podrán agregar productos a su carrito de compras, especificando la **cantidad** deseada. Posteriormente, podrán seleccionar una franja horaria para la entrega y proceder al pago utilizando diferentes métodos como **tarjeta de crédito**, **PayPal** o **transferencia bancaria**.

La aplicación permitirá a los administradores actualizar el inventario en tiempo real, gestionar precios y promociones, y visualizar reportes de ventas que incluirán **ventas por categoría**, **ventas por periodo** y **productos más vendidos**. Además, deberán poder gestionar las cuentas de usuario y configuraciones de seguridad, incluyendo roles y permisos.

La seguridad de la aplicación será reforzada con autenticación de usuarios y cifrado de datos sensibles, especialmente la información financiera. La aplicación debe ser capaz de manejar un alto volumen de transacciones durante picos de demanda, y ser escalable para soportar el crecimiento en usuarios y transacciones.

Consideraciones para el diseño de la base de datos:

- **Tablas sugeridas:** Productos, Categorías, Carritos, Pedidos, Clientes, Promociones, Métodos de Pago, Usuarios, Roles, Reportes.

- **Relaciones:** Un pedido pertenece a un cliente, un producto puede estar en múltiples pedidos, un carrito puede contener múltiples productos, un administrador gestiona productos y promociones.
- **Atributos clave:** ID de producto, ID de categoría, ID de cliente, ID de pedido, ID de promoción.

1. Análisis Funcional

El sistema de compras en línea para SuperMart debe incluir las siguientes funcionalidades:

Funcionalidades del sistema:

1. **Gestión de Productos**:
   - Administradores pueden agregar, actualizar y eliminar productos, que incluyen detalles como ID, nombre, descripción, categoría, precio, cantidad disponible y promociones aplicables.
2. **Carrito de Compras**:
   - Los clientes pueden agregar productos a un carrito, especificando la cantidad deseada.
   - Los clientes pueden revisar y modificar su carrito antes de proceder al pago.
3. **Gestión de Pedidos**:
   - Los pedidos son creados por los clientes al confirmar la compra.
   - Cada pedido está asociado a un cliente y puede contener múltiples productos.
4. **Métodos de Pago**:
   - Los clientes pueden elegir entre diferentes métodos de pago (tarjeta de crédito, PayPal, transferencia bancaria) al finalizar la compra.
5. **Gestión de Promociones**:
   - Los administradores pueden crear y gestionar promociones que se aplican a productos.
6. **Reportes de Ventas**:
   - El sistema permite a los administradores generar reportes sobre ventas por categoría, periodo y productos más vendidos.
7. **Gestión de Usuarios**:
   - Se gestionan cuentas de usuario y configuraciones de seguridad, incluyendo roles y permisos.
8. **Seguridad**:
   - La aplicación implementa autenticación de usuarios y cifrado de datos sensibles.
9. **Escalabilidad**:
   - El sistema está diseñado para manejar un alto volumen de transacciones y soportar el crecimiento de usuarios.

