MANUAL TÉCNICO

SISTEMA DE GESTIÓN PARA FUNERARIA

Docente: Felipe Buitrago Carmona

Asignatura: Programación III

Universidad de Caldas

Ingeniería de Sistemas y Computación

Estudiantes

Daner Alejandro Salazar Colorado

Jaime Andrés Cardona Díaz

Santiago García Medina

Fecha: 20/06/2024

Introducción

Bienvenido al manual técnico del Sistema de Gestión para Funeraria, una plataforma diseñada para satisfacer las necesidades complejas y sensibles de la gestión integral de servicios funerarios. Este sistema no solo busca optimizar la administración de recursos y servicios, sino también ofrecer una experiencia compasiva y respetuosa para los familiares y amigos de los difuntos.

Necesidad General del Sistema

La gestión de servicios funerarios implica una serie de procesos críticos que van desde la coordinación de servicios específicos como la cremación o el entierro, hasta la gestión administrativa y financiera asociada. La complejidad de estos procesos requiere una solución tecnológica robusta que permita:

- **Gestión Eficiente**: Coordinar todos los aspectos relacionados con la planificación y ejecución de servicios funerarios de manera ordenada y eficiente.
- **Interacción Personalizada**: Proporcionar una plataforma donde los familiares y amigos puedan interactuar de manera personalizada, registrando detalles específicos y solicitando servicios según sus necesidades particulares.
- **Seguridad y Confidencialidad**: Garantizar la seguridad y confidencialidad de la información sensible relacionada con los difuntos y sus familiares, cumpliendo con estándares rigurosos de protección de datos.

Objetivos del Sistema

El objetivo principal del Sistema de Gestión para Funeraria es ofrecer una plataforma integral que facilite:

- **Registro y Gestión de Clientes**: Centralizar la información de clientes y beneficiarios, permitiendo un acceso rápido y seguro a los datos necesarios para la prestación de servicios.
- **Administración de Servicios y Recursos**: Facilitar la administración de servicios funerarios como la selección de ataúdes, flores, música, entre otros, asegurando una oferta variada y personalizada.
- **Gestión de Pagos y Finanzas**: Automatizar y asegurar la gestión eficiente de transacciones financieras asociadas a los servicios funerarios, garantizando la transparencia y precisión en los registros financieros.
- **Comunicación Efectiva**: Proporcionar un sistema de notificaciones en tiempo real que mantenga informados a los usuarios sobre el estado y detalles de los servicios contratados.

Innovación Tecnológica y Valor Agregado

La adopción de una arquitectura de microservicios y la utilización de tecnologías modernas como Java Spring Boot, AdonisJS, Laravel, Flask y Angular, no solo aseguran un rendimiento óptimo del sistema, sino que también ofrecen las siguientes ventajas:

- **Escalabilidad**: Capacidad para manejar aumentos significativos en la carga de trabajo mediante la escalabilidad horizontal de los microservicios individuales.
- **Flexibilidad**: Permite la integración de nuevas funcionalidades y adaptaciones rápidas a los cambios del mercado y las necesidades del usuario final.
- **Mantenimiento Simplificado**: La modularidad de los microservicios facilita el mantenimiento y la actualización de cada componente de manera independiente, minimizando el impacto en el sistema global.
- **Experiencia de Usuario Mejorada**: A través de interfaces modernas y intuitivas, los usuarios pueden navegar fácilmente por las opciones disponibles y realizar acciones de manera eficiente.

Arquitectura de Software

La arquitectura del sistema está basada en una serie de microservicios independientes, cada uno encargado de una funcionalidad específica. Esta arquitectura permite una alta cohesión y bajo acoplamiento, facilitando el desarrollo, despliegue y mantenimiento del sistema.

Componentes de la Arquitectura

1. **Microservicio de Seguridad**
    - **Tecnología**: Java Spring Boot
    - **Funcionalidad**: Autenticación y autorización de usuarios.
2. **Microservicio de Negocio**
    - **Tecnología**: Adonis JS
    - **Funcionalidad**: Gestión de operaciones centrales de la funeraria.
    - **Integración**: API externa ApiColombia.
3. **Microservicio de Pagos**
    - **Tecnología**: Laravel
    - **Funcionalidad**: Procesamiento de transacciones.
    - **Integración**: API externa ePayCo.
4. **Microservicio de Notificaciones**
    - **Tecnología**: Flask Python
    - **Funcionalidad**: Envío de comunicaciones y actualizaciones en tiempo real.

