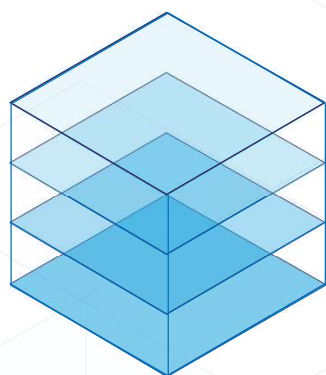




MISO
Maestría en Ingeniería de Software

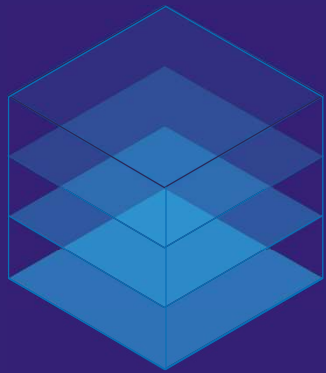


MISO

Maestría en Ingeniería de Software

Experimentación

SaludTech – Alpes
ByteBros



MISO

Maestría en Ingeniería de Software

Experimento 3 - Modificabilidad Modificaciones y extensibilidad en el proceso de suscripción

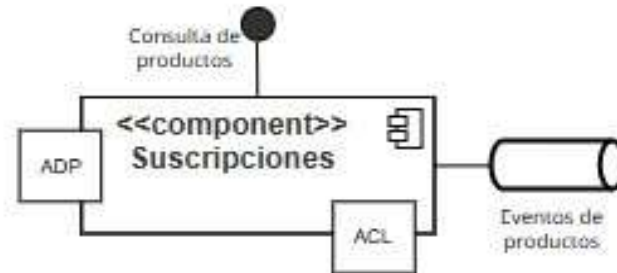
Atributo de calidad 3: Modificabilidad

Escenario de calidad: Modificaciones y extensibilidad en el proceso de suscripción			
Escenario #: 8	El sistema debe permitir modificaciones y extensibilidad en el proceso de suscripción, asegurando bajo acoplamiento y facilitando la integración de nuevos productos y cambios sin afectar la arquitectura existente.		
Fuente	Nuevos suscriptores (consumidores)		
Estímulo	Requerimiento de modificar o extender la lógica del proceso de suscripción para incluir nuevos productos o mejorar funcionalidades sin afectar componentes existentes.		
Ambiente	Desarrollo con evolución y mejoras continuas.		
Artefacto	Microservicios de suscripción		
Respuesta	Facilitar la evolución del sistema mediante patrones de reducir acoplamiento, utilizando adaptadores, mediadores y eventos.		
Medida de la respuesta	La integración de un nuevo producto en el proceso de suscripciones generó 0% de cambios en los microservicios existentes, manteniendo la estabilidad del sistema y asegurando la extensibilidad de la arquitectura.		
Decisiones Arquitecturales	Punto de sensibilidad	Tradeoff	Riesgo
Tactica de reducir el acoplamiento con el uso de adaptadores, mediadores y eventos en la lógica del microservicio de suscripción.	Reducción del acoplamiento y facilitar la evolución del sistema.	Latencia por la comunicación asíncrona, menor control en los flujos de negocio, impacto en las pruebas.	Complejidad en la gestión de eventos y mediadores, dificultad en la depuración y afectación en la integridad de los datos si no se hace una correcta gestión de las transacciones.
Justificación	Para permitir la evolución del sistema sin afectar los componentes existentes, es fundamental aplicar patrones de desacoplamiento. Respecto al uso de adaptadores facilita la conexión con nuevas fuentes de datos sin modificar el microservicio de suscripción, además los mediadores permiten gestionar la lógica de negocio de manera centralizada sin generar dependencias directas entre servicios y los eventos puedan ser manejados de manera extensible sin afectar la lógica interna del microservicio.		

Atributo de calidad: Seguridad

Integridad y seguridad de datos

Diagrama de arquitectura



Hipótesis de diseño asociada al experimento

Punto de sensibilidad	La gestión de eventos y mediadores es el punto crítico en el que podrían surgir problemas. Si los eventos no se gestionan correctamente, podría haber desincronización entre los servicios, lo que podría generar inconsistencias en el flujo de información. Además, el uso de mediadores aumenta la complejidad de la gestión centralizada de la lógica de negocio y puede dificultar la depuración si no se implementa de manera adecuada.
Historia de arquitectura asociada	Cada microservicio fue diseñado para ser autónomo y desacoplado, permitiendo la incorporación de nuevos productos y funcionalidades sin necesidad de modificar la lógica central del sistema. En este proceso de evolución, se implementaron patrones como adaptadores , mediadores y eventos para reducir aún más el acoplamiento entre los microservicios y facilitar la evolución continua del sistema.
Nivel de incertidumbre	<p>El nivel de incertidumbre se considera moderado a alto debido a la complejidad añadida por la implementación de eventos y mediadores. Estos patrones pueden generar problemas de desincronización entre los microservicios si no se gestionan correctamente, afectando la coherencia de los datos y la integridad de las transacciones. Además, el uso de comunicación asíncrona mediante eventos podría causar latencia, impactando el rendimiento y la capacidad de respuesta del sistema si la gestión de los eventos no es eficiente.</p> <p>Otro factor de incertidumbre es la complejidad en la depuración y las pruebas, ya que la lógica centralizada y la asincronía dificultan el monitoreo y la identificación de errores en el sistema. También, aunque la arquitectura es extensible, el creciente número de eventos generados por nuevos productos o funcionalidades podría sobrecargar la infraestructura si no se controla adecuadamente la escalabilidad de los microservicios, afectando su capacidad de procesamiento.</p>

Análisis de los resultados obtenidos

- 1- Indique si la hipótesis de diseño pudo ser confirmada o no
- 2- En caso de que la hipótesis se haya confirmado, explique las decisiones de arquitectura que favorecieron el resultado
- 3- En caso de que los resultados del experimento no hayan sido favorables, explique por qué y cuáles cambios realizaría en el diseño

Sí, la hipótesis de diseño fue confirmada. El sistema demostró ser flexible y extensible, permitiendo la integración de nuevos productos sin afectar los microservicios existentes. El uso de **adaptadores, mediadores y eventos** para reducir el acoplamiento entre los servicios facilitó la incorporación de nuevas funcionalidades, manteniendo la estabilidad y escalabilidad de la arquitectura sin causar interrupciones en el servicio.