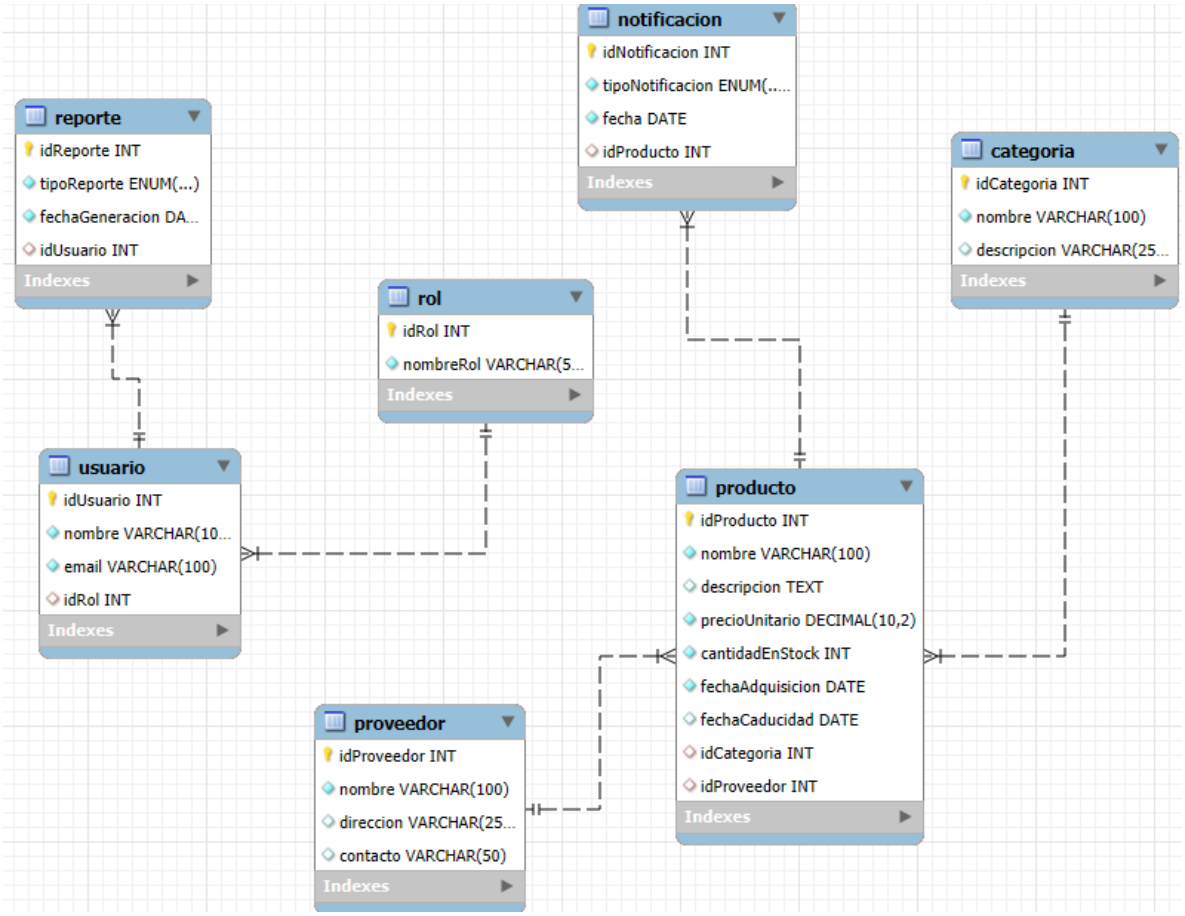
2. Diagrama UML de Clases

Clases principales:

1. **Producto**:
   - Atributos: idProducto, nombre, descripcion, categoria, precio, cantidadDisponible, idPromocion.
   - Métodos: actualizarPrecio(), actualizarInventario().
2. **Categoria**:
   - Atributos: idCategoria, nombre.

- Métodos: agregarCategoria().

3. **Carrito**:
   - Atributos: idCarrito, idCliente.
   - Métodos: agregarProducto(), eliminarProducto(), calcularTotal().
4. **Pedido**:
   - Atributos: idPedido, fecha, idCliente, metodoPago, estado.
   - Métodos: confirmarPedido(), cancelarPedido().
5. **Cliente**:
   - Atributos: idCliente, nombre, email, telefono.
   - Métodos: registrar(), iniciarSesion().
6. **Promocion**:
   - Atributos: idPromocion, descripcion, descuento.
   - Métodos: aplicarPromocion().
7. **MetodoPago**:
   - Atributos: idMetodo, tipo.
   - Métodos: agregarMetodoPago().
8. **Usuario**:
   - Atributos: idUsuario, nombre, email, idRol.
   - Métodos: registrarUsuario(), asignarRol().
9. **Rol**:
   - Atributos: idRol, nombreRol.
   - Métodos: agregarRol().
10. **Reporte**:
    - Atributos: idReporte, tipoReporte, fechaGeneracion.
    - Métodos: generarReporte().



Ejercicio 4:

La empresa TechGiant está desarrollando un sistema de reservas en línea para salas de conferencias. Este sistema permitirá a los usuarios ver la disponibilidad de salas en tiempo real, con detalles como **ID de sala**, **ubicación**, **capacidad máxima**, **equipamiento disponible** (por ejemplo, proyectores, sistemas de videoconferencia), y **horarios disponibles**.

Los usuarios podrán realizar reservas especificando la **fecha**, **hora de inicio y fin**, y **ID de usuario** que realiza la reserva. Será posible filtrar las salas por ubicación, capacidad y equipamiento. Además, el sistema enviará recordatorios automáticos por correo electrónico y SMS a los usuarios con reservas programadas.

Los administradores del sistema tendrán la capacidad de gestionar la información de las salas, editar detalles del equipamiento disponible y acceder a reportes detallados sobre el uso de las salas, como **tasa de ocupación**, **frecuencia de uso** y **problemas reportados**. El sistema deberá asegurar la privacidad de los datos de los usuarios mediante cifrado y autenticación de usuarios con diferentes niveles de acceso.

El sistema debe ser altamente disponible y capaz de manejar múltiples reservas simultáneas sin afectar el rendimiento. La interfaz será responsiva, compatible con dispositivos móviles y fácil de usar.

Consideraciones para el diseño de la base de datos:

- **Tablas sugeridas:** Salas, Equipamientos, Reservas, Usuarios, Recordatorios, Reportes.
- **Relaciones:** Una sala puede tener múltiples equipamientos, una reserva pertenece a un usuario, un administrador gestiona las salas y equipamientos.
- **Atributos clave:** ID de sala, ID de equipamiento, ID de reserva, ID de usuario.

