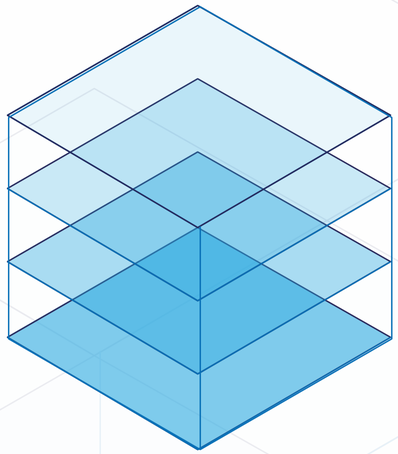


MISO

Maestría en Ingeniería de Software

Diseño de experimento – Multitenant

Proyecto Final 1

Titulo del experimento	**Separación de información por inquilino (tenant)**
Propósito del experimento	El objetivo del experimento es validar que el sistema garantiza que las operaciones e información de cada cliente (tenant) están completamente aisladas, asegurando que ningún cliente puede acceder o afectar los datos de otro cliente (tenant), independientemente del volumen de usuarios concurrentes.
Resultados esperados	Se espera que el 100% de las operaciones de cada cliente (tenant) se limitan exclusivamente a su propio conjunto de datos. No se observa ningún cruce de información entre tenants bajo cualquier tipo de carga o escenario de uso.
Recursos requeridos	**Software:** NestJS para la implementación del backend, herramientas de pruebas como Postman para simular acciones de usuario, y una base de datos PostgreSQL para almacenamiento de registros. **Librerías:** Librerías para soporte de comunicación con gRPC. Driver de Postgresql, TypeORM. **Hardware:** Instancias de CloudRun en la nube para desplegar el sistema y simular diferentes condiciones de carga, Instancia de Cloud SQL para PostgreSQL.
Elementos de arquitectura involucrados	Componentes **API Gateway, Mediator, incident process y customer tenant management**. Ubicados en la vista de desarrollo en los modelos de Gateway, Integration, Business Logic y commons.
Esfuerzo estimado	Aproximamente 20 Horas/hombre distribuidas entre 4 ingenieros de desarrollo.

Hipótesis de diseño	
Punto de sensibilidad	Este experimento se enfoca en la confidencialidad del sistema, garantizando que la información de cada cliente de ABCall se encuentre aislada, de esta forma se puede garantizar que la informacion de un cliente no se puede acceder ni relacionar por otro cliente.
Historia de arquitectura asociada	Como **cliente de ABCall**, cuando **un usuario final registra un Incidente**, dado que **el sistema es utilizado simultáneamente por múltiples tenants (clientes)**, quiero que **mis operaciones estén completamente aisladas y solo afecten a mis propios datos** para **garantizar que no accedo ni modifico información de otros tenants**. Esto debe suceder **sin errores en el 100% de los casos y validarse mediante auditorías y pruebas regulares**.
Nivel de incertidumbre	**Medio:** Aunque se esta planteando un diseño inicial de un patrón multitenant para el sistema de **ABCall**, es necesario garantizar el aislamiento de la información por cliente, ya que es fundamental que se garantice a los clientes la confidencialidad de su información.

Estilos de Arquitectura asociados al experimento	Análisis (Atributos de calidad que favorece y desfavorece)
Microservicios	**Favorece:** • **Confidencialidad:** Cada microservicio accedera unicamente a la informacion del cliente (tenant) especificado en cada petición. • **Consistencia de Datos:** Se favorece la consistencia de los datos debido a que se aislara la informacion de cada cliente (tenant) en un espacio dedicado para dicho cliente, esto facilita que la consistencia de los datos se asegure. • **Modificabilidad:** Se favorece la modificabilidad debido a que el sistema sera homogeneo para todos los clientes (tenant) cualquier cambio en el sistema impactara directamente a todos los clientes (tenants) que utilicen el sistema. **Desfavorece:** • **Mantenibilidad:** Desfavorece la mantenibilidad, debido a que al implementar un patron de multitenant se vuelve mas complejo el manejo de la infraestructura donde se almacena la información de cada cliente.

Tácticas de Arquitectura asociadas al experimento	Descripción
Seguridad: Separate Entities	La táctica "Separate Entities" es fundamental para la seguridad en un sistema multitenant, ya que proporciona mecanismos para aislar a los tenants y proteger los datos.