5. **Frontend**
    - **Tecnología**: Angular
    - **Funcionalidad**: Interfaz de usuario.
    - **Integración**: API externa ApiColombia.

Beneficios de la Arquitectura de Microservicios

La arquitectura de microservicios, empleada en el desarrollo del sistema de gestión para funeraria, ofrece múltiples beneficios cruciales para el rendimiento, escalabilidad y mantenimiento del sistema. A continuación, se detallan estos beneficios en profundidad:

Escalabilidad

1. **Escalabilidad Horizontal**: Cada microservicio puede escalarse de manera independiente según la demanda. Por ejemplo, si el microservicio de pagos experimenta un alto volumen de transacciones, se pueden añadir más instancias de este microservicio sin afectar a los demás componentes del sistema.
2. **Uso Eficiente de Recursos**: Al escalar servicios específicos en lugar de toda la aplicación, se optimiza el uso de recursos, reduciendo costos y mejorando la eficiencia operativa.

Modularidad

1. **Desarrollo Independiente**: Los equipos pueden trabajar en diferentes microservicios de manera independiente y en paralelo. Esto acelera el proceso de desarrollo, ya que no se requiere una coordinación constante entre los equipos para realizar cambios o añadir nuevas funcionalidades.
2. **Facilidad de Mantenimiento**: La modularidad permite una mejor organización del código, facilitando el mantenimiento y la evolución del sistema. Cada microservicio es una unidad autónoma, lo que simplifica la localización y resolución de problemas.

Capacidad de Desarrollo Ágil

1. **Ciclo de Desarrollo Rápido**: Los microservicios permiten realizar despliegues más frecuentes y rápidos. Las actualizaciones y mejoras pueden implementarse en servicios específicos sin necesidad de desplegar toda la aplicación.
2. **Integración Continua y Despliegue Continuo (CI/CD)**: La arquitectura de microservicios facilita la adopción de prácticas CI/CD, permitiendo un desarrollo más ágil y la entrega continua de nuevas funcionalidades y correcciones de errores.

Resiliencia

1. **Aislamiento de Fallos**: En una arquitectura de microservicios, los fallos en un servicio no afectan a los demás. Si un microservicio falla, los demás continúan funcionando, lo que mejora la disponibilidad y confiabilidad del sistema.
2. **Mantenimiento y Actualizaciones Sin Interrupciones**: Se pueden realizar actualizaciones y mantenimientos en microservicios individuales sin necesidad de detener todo el sistema, lo que minimiza el tiempo de inactividad.

Flexibilidad Tecnológica

1. **Uso de Tecnologías Heterogéneas**: Cada microservicio puede desarrollarse utilizando la tecnología más adecuada para su funcionalidad específica. Por ejemplo, se utiliza Java Spring Boot para la seguridad, AdonisJS para la lógica de negocio y Flask para notificaciones. Esta flexibilidad permite aprovechar las fortalezas de diferentes tecnologías.
2. **Facilidad para Adoptar Nuevas Tecnologías**: La adopción de nuevas tecnologías o herramientas en un microservicio específico no requiere una reestructuración completa del sistema, facilitando la innovación y mejora continua.

Seguridad

1. **Gestión Específica de Seguridad**: Los microservicios permiten implementar y gestionar la seguridad de manera granular. Por ejemplo, el microservicio de seguridad desarrollado con Java Spring Boot y MongoDB maneja la autenticación y autorización, implementando medidas avanzadas como la gestión de sesiones y la validación de roles y permisos.
2. **Protección de Datos Sensibles**: La arquitectura de microservicios permite la segregación de datos, lo que facilita la implementación de políticas de seguridad y cumplimiento normativo específicas para cada tipo de dato manejado por los diferentes microservicios.

Eficiencia Operacional

1. **Despliegue en la Nube**: Los microservicios son ideales para entornos de nube, permitiendo una gestión más eficiente de los recursos y el aprovechamiento de servicios de escalado automático, balanceo de carga y recuperación ante desastres.
2. **Monitoreo y Trazabilidad**: La arquitectura de microservicios facilita la implementación de herramientas de monitoreo y trazabilidad específicas para cada servicio, permitiendo una detección temprana de problemas y una respuesta rápida.