1. Análisis Funcional

Funcionalidades del sistema:

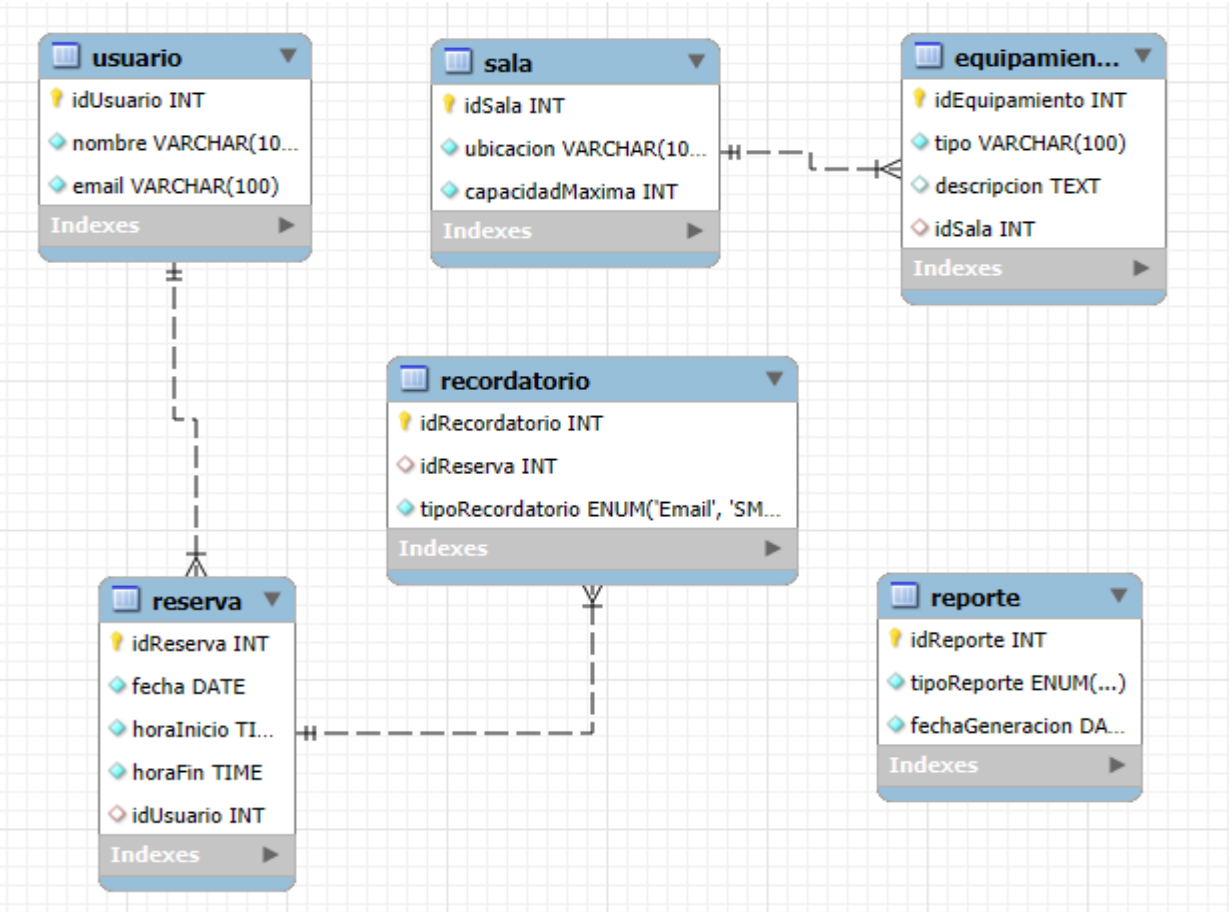
1. **Gestión de Salas**:
    - Los administradores pueden agregar, actualizar y eliminar salas, especificando detalles como ID, ubicación, capacidad máxima y equipamiento disponible.
2. **Reserva de Salas**:
    - Los usuarios pueden ver la disponibilidad de salas en tiempo real.
    - Los usuarios pueden realizar reservas especificando la fecha, hora de inicio y fin, y su ID de usuario.
3. **Filtros de Búsqueda**:
    - Los usuarios pueden filtrar las salas por ubicación, capacidad y equipamiento.
4. **Recordatorios Automáticos**:
    - El sistema enviará recordatorios a los usuarios por correo electrónico y SMS sobre sus reservas programadas.
5. **Reportes de Uso**:
    - Los administradores pueden acceder a reportes detallados sobre el uso de las salas, incluyendo tasa de ocupación y problemas reportados.
6. **Seguridad**:
    - El sistema implementará cifrado de datos y autenticación de usuarios, asegurando la privacidad y el acceso adecuado según el rol.

7. **Interfaz de Usuario**:
    - La interfaz será responsiva y fácil de usar, compatible con dispositivos móviles.

