

Tema 2

Domain-Driven Design

Ejemplo - Suscripciones



Introducción – Motivación

La Universidad de Murcia (UMU) quiere implantar un servicio digital de **suscripciones**, pensado para estudiantes que utilizan de forma intensiva distintos servicios digitales (música, vídeo, deporte, bibliotecas online, etc.).

La idea es ofrecer una **plataforma sencilla y flexible** que permita a los estudiantes **contratar, modificar y cancelar suscripciones** de forma cómoda, con **planes especiales y ventajas para la comunidad universitaria**.

NOTA:

Este ejemplo es deliberadamente incompleto y simplificado para ilustrar decisiones de diseño, no una arquitectura final.



Situaciones habituales del dominio

En esta plataforma se dan situaciones como:

- Estudiantes que **contratan un plan de suscripción**.
- Estudiantes que **cancelan su suscripción**, pero siguen teniendo acceso hasta final de mes.
- Estudiantes que **cambian de plan, pero el cambio entra en vigor en el siguiente ciclo**.
- Planes que permiten **pausar temporalmente la suscripción**.
- Los estudiantes **no pueden tener más de una suscripción activa** al mismo tiempo.
- Estudiantes que intentan **contratar un nuevo plan** teniendo ya uno activo.

Historias de usuario