Beneficios Específicos de las Tecnologías Utilizadas

1. **Java Spring Boot con MongoDB**: Simplifica el desarrollo de aplicaciones robustas y escalables, con un modelado de datos flexible y un alto rendimiento para la gestión de la autenticación y autorización de usuarios.
2. **AdonisJS con MySQL**: Optimiza el rendimiento y escalabilidad del sistema de negocio central, asegurando un desarrollo ágil y mantenible.
3. **Flask con Servicios de Comunicación de Azure**: Proporciona un sistema ágil, seguro y escalable para el envío de notificaciones, con una infraestructura robusta para manejar altas demandas.
4. **Laravel con MySQL**: Simplifica la integración con APIs de pagos, garantizando la seguridad e integridad de las transacciones financieras, además de ser adaptable a futuras expansiones.
5. **Angular**: Proporciona una interfaz de usuario dinámica y coherente, facilitando la navegación y el uso intuitivo de todas las funcionalidades disponibles en el sistema

Motor de Base de Datos

Para el diseño y desarrollo del Sistema de Gestión para Funeraria, se ha realizado una elección cuidadosa de los motores de base de datos a fin de satisfacer las diversas necesidades del sistema. Se han seleccionado dos tipos de bases de datos, MongoDB y MySQL, cada una optimizada para diferentes aspectos del sistema.

MongoDB

MongoDB es una base de datos NoSQL que se utiliza en el sistema principalmente para manejar la autenticación y autorización de usuarios en el microservicio de seguridad. Las razones para elegir MongoDB incluyen:

1. **Modelado de Datos Flexible**: MongoDB almacena datos en documentos BSON (Binary JSON), lo que permite una mayor flexibilidad en el modelado de datos, adaptándose fácilmente a cambios y evolucionando rápidamente con las necesidades de la aplicación.
2. **Escalabilidad Horizontal**: MongoDB está diseñado para escalar horizontalmente mediante el sharding, lo que significa que los datos pueden distribuirse a través de múltiples servidores, mejorando la capacidad de manejo de grandes volúmenes de datos y aumentando la disponibilidad.
3. **Rendimiento**: MongoDB ofrece un alto rendimiento en términos de velocidad de lectura y escritura, lo que es esencial para las operaciones de autenticación y autorización, que requieren respuestas rápidas.
4. **Gestión de Sesiones y Roles**: La flexibilidad de MongoDB facilita la implementación de complejas estructuras de gestión de sesiones y roles,

permitiendo almacenar y consultar rápidamente los permisos y roles de los usuarios.
5. **Alta Disponibilidad**: Con características como la replicación, MongoDB garantiza que los datos estén disponibles y accesibles, incluso en casos de fallos en los servidores, lo cual es crítico para un sistema que maneja información sensible.

MySQL

MySQL es una base de datos relacional que se utiliza en los microservicios de negocio y de pagos. Las razones para elegir MySQL incluyen:

1. **Integridad de Datos**: MySQL garantiza la integridad y consistencia de los datos mediante el uso de transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), lo cual es crucial para operaciones financieras y de negocio.
2. **Eficiencia en Consultas Complejas**: MySQL es altamente eficiente en la ejecución de consultas SQL complejas, lo que es necesario para manejar la lógica de negocio y las operaciones de pagos, donde se requieren consultas detalladas y reportes financieros precisos.
3. **Soporte y Comunidad**: MySQL tiene una amplia comunidad de usuarios y un soporte sólido, lo que facilita la resolución de problemas y la implementación de nuevas funcionalidades.
4. **Escalabilidad Vertical**: Aunque MySQL también puede escalar horizontalmente, es especialmente conocido por su capacidad de escalar verticalmente, optimizando el rendimiento en servidores más potentes.
5. **Compatibilidad con Frameworks**: MySQL es compatible con una amplia gama de frameworks y lenguajes de programación, incluyendo Laravel y AdonisJS, utilizados en nuestro sistema para el microservicio de pagos y negocio respectivamente. Esto asegura una integración suave y eficiente.

Integración de Bases de Datos

La combinación de MongoDB y MySQL en el sistema permite aprovechar las ventajas de ambos tipos de bases de datos, asegurando que cada componente del sistema esté optimizado para sus respectivas tareas. La arquitectura de microservicios facilita esta integración, permitiendo que cada microservicio utilice el motor de base de datos más adecuado para su funcionalidad específica.

1. **Microservicio de Seguridad con MongoDB**: Utiliza MongoDB para manejar la autenticación y autorización, aprovechando su flexibilidad y rendimiento para gestionar sesiones y roles de usuarios.
2. **Microservicio de Negocio con MySQL**: Utiliza MySQL para gestionar la lógica de negocio central, beneficiándose de la integridad de datos y la eficiencia en consultas complejas.