2. Diagrama UML de Clases

Clases principales:

1. **Sala**:
    - Atributos: idSala, ubicacion, capacidadMaxima.
    - Métodos: actualizarSala(), verDisponibilidad().
2. **Equipamiento**:
    - Atributos: idEquipamiento, tipo, descripcion.
    - Métodos: agregarEquipamiento(), actualizarEquipamiento().
3. **Reserva**:
    - Atributos: idReserva, fecha, horaInicio, horaFin, idUsuario.
    - Métodos: realizarReserva(), cancelarReserva().
4. **Usuario**:
    - Atributos: idUsuario, nombre, email.
    - Métodos: registrar(), iniciarSesion().
5. **Recordatorio**:
    - Atributos: idRecordatorio, idReserva, tipoRecordatorio.
    - Métodos: enviarRecordatorio().
6. **Reporte**:
    - Atributos: idReporte, tipoReporte, fechaGeneracion.
    - Métodos: generarReporte().



Ejercicio 5:

El gobierno local busca desarrollar un sistema de gestión de trámites en línea que permita a los ciudadanos solicitar permisos, realizar pagos y gestionar citas con departamentos municipales. Cada ciudadano podrá crear una cuenta personal donde se registrarán sus datos, como **ID de usuario**, **nombre**, **dirección**, **correo electrónico**, y **documentos cargados**. Desde su cuenta, podrán presentar solicitudes de permisos con información detallada como **tipo de permiso**, **fecha de solicitud**, **estado de la solicitud**, y **documentación adjunta**.

El sistema permitirá la realización de pagos en línea, registrando detalles como **ID de transacción**, **monto pagado**, **fecha de pago** y **ID de usuario**. También se integrará un calendario para la programación de citas con departamentos específicos, registrando **fecha y hora de la cita**, **departamento** y **motivo de la cita**.

Para los empleados municipales, el sistema facilitará la revisión y aprobación de solicitudes, el manejo de citas y la comunicación directa con los ciudadanos a través de un portal seguro. Se generarán reportes estadísticos que incluirán **número de solicitudes procesadas**, **tiempo promedio de aprobación**, y **recaudación por tasas**.

El sistema deberá cumplir con normativas de protección de datos personales, utilizando cifrado de datos sensibles y autenticación multifactor para los usuarios. La interfaz debe ser intuitiva y accesible desde cualquier dispositivo, incluyendo móviles, tabletas y computadoras de escritorio.

Consideraciones para el diseño de la base de datos:

- **Tablas sugeridas:** Ciudadanos, Solicitudes, Permisos, Pagos, Citas, Departamentos, Empleados, Reportes.
- **Relaciones:** Una solicitud es presentada por un ciudadano, un pago está asociado a una solicitud, una cita es gestionada por un departamento, un empleado procesa múltiples solicitudes.
- **Atributos clave:** ID de usuario, ID de solicitud, ID de permiso, ID de pago, ID de cita.

1. Análisis Funcional

El sistema de gestión de trámites en línea deberá permitir a los ciudadanos gestionar solicitudes, realizar pagos y programar citas con los departamentos municipales. Para los empleados municipales, el sistema permitirá revisar y procesar estas solicitudes. Se deberán cumplir normativas de protección de datos y la interfaz debe ser accesible desde cualquier dispositivo.

Principales funcionalidades del sistema:

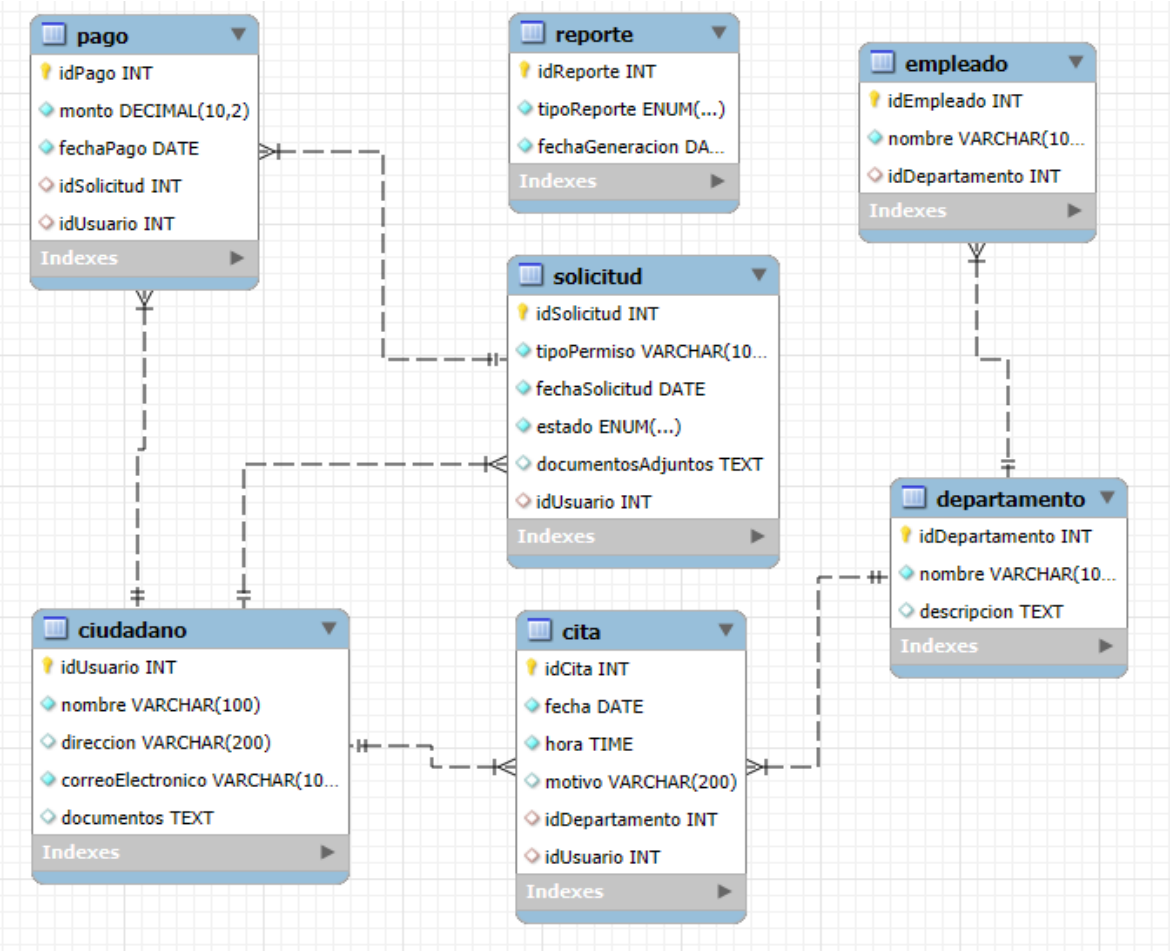
1. **Gestión de Ciudadanos**:
    - Los ciudadanos podrán crear una cuenta personal donde registrarán sus datos personales y cargarán documentos.
2. **Solicitudes de Permisos**:
    - Los ciudadanos podrán presentar solicitudes de permisos que incluirán información como tipo de permiso, estado de la solicitud, y documentos adjuntos.
3. **Realización de Pagos en Línea**:
    - Los ciudadanos podrán realizar pagos en línea, y el sistema registrará los detalles del pago, como ID de transacción, monto pagado y fecha.

4. **Gestión de Citas**:
    - Los ciudadanos podrán programar citas con departamentos específicos, y el sistema registrará la fecha, hora y motivo de la cita.
5. **Revisión y Aprobación de Solicitudes**:
    - Los empleados municipales podrán revisar y aprobar las solicitudes presentadas por los ciudadanos.
6. **Reportes Estadísticos**:
    - El sistema generará reportes sobre el número de solicitudes procesadas, tiempo promedio de aprobación y recaudación de tasas.
7. **Seguridad y Protección de Datos**:
    - El sistema debe implementar cifrado de datos sensibles y autenticación multifactor para garantizar la seguridad de la información.

2. Diagrama UML de Clases

Clases Principales:

1. **Ciudadano**:
    - Atributos: idUsuario, nombre, direccion, correoElectronico, documentos.
    - Métodos: registrar(), iniciarSesion(), actualizarDatos().
2. **Solicitud**:
    - Atributos: idSolicitud, tipoPermiso, fechaSolicitud, estado, documentosAdjuntos.
    - Métodos: crearSolicitud(), modificarSolicitud(), verEstado().
3. **Pago**:
    - Atributos: idPago, monto, fechaPago, idSolicitud, idUsuario.
    - Métodos: realizarPago(), verificarPago().
4. **Cita**:
    - Atributos: idCita, fecha, hora, motivo, idDepartamento, idUsuario.
    - Métodos: agendarCita(), cancelarCita(), modificarCita().
5. **Departamento**:
    - Atributos: idDepartamento, nombre, descripcion.
    - Métodos: gestionarCitas(), procesarSolicitud().
6. **Empleado**:
    - Atributos: idEmpleado, nombre, idDepartamento.
    - Métodos: revisarSolicitud(), aprobarSolicitud().
7. **Reporte**:
    - Atributos: idReporte, tipoReporte, fechaGeneracion.
    - Métodos: generarReporte(), verReporte().



Ejercicio 6:

La compañía de seguros InsureFast está planificando desarrollar un portal en línea para mejorar la experiencia de sus clientes y optimizar sus operaciones. El portal permitirá a los clientes consultar y gestionar sus pólizas de seguro, con detalles como **ID de póliza**, **tipo de seguro** (vehículo, hogar, vida), **fecha de inicio**, **fecha de expiración**, **monto asegurado** y **estado de la póliza**.

Los clientes también podrán presentar reclamaciones, que estarán registradas con un **ID de reclamación**, **fecha de presentación**, **monto reclamado**, **estado de la reclamación** y **documentación adjunta**. Además, podrán actualizar su información personal, realizar pagos de primas y consultar el historial de transacciones.

Los agentes de seguros utilizarán el portal para gestionar las cuentas de los clientes, procesar reclamaciones y generar informes personalizados que incluyan **número de pólizas activas**, **reclamaciones pendientes**, **tendencias de seguros** y **historial de renovaciones**. El sistema ofrecerá funcionalidades automatizadas para la renovación de pólizas y el envío de recordatorios antes de su expiración.

En términos de seguridad, el sistema implementará autenticación robusta y medidas de protección de datos para asegurar la privacidad de la información del cliente, cumpliendo con regulaciones locales e internacionales. El portal debe ser accesible en todo momento, con capacidad para manejar altos volúmenes de usuarios simultáneos, especialmente durante los periodos de renovación de pólizas.

Consideraciones para el diseño de la base de datos:

- **Tablas sugeridas:** Clientes, Pólizas, Reclamaciones, Transacciones, Agentes, Pagos, Recordatorios, Informes.
- **Relaciones:** Un cliente puede tener múltiples pólizas, una póliza puede estar asociada a múltiples reclamaciones, un agente gestiona múltiples clientes y pólizas.
- **Atributos clave:** ID de póliza, ID de cliente, ID de reclamación, ID de agente, ID de transacción.

1. Análisis Funcional

El portal en línea para la compañía de seguros **InsureFast** permitirá a los clientes gestionar sus pólizas de seguro, presentar reclamaciones y realizar pagos de primas, mientras que los agentes podrán gestionar cuentas de clientes y procesar las reclamaciones.

Principales funcionalidades del sistema:

1. **Gestión de Pólizas**:
   - Los clientes podrán consultar sus pólizas de seguro con detalles como tipo de seguro, fecha de inicio, fecha de expiración, monto asegurado y estado de la póliza.
2. **Reclamaciones**:
   - Los clientes podrán presentar reclamaciones, registrar el monto reclamado y adjuntar documentación.
3. **Pagos de Primas**:
   - Los clientes podrán realizar pagos de primas asociadas a sus pólizas y consultar el historial de transacciones.
4. **Actualización de Información Personal**:
   - Los clientes podrán actualizar su información personal, como dirección, teléfono y correo electrónico.
5. **Gestión por parte de Agentes**:
   - Los agentes podrán gestionar las cuentas de los clientes, procesar reclamaciones y generar informes personalizados sobre el estado de pólizas y reclamaciones.
6. **Renovación de Pólizas**:
   - El sistema permitirá la renovación automática de pólizas y enviará recordatorios antes de la expiración.
7. **Seguridad y Cumplimiento de Normativas**:
   - El sistema implementará medidas de autenticación robusta y encriptación para cumplir con las normativas de protección de datos.
8. **Alta Disponibilidad y Escalabilidad**:
   - El sistema debe estar disponible 24/7 y ser capaz de manejar un alto volumen de usuarios, especialmente durante los periodos de renovación de pólizas.