Las decisiones clave que favorecieron la confirmación de la hipótesis incluyen el uso de **microservicios modulares**, que permitieron la integración de nuevos productos sin modificar los servicios existentes, reduciendo el riesgo de errores. Además, la **implementación de adaptadores** facilitó la integración de nuevas fuentes de datos sin afectar la arquitectura principal. La **centralización de la lógica con mediadores** mejoró la flexibilidad del sistema, gestionando las interacciones sin generar dependencias directas. Finalmente, el **uso de eventos para comunicación asíncrona** redujo el acoplamiento entre microservicios, permitiendo una integración más flexible de nuevos productos sin generar dependencias rígidas.

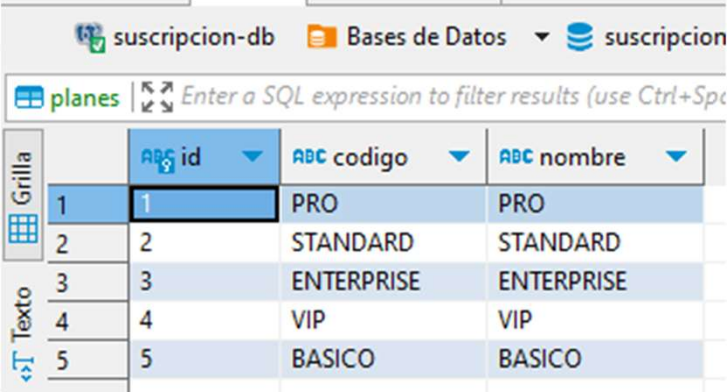
Se obtuvieron los siguientes resultados:

Evidencias

1- Presente evidencias de los resultados obtenidos en el experimento.

Para realizar las pruebas en el microservicio de suscripción se validaron varios planes mas de los que se encontraban diseñados, para agregarlos fue necesario agregar a nivel de base de datos la información de los planes que quedarían aprobados, lo cuál no requirio ningún cambio en el microservicio. Además, por el manejo de eventos gordos en una transacción larga entre los microservicios asociados se comparte la información del código del plan. Como se muestra en el llamado al BFF para la ejecución de la Saga.

```
{
  "id_suscripcion": "{{${randomUUID}}}",
  "cliente_codigo": "0001",
  "cliente_nombres": "Pablo Saga",
  "cliente_apellidos": "Perez Prieto",
  "cliente_usuario": "pperez",
  "cliente_rut": "1234567890",
  "cliente_cedula": "987654321",
  "cliente_email": "pperez@domain.com",
  "plan_codigo": "pro",
  "plan_nombre": "PRO"
}
```



	id	codigo	nombre
1	1	PRO	PRO
2	2	STANDARD	STANDARD
3	3	ENTERPRISE	ENTERPRISE
4	4	VIP	VIP
5	5	BASICO	BASICO

Evidencias

1- Presente evidencias de los resultados obtenidos en el experimento.

Además dentro del código de microservicio suscripciones vemos la entidad Plan, la cual es usada de la misma forma en el microservicio de servicio de datos. En cada microservicio se almacena la información correspondiente con el fin de no tener dependencias.

```
entidades.py X
microservicio-suscripciones > src > saludtecalpes > modulos > suscripciones > dominio > entidades.py > ...
19 email: Email = field(default_factory=Email)
20
21 @dataclass
22 class Plan(Entidad):
23     codigo:Codigo = field(default_factory=Codigo)
24     nombre:NombrePlan = field(default_factory=NombrePlan)
25
```

```
entidades.py X
microservicio-servicio-datos > src > saludtecalpes > modulos > servicioDatos > dominio > entidades.py > ...
17
18 @dataclass
19 class Plan(Entidad):
20     codigo:Codigo = field(default_factory=Codigo)
21     nombre:NombrePlan = field(default_factory=NombrePlan)
22
```


Evidencias

1- Presente evidencias de los resultados obtenidos en el experimento.

Además, gracias a la separación entre las capas del microservicio de Suscripciones y al uso de adaptadores, mediadores y eventos, es posible adaptarlo fácilmente a los cambios del negocio sin necesidad de modificaciones en el código.

```
└─ microservicio-suscripciones
  ├── pgdata
  ├── sql
  └── src\saludtechalpes
      ├── api
      ├── config
      ├── modulos
      └── suscripciones
          ├── aplicacion
          ├── dominio
          ├── infraestructura
          └── init .py
```

```
└─ suscripciones
  └── aplicacion
      ├── comandos
      ├── queries
      ├── __init__.py
      ├── dto.py
      ├── handlers.py
      ├── mapeadores.py
      └── servicios.py
```

```
└─ dominio
  ├── __init__.py
  ├── entidades.py
  ├── eventos.py
  ├── excepciones.py
  ├── fabricas.py
  ├── objetos_valor.py
  ├── reglas.py
  └── repositorios.py
```

```
└─ infraestructura
  ├── schema
  ├── consumidores.py
  ├── despachadores.py
  ├── dto.py
  ├── excepciones.py
  ├── fabricas.py
  ├── mapeadores.py
  └── repositorios.py
```