3. **Microservicio de Pagos con MySQL**: Utiliza MySQL para procesar transacciones financieras, asegurando la precisión y consistencia de los datos de pagos y reembolsos.
4. **Microservicio de Notificaciones**: Aunque no se especifica el uso de una base de datos en particular para este microservicio, se puede optar por MongoDB o una base de datos similar para gestionar el envío de notificaciones en tiempo real, beneficiándose de la flexibilidad y escalabilidad de MongoDB.

Resumen de Ventajas de las Bases de Datos

- **MongoDB**:
  - Flexibilidad en el modelado de datos.
  - Escalabilidad horizontal.
  - Alto rendimiento.
  - Gestión eficiente de sesiones y roles.
  - Alta disponibilidad.
- **MySQL**:
  - Integridad y consistencia de datos.
  - Eficiencia en consultas complejas.
  - Amplio soporte y comunidad.
  - Escalabilidad vertical.
  - Compatibilidad con múltiples frameworks.

La elección de estas bases de datos garantiza que el sistema de gestión para funeraria sea robusto, eficiente y capaz de manejar tanto las operaciones críticas como las necesidades de escalabilidad y flexibilidad.

Diccionario de Datos

Tabla: administrators

Propósito: Almacenar información sobre los administradores del sistema.

- **id:** Un identificador único para cada administrador.
- **Privileges:** Lista de privilegios o permisos asignados al administrador.

- **Responsabilities:** Descripción de las responsabilidades específicas del administrador.

administrators

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
privileges	VARCHAR(255)									NULL	
responsabilities	VARCHAR(255)									NULL	
user_id	VARCHAR(255)									NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: adonis_schema

Propósito: Rastrear las migraciones de la base de datos.

- **id:** Identificador único de cada registro de migración.
- **Name:** Nombre de la migración.
- **Batch:** Número del lote de migración.
- **Migration_time:** Fecha y hora en que se realizó la migración**.**

adonis_schema

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
name	VARCHAR(255)			✓							
batch	INT			✓							
migration_time	TIMESTAMP									CURRENT_TIMESTAMP	

Tabla: adonis_schema_versions

Propósito: Mantener un historial de las versiones del esquema de la base de datos.

- **id:** Identificador único de la versión del esquema.
- **Versión:** Número de la versión del esquema.

adonis_schema_versions

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
version	INT			✓							

Tabla: beneficiaries

Propósito: Almacenar información sobre los beneficiarios de los servicios funerarios.

- **id:** Identificador único de cada beneficiario.
- **Beneficiary_status:** Estado del beneficiario.
- **Customer_id:** Relación con la tabla de clientes, identificando al cliente que designó al beneficiario.
- **Owner_id:** Identificador del propietario del beneficiario en el sistema.

beneficiaries

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
beneficiary_status	VARCHAR(255)									NULL	
customer_id	INT		✓				✓			NULL	
owner_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: burials

Propósito: Almacenar información sobre los servicios de entierro.

- **id:** Identificador único de cada servicio de entierro.
- **Location:** Ubicación del entierro.
- **Burial_type:** Tipo de entierro (tradicional, etc.).
- **Cementery:** Cementerio donde se hará el entierro
- **Wake_room_id:** Relación con la tabla de salas de velación, identificación de la sala para el velatorio del entierro
- **Service_id:** Relación con la table de servicio, identificación del servicio para el velatorio

burials

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
location	VARCHAR(255)									NULL	
burial_type	VARCHAR(255)									NULL	
cementery	VARCHAR(255)									NULL	
wake_room_id	INT		✓				✓			NULL	
service_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: chats

Propósito: Almacenar información sobre las sesiones de chat entre usuarios y personal del servicio.

- **id:** Identificador único de cada sesión de chat.
- **Start_date:** Fecha de inicio del chat
- **State:** Estado del chat

chats

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
start_date	TIMESTAMP									NULL	
state	TINYINT(1)									NULL	
incident_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: comments

Propósito: Almacenar comentarios realizados por usuarios o personal sobre distintos aspectos del servicio.

- **id:** Identificador único de cada comentario.
- **Message:** Contenido del comentario.
- **Send_date:** Fecha del envio del mensaje
- **Incident_id:** Relación con la tabla incidente, idéntica a que incidente se le hizo un comentario

comments

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
message	VARCHAR(255)									NULL	
send_date	TIMESTAMP									NULL	
incident_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: cremations

Propósito: Almacenar información sobre los servicios de cremación.

- **id:** Identificador único de cada servicio de cremación.

- **Cremation_type:** Tipo de cremación (estándar, ecológica, etc.).
- **Destiny_ashes:** Destino de las cenizas

cremations

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
urn_type	VARCHAR(255)									NULL	
destiny_ashes	VARCHAR(255)									NULL	
wake_room_id	INT		✓				✓			NULL	
service_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: customers

Propósito: Almacenar información sobre los clientes que contratan los servicios.

- **id:** Identificador único de cada cliente.
- **address:** Dirección del cliente
- **Phone_number:** Número telefónico del cliente

customers

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
address	VARCHAR(255)									NULL	
phone_number	VARCHAR(255)									NULL	
user_id	VARCHAR(255)									NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: employees

Propósito: Almacenar información sobre los empleados de la funeraria.

- **id:** Identificador único de cada empleado.
- **Position:** Cargo o puesto del empleado.
- **Salary:** Salario del empleado

employees

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
position	VARCHAR(60)			✓							
salary	DOUBLE									NULL	
user_id	VARCHAR(255)									NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: execution_services

Propósito: Almacenar información sobre la ejecución de los servicios funerarios contratados.

- **id:** Identificador único de cada ejecución de servicio.
- **Cost:** Costo del servicio
- **Duration:** Duración del servicio
- **State:** Estado del servicio
- **Incident_id:** Relación con la tabla incidente, identificando al incidente del servicio
- **Service_id:** Relación con la tabla de servicios, identificando al servicio ejecutado.

execution_services

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
cost	INT									NULL	
duration	TIMESTAMP									NULL	
state	TINYINT(1)									NULL	
incident_id	INT		✓				✓			NULL	
service_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: incidents

Propósito: Almacenar información sobre incidentes reportados durante la prestación de servicios.

- **id:** Identificador único de cada incidente.
- **Date_decease:** Fecha del incidente.
- **Place_decease:** Lugar del incidente

- **Cause_decease:** Causa del incidente

incidents

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
date_decease	TIMESTAMP									NULL	
place_decease	VARCHAR(255)									NULL	
cause_decease	VARCHAR(255)									NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: messages

Propósito: Almacenar mensajes intercambiados en las sesiones de chat.

- **id:** Identificador único de cada mensaje.
- **chat_id:** Relación con la tabla de chats, identificando la sesión de chat correspondiente.
- **Information:** Contenido del mensaje.
- **User_id:** Identificador del usuario que manda el mensaje

messages

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
information	VARCHAR(255)									NULL	
chat_id	INT		✓				✓			NULL	
user_id	VARCHAR(255)									NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: notifications

Propósito: Almacenar notificaciones enviadas a los clientes.

- **id:** Identificador único de cada notificación.
- **message:** Contenido del mensaje de la notificación.
- **Date_shipped:** Fecha y hora en que se envió la notificación.
- **Service_id:** Relación con la tabla servicios, identificando la sesión a la que corresponde

notifications

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
message	VARCHAR(255)									NULL	
date_shipped	TIMESTAMP									NULL	
service_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: owners

Propósito: Almacenar información sobre los propietarios de los servicios contratados.

- **id:** Identificador único de cada propietario.
- **Contract_status:** Estado del contrato que solicitó
- **Customer_id:** Herencia de la tabla clientes

owners

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
contract_status	VARCHAR(255)									NULL	
customer_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: pays

Propósito: Almacenar información sobre los pagos realizados.

- **id:** Identificador único de cada pago.
- **Pay_day:** Fecha en que se realizó el pago
- **Amount:** Monto del pago.
- **Subscription_id:** Relación con la tabla de subscripciones, identificando al servicio pagado.

pays

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
pay_day	TIMESTAMP									NULL	
amount	INT									NULL	
subscription_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: plans

Propósito: Almacenar información sobre los diferentes planes de servicios funerarios disponibles.

- **id:** Identificador único de cada plan.
- **Description:** Descripción del plan.
- **Price:** Precio del plan.
- **Duration:** Duración del plan

plans

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
description	VARCHAR(255)			✓							
price	FLOAT(8,2)			✓							
duration	INT			✓							
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: reports

Propósito: Almacenar información sobre los informes generados en el sistema.

- **id:** Identificador único de cada informe.
- **Reporting_date:** Fecha del reporte.
- **Description:** Descripción del reporte.
- **State:** Estado del reporte.
- **Customer_id:** Relación con la tabla clientes, identificando al cliente que hizo el reporte
- **Incident_id:** Relación con la tabla incidentes, identificando que incidente se reportó

reports

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
reporting_date	TIMESTAMP									NULL	
description	VARCHAR(255)									NULL	
state	TINYINT(1)									NULL	
customer_id	INT		✓				✓			NULL	
incident_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: service_plans

Propósito: Relacionar los servicios específicos con los planes a los que pertenecen.

- **id:** Identificador único de cada relación.
- **Status_hiring:** Estado del contrato del servicio con un plan
- **Date_hiring:** Fecha del contrato
- **Date_expiration:** Fecha de expiración del contrato
- **plan_id:** Relación con la tabla de planes, identificando al plan correspondiente.
- **service_id:** Relación con la tabla de servicios, identificando al servicio correspondiente.

service_plans

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
status_hiring	TINYINT(1)									NULL	
date_hiring	TIMESTAMP									NULL	
date_expiration	TIMESTAMP									NULL	
service_id	INT		✓				✓			NULL	
plan_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: services

Propósito: Almacenar información sobre los distintos servicios funerarios ofrecidos.

- **id:** Identificador único de cada servicio.
- **Type:** Tipo de servicio
- **Start_date:** Fecha de inicio del servicio
- **End_date:** Fecha de finalización del servicio

services

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
type	VARCHAR(255)			✓							
start_date	TIMESTAMP			✓							
end_date	TIMESTAMP			✓							
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: sites

Propósito: Almacenar información sobre las sedes asociadas a los servicios funerarios (p. ej., cementerios, salas de velación).

- **id:** Identificador único de cada sede.
- **Name:** Nombre de la sede.
- **Location:** Ubicación de la sede.
- **Email:** Correo de la sede
- **City_id:** Identificación de la ciudad a la que pertenece la sede
- **City_name:** Nombre de la ciudad en cuestión
- **Departament_id:** Identificación del departamento a la que pertenece la sede
- **Departamente_name:** Nombre del departamento en cuestión

sites

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
name	VARCHAR(255)			✓							
location	VARCHAR(255)			✓							
email	VARCHAR(255)			✓							
city_id	VARCHAR(255)			✓							
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	
city_name	VARCHAR(255)			✓							
department_id	VARCHAR(255)			✓							
department_name	VARCHAR(255)			✓							

Tabla: subscriptions

Propósito: Almacenar información sobre las suscripciones de los clientes a los diferentes planes de servicios.

- **id:** Identificador único de cada suscripción.
- **Subscription_type:** Tipo de subscripción

- **Start_date:** Fecha de inicio de la suscripción.
- **End_date:** Fecha de finalización de la suscripción.
- **State:** Estado de la subscripción.
- **Plan_id:** Relación con la tabla de planes, identificando al plan correspondiente.
- **Customer_id:** Relación con la tabla de clientes, identificando al cliente correspondiente.

subscriptions

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
subscription_type	VARCHAR(255)									NULL	
start_date	TIMESTAMP									NULL	
end_date	DATETIME									NULL	
state	TINYINT(1)									NULL	
plan_id	INT		✓				✓			NULL	
customer_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: transfers

Propósito: Almacenar información sobre las transferencias de restos entre diferentes ubicaciones.

- **id:** Identificador único de cada transferencia.
- **Place_origin:** Ubicación de origen.
- **Destination:** Ubicación de destino.
- **Distance:** Distancia por recorrer.
- **Type_vehicle:** Tipo de vehículo.
- **Service_id:** Referencia a la tabla servicio, identificado a que servicio le pertenece esa transferencia

transfers

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
place_origin	VARCHAR(255)									NULL	
destination	VARCHAR(255)									NULL	
distance	FLOAT(8,2)									NULL	
type_vehicle	VARCHAR(255)									NULL	
service_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Tabla: wake_rooms

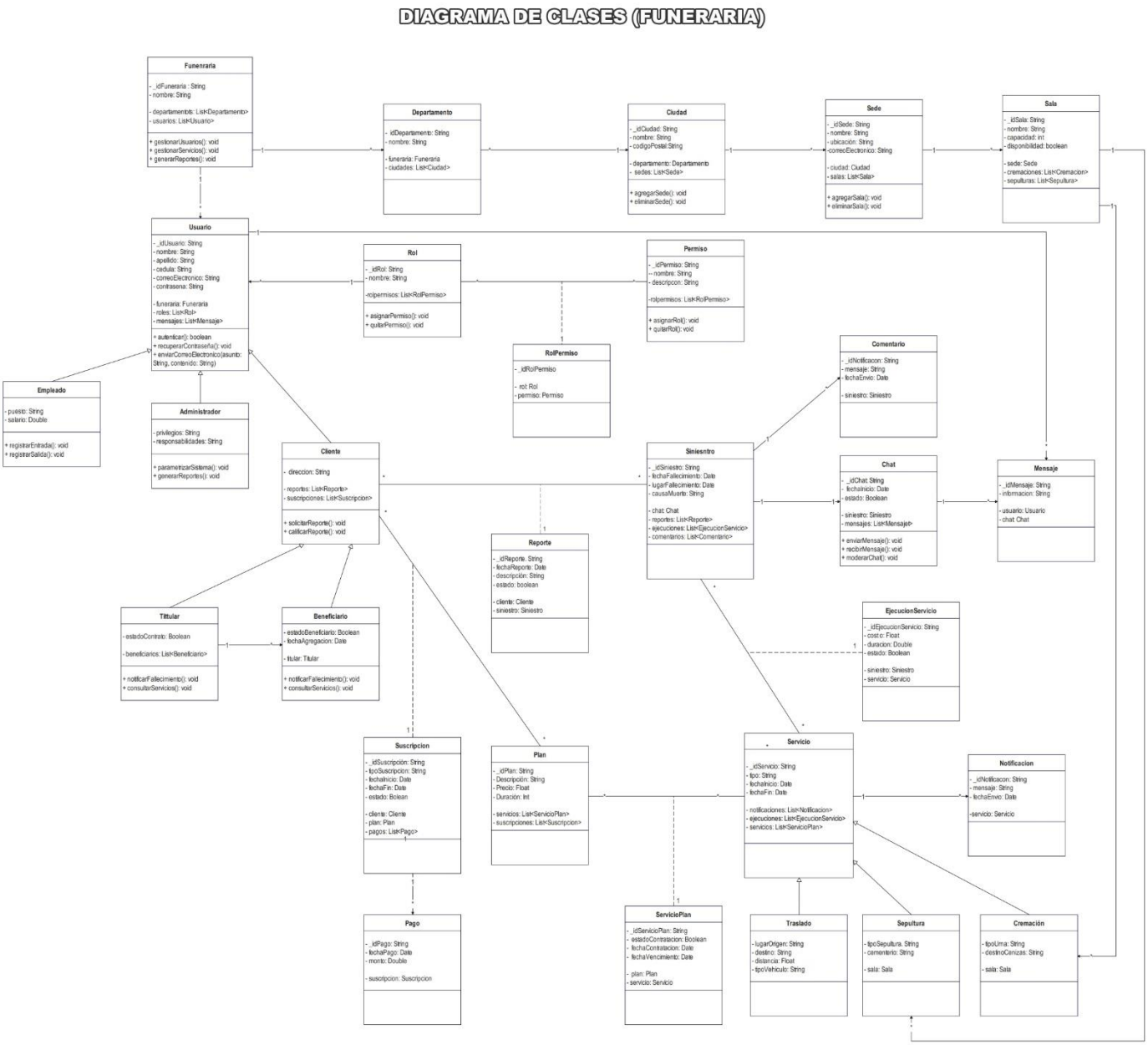
Propósito: Almacenar información sobre las salas de velación disponibles.

- **id:** Identificador único de cada sala de velación.
- **Name:** Nombre de la sala de velación.
- **Capacity:** Capacidad de la sala de velación.
- **Availability:** Disponibilidad de la sala de velación.
- **Site_id:** Referencia a la tabla sedes, identificando a que sede le pertenece esa sala de velación.

wake_rooms

Column name	DataType	PK	FK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓		✓			✓		✓		
name	VARCHAR(255)									NULL	
capacity	INT									NULL	
availability	TINYINT(1)									NULL	
site_id	INT		✓				✓			NULL	
created_at	TIMESTAMP									NULL	
updated_at	TIMESTAMP									NULL	

Diagrama de Clases



El diagrama de clases presentado ilustra la estructura de una aplicación de gestión de servicios funerarios. A continuación, se describe en detalle cada una de las entidades y sus relaciones, reflejadas en el diagrama:

Entidades Principales

- **Cliente**: Esta entidad representa a los usuarios que contratan los servicios funerarios. Tiene relaciones con varias otras entidades, lo que permite una gestión integral de sus solicitudes y servicios.

- **Beneficiario**: Conectada a la entidad Cliente, esta clase representa a las personas que se benefician de los servicios contratados. Incluye atributos que permiten identificar y gestionar a los beneficiarios de manera efectiva.
- **Suscripción**: Gestiona las suscripciones a los diferentes planes de servicios funerarios. Incluye detalles como el tipo de plan, la fecha de inicio, y otros atributos relevantes para definir el alcance y la duración del servicio contratado.

Administración de la Funeraria

- **Empleado**: Representa a los empleados de la funeraria, gestionando sus roles y responsabilidades dentro de la organización.
- **Departamento**: Agrupa a los empleados en diferentes áreas de trabajo, facilitando la organización interna y la asignación de tareas específicas.
- **Ciudad** y **Sede**: Estas clases gestionan las ubicaciones geográficas donde se ofrecen los servicios funerarios. La clase Ciudad define el contexto geográfico amplio, mientras que Sede especifica las ubicaciones particulares de las oficinas o instalaciones de la funeraria.

Servicios y Transacciones

- **Servicio**: Detalla los diferentes servicios ofrecidos por la funeraria, como cremaciones, entierros, arreglos florales, entre otros. Cada servicio tiene atributos específicos que describen su naturaleza y características.
- **Pago**: Maneja toda la información relativa a las transacciones financieras realizadas por los clientes. Esta entidad asegura que todos los pagos sean registrados y gestionados de manera eficiente y segura.

Comunicación y Notificaciones

- **Notificación**: Gestiona las alertas y recordatorios enviados a los usuarios. Esta clase es crucial para mantener a los clientes informados sobre el estado de sus servicios y cualquier actualización importante.

Otras Entidades Importantes

- **Administrador**: Encargado de gestionar las operaciones del sistema, tiene la capacidad de registrar y administrar los detalles de los clientes y otros aspectos de la funeraria.
- **Titular**: Representa al titular de una suscripción o servicio, gestionando sus detalles específicos.
- **Rol y Permiso**: Estas entidades gestionan los roles y permisos de los usuarios del sistema, asegurando que cada usuario tenga acceso adecuado según su función.
- **ServicioPlan**: Relaciona los diferentes planes con los servicios ofrecidos, detallando qué servicios están incluidos en cada plan.

- **Transferencia** y **EjecuciónServicio**: Gestionan los aspectos operativos de la transferencia de restos y la ejecución de los servicios funerarios.
- **Sepultura** y **Cremación**: Detallan los servicios específicos de sepultura y cremación, gestionando sus características y procesos específicos.
- **Reporte**: Genera informes sobre las diferentes actividades y servicios de la funeraria, proporcionando datos para la toma de decisiones.
- **Chat** y **Mensaje**: Facilitan la comunicación interna y externa, permitiendo el envío de mensajes y la coordinación entre los diferentes actores del sistema.

Conexiones e Interacciones

Las entidades descritas están interconectadas para asegurar una gestión coherente y fluida de todos los aspectos relacionados con los servicios funerarios:

- **Relaciones del Cliente**: El Cliente se relaciona directamente con el Beneficiario y la Suscripción, garantizando que los servicios contratados estén alineados con las necesidades y expectativas de los beneficiarios.
- **Administración Integrada**: Empleado, Departamento, Ciudad y Sede trabajan en conjunto para gestionar la logística y la administración interna de la funeraria, asegurando que los servicios sean ofrecidos de manera organizada y eficiente.
- **Gestión de Servicios y Pagos**: Las clases Servicio y Pago interactúan para ofrecer un registro detallado de los servicios prestados y los pagos recibidos, asegurando la transparencia y eficiencia en las transacciones.
- **Notificaciones Efectivas**: La clase Notificación se integra con otras entidades para enviar comunicaciones relevantes a los usuarios, manteniéndolos informados y comprometidos con el servicio.

Información Técnica de los Frameworks Utilizados

Java Spring Boot

- **Versión**: 3.2.3
- **Funcionalidad**: Autenticación y autorización de usuarios.
- **Ejecución**: Ejecutar la clase MsSecurityApplication como una aplicación Spring Boot

Adonis JS

- **Versión**: Node 21.2.0
- **Funcionalidad**: Gestión de operaciones centrales de la funeraria.
- **Ejecución**: node ace serve

Laravel

- **Versión**: 8.x
- **Funcionalidad**: Procesamiento de transacciones.
- **Ejecución**: php artisan serve

Flask Python

- **Versión**: 2.0.1
- **Funcionalidad**: Envío de notificaciones en tiempo real.
- **Ejecución**: python app.py

Angular

- **Versión**: 17
- **Funcionalidad**: Interfaz de usuario.
- **Ejecución**: ng serve