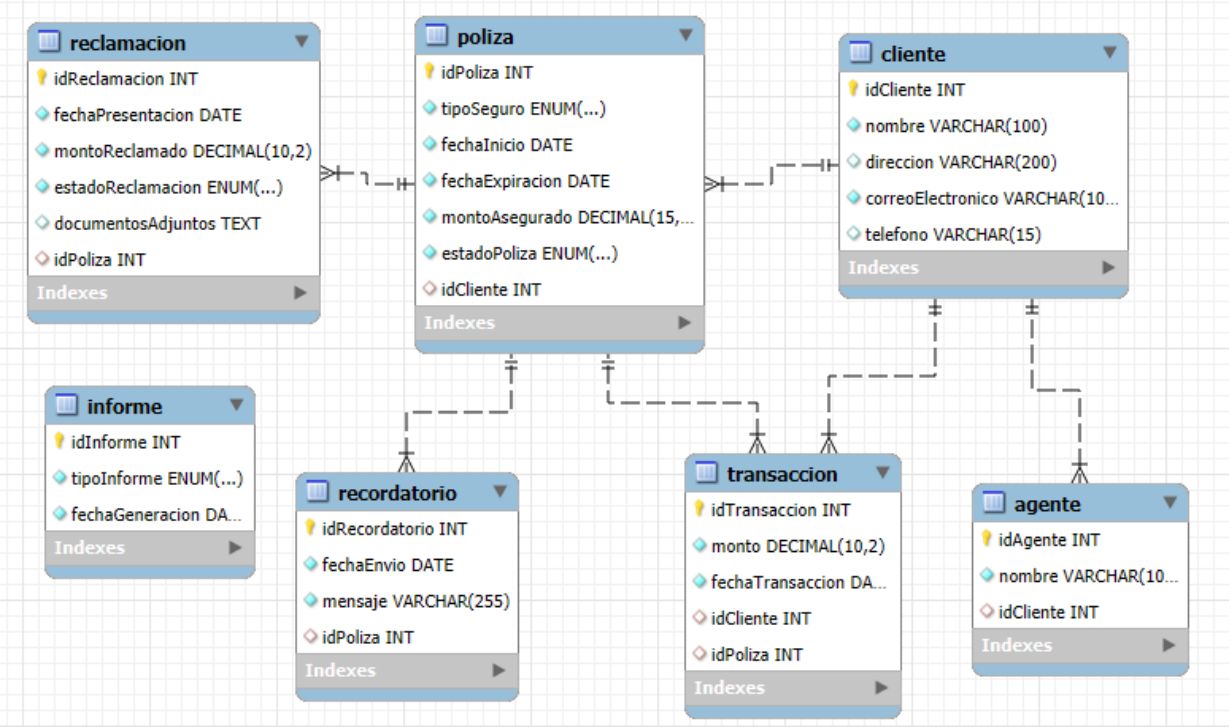
2. Diagrama UML de Clases

Clases Principales:

1. **Cliente**:
   - Atributos: idCliente, nombre, direccion, correoElectronico, telefono.
   - Métodos: consultarPolizas(), actualizarInfoPersonal(), presentarReclamacion().
2. **Poliza**:
   - Atributos: idPoliza, tipoSeguro, fechaInicio, fechaExpiracion, montoAsegurado, estadoPoliza.

- Métodos: consultarPoliza(), renovarPoliza().

3. **Reclamacion**:
   - Atributos: idReclamacion, fechaPresentacion, montoReclamado, estadoReclamacion, documentosAdjuntos, idPoliza.
   - Métodos: registrarReclamacion(), modificarReclamacion(), verEstadoReclamacion().
4. **Transaccion**:
   - Atributos: idTransaccion, monto, fechaTransaccion, idCliente, idPoliza.
   - Métodos: realizarPago(), consultarTransacciones().
5. **Agente**:
   - Atributos: idAgente, nombre, idCliente.
   - Métodos: gestionarPolizas(), procesarReclamaciones(), generarInformes().
6. **Informe**:
   - Atributos: idInforme, tipoInforme, fechaGeneracion.
   - Métodos: generarInforme().
7. **Recordatorio**:
   - Atributos: idRecordatorio, fechaEnvio, mensaje, idPoliza.
   - Métodos: enviarRecordatorio().



Ejercicio 7 (ampliación):

Intergalactic Ventures Inc. es una corporación pionera en la industria de viajes espaciales, especializada en ofrecer vuelos interestelares que conectan distintos planetas del sistema solar y sistemas estelares cercanos. Además, la empresa gestiona una red de hoteles espaciales de lujo ubicados en órbitas planetarias y estaciones espaciales. Estas operaciones requieren una gestión meticulosa de la flota de naves espaciales, los vehículos de transporte planetario, el personal especializado, y los inventarios de suministros esenciales, como alimentos y combustible.

Para operar eficientemente, Intergalactic Ventures necesita un sistema robusto de gestión que permita coordinar múltiples aspectos de sus operaciones:

1. **Naves espaciales**: La compañía posee una flota diversa de naves espaciales, cada una con distintas capacidades de pasajeros y carga, especificaciones técnicas como velocidad máxima, rango de vuelo, y tipos de combustible. Las naves deben ser programadas para viajes entre diferentes planetas, y es fundamental mantener un registro detallado de cada viaje, incluyendo origen, destino, rutas interplanetarias, tiempos estimados de vuelo, y registros de mantenimiento y reparaciones.
2. **Hoteles espaciales**: Los hoteles de la empresa están equipados con tecnología avanzada, incluyendo sistemas de gravedad artificial y soporte vital. Cada hotel tiene un número específico de habitaciones, que varían en tipo y servicios ofrecidos. La empresa debe gestionar la disponibilidad de habitaciones, así como las reservas de los huéspedes, quienes pueden seleccionar servicios adicionales durante su estancia.
3. **Personal especializado**: El personal que opera tanto las naves como los hoteles incluye pilotos, ingenieros, chefs espaciales, y otros roles críticos. Es esencial mantener un registro de los empleados, sus certificaciones, asignaciones a naves o hoteles específicos, y los turnos de trabajo. También es importante gestionar su formación continua y certificación en áreas clave como soporte vital y manejo de emergencias en el espacio.
4. **Vehículos espaciales**: Además de las naves, la empresa utiliza una variedad de vehículos espaciales, como rovers y shuttles, para el transporte dentro de planetas o entre estaciones espaciales. Estos vehículos deben ser asignados a misiones específicas y es necesario registrar su estado y capacidad.
5. **Inventarios de suministros**: La logística espacial implica gestionar inventarios críticos de suministros como alimentos, oxígeno, combustible, y piezas de repuesto. Estos suministros deben ser distribuidos eficientemente entre las naves en vuelo y los hoteles en funcionamiento.
6. **Operaciones interplanetarias**: La coordinación de vuelos entre diferentes sistemas estelares requiere un seguimiento constante de las rutas interplanetarias, el cumplimiento de normativas locales, y la obtención de permisos necesarios para operar en diferentes planetas. Además, es crucial implementar un sistema de monitoreo en tiempo real para las naves, registrando datos de telemetría y el estado de los sistemas vitales durante el vuelo.

1. Análisis Funcional

Intergalactic Ventures Inc. necesita un sistema robusto para gestionar sus operaciones interplanetarias, que incluyen:

- Gestión de **naves espaciales**, **viajes**, **hoteles espaciales**, **reservas de habitaciones**, **personal especializado**, **vehículos planetarios**, **inventarios de suministros**, y **rutas interplanetarias**.
- Además, se debe coordinar la **formación** del personal y el **monitoreo en tiempo real** de las naves y sus sistemas vitales.

Principales funcionalidades del sistema:

1. **Gestión de Naves Espaciales**:
    - Registrar naves espaciales con atributos como capacidad de pasajeros, carga, velocidad máxima, tipo de combustible, y rango de vuelo.
    - Programar viajes, registrar detalles de mantenimiento y estado de cada viaje.

2. **Gestión de Hoteles Espaciales**:
    - Registro de hoteles con atributos como número de habitaciones, tipo de habitaciones y servicios ofrecidos.
    - Gestión de reservas de huéspedes, selección de servicios adicionales y control de disponibilidad de habitaciones.
3. **Gestión de Personal Especializado**:
    - Registro de empleados con roles específicos (pilotos, ingenieros, chefs, etc.), certificaciones y asignaciones.
    - Gestión de turnos de trabajo, formación continua y certificaciones.
4. **Gestión de Vehículos Espaciales**:
    - Registro de vehículos planetarios con capacidades, estado y asignaciones a misiones específicas.
    - Monitoreo de su estado operativo.
5. **Gestión de Inventarios de Suministros**:
    - Control de inventarios críticos (alimentos, oxígeno, combustible, piezas de repuesto).
    - Distribución de suministros entre naves en vuelo y hoteles.
6. **Operaciones Interplanetarias**:
    - Coordinación de rutas interplanetarias, seguimiento de normativas locales y permisos necesarios.
    - Monitoreo en tiempo real de los sistemas vitales y telemetría de las naves.

2. Diagrama UML de Clases

Clases Principales:

1. **NaveEspacial**:
    - **Atributos**: idNave, capacidadPasajeros, capacidadCarga, velocidadMaxima, rangoVuelo, tipoCombustible.
    - **Métodos**: programarViaje(), registrarMantenimiento().
2. **ViajeEspacial**:
    - **Atributos**: idViaje, origen, destino, ruta, tiempoEstimadoVuelo, estadoViaje, idNave.
    - **Métodos**: consultarDetallesViaje(), modificarEstadoViaje().
3. **HotelEspacial**:
    - **Atributos**: idHotel, nombre, ubicacion, numeroHabitaciones, sistemaGravedadArtificial, soporteVital.
    - **Métodos**: consultarDisponibilidad(), gestionarReservas().
4. **Habitacion**:
    - **Atributos**: idHabitacion, tipoHabitacion, precio, estado, idHotel.
    - **Métodos**: consultarEstadoHabitacion(), modificarEstadoHabitacion().
5. **Reserva**:
    - **Atributos**: idReserva, fechaReserva, serviciosAdicionales, idHabitacion, idHuesped, estadoReserva.
    - **Métodos**: crearReserva(), modificarReserva(), cancelarReserva().
6. **Huesped**:
    - **Atributos**: idHuesped, nombre, email, telefono, direccion, nacionalidad.
    - **Métodos**: registrarHuesped(), modificarDatosHuesped().
7. **Empleado**:
    - **Atributos**: idEmpleado, nombre, rol, certificaciones, asignacionActual, turnoTrabajo.
    - **Métodos**: actualizarCertificaciones(), asignarTurno().

8. **VehiculoEspacial**:
    - **Atributos**: idVehiculo, tipoVehiculo, capacidadCarga, estadoVehiculo, idMision.
    - **Métodos**: asignarMision(), modificarEstadoVehiculo().
9. **InventarioSuministros**:
    - **Atributos**: idSuministro, tipoSuministro, cantidadDisponible, ubicacionActual.
    - **Métodos**: actualizarInventario(), distribuirSuministros().
10. **Mision**:
    - **Atributos**: idMision, descripcion, estadoMision, idVehiculo, idNave.
    - **Métodos**: consultarMision(), modificarEstadoMision().
11. **RutaInterplanetaria**:
    - **Atributos**: idRuta, planetaOrigen, planetaDestino, distancia, normativaLocal.
    - **Métodos**: consultarRuta(), modificarRuta().
12. **TelemetriaNave**:
    - **Atributos**: idTelemetria, idNave, velocidadActual, estadoSistemasVitales.
    - **Métodos**: registrarTelemetria(), consultarEstadoNave().