Listado de componentes (Microservicios) involucrados en el experimento		
Microservicio	Propósito y comportamiento esperado	Tecnología Asociada
API Gateway	Orquestar y balancear la carga de todas las peticiones que reciba el sistema de ABCall. Debe redirigir la petición hacia el mediator.	NestJS, REST, gRPC,
Mediator	Se encarga de validar, orquestar y redirigir las peticiones hacia las capacidades expuestas en la capa de business logic del sistema de ABCall. Debe recibir la petición y redirigir la petición hacia el Incident Process Service.	NestJS, gRPC
Incident Process	Es el responsable de toda la gestión relacionada a los incidentes del sistema, desde su registro hasta la resolución del mismo. Debe recibir la petición y registrar correctamente el incidente en la base de datos.	NestJS, gRPC, PostgreSQL
Customer tenant management	Es el responsable de proveer la configuracion especifica de cada tenant para todas las interacciones con el sistema	NestJS, gRPC, PostgreSQL, Cache

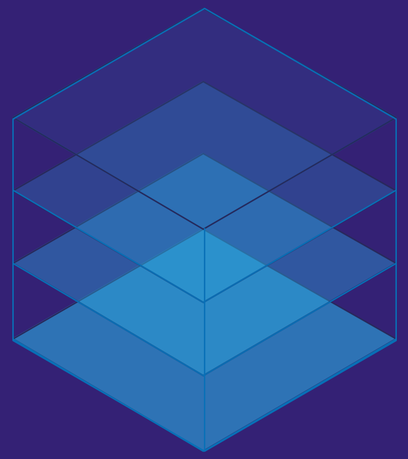
Listado de conectores involucrados en el experimento

Conector	Comportamiento deseado en el experimento	Tecnología Asociada
HTTP/Rest in API Gateway	Todas las peticiones del experimento tendrán como punto de entrada será el API Gateway, estas peticiones serán redirigidas hacia el mediator indicado.	NestJs, REST, gRPC
gRPC in Mediator	Gestionara las peticiones recibidas por gRPC desde el API Gateway y las redirigirá hacia el Incident Process Service.	NestJs, gRPC
gRPC in Incident Process	Se encarga de registrar los incidentes en la base de datos del sistema.	NestJs, gRPC, PostgreSQL
gRPC in Customer tenant management	Se encarga de proveer las configuraciones por tenant a cada microservicio del sistema	NestJs, gRPC, PostgreSQL, Cache

Tecnología asociada con el experimento (Desarrollo, infraestructura, almacenamiento)	Justificación
Lenguajes de programación	**TypeScript**: Utilizado con **NestJS** para el desarrollo del backend, proporciona un tipado estático que mejora la robustez y mantenibilidad del código.
Plataforma de despliegue	**Google Cloud Run**: Permite desplegar contenedores en la nube con escalabilidad automática y gestión simplificada, ideal para microservicios desarrollados en NestJS.
Bases de datos	**PostgreSQL**: Elegido por su robustez, soporte de transacciones complejas, y capacidad para manejar un gran volumen de registros de auditoría de manera eficiente.
Herramientas de análisis	Jmeter, Apache AB Benchmark.
Librerías	**gRPC:** Se utilizara las librerías necesarias para poder utilizar Protocol Buffers con gRPC dentro de NestJS.
Frameworks de desarrollo	**NestJS**: Un framework progresivo de Node.js que facilita la creación de aplicaciones escalables y mantenibles mediante el uso de patrones de diseño robustos y buenas prácticas.

Distribución de actividades por integrante

Integrante	Tareas a realizar	Esfuerzo Estimado
Jesús Henríquez	- Desarrollo del API Gateway en NestJ y su implementación HTTP/Rest y con gRPC. - Desarrollo del Mediator en NestJs y su implementación HTTP/Rest y con gRPC. - Desarrollo del Incident Process en NestJS y su implementación HTP/Rest y con gRPC. - Desarrollo del Customer tenant management en NestJS y su implementación con gRPC y Cache.	15h
Daniel Jimenez	- Configuración de la infraestructura en **Google Cloud Run** para despliegue de microservicios. - Configuración de **Google Cloud Pub/Sub** para comunicación asíncrona. - Configuración de bases de datos en **PostgreSQL** para el almancenamiento de los Incidentes.	5h
Carlos Castillo	Realizar pruebas de carga para evaluar los tiempos de respuestas de gRPC y HTTP/Rest con Jmeter.	1h
Santiago Begambre	Realizar pruebas de carga para evaluar los tiempos de respuestas de gRPC y HTTP/Rest con Apache AB Benchmark.	1h



MISO

Maestría en Ingeniería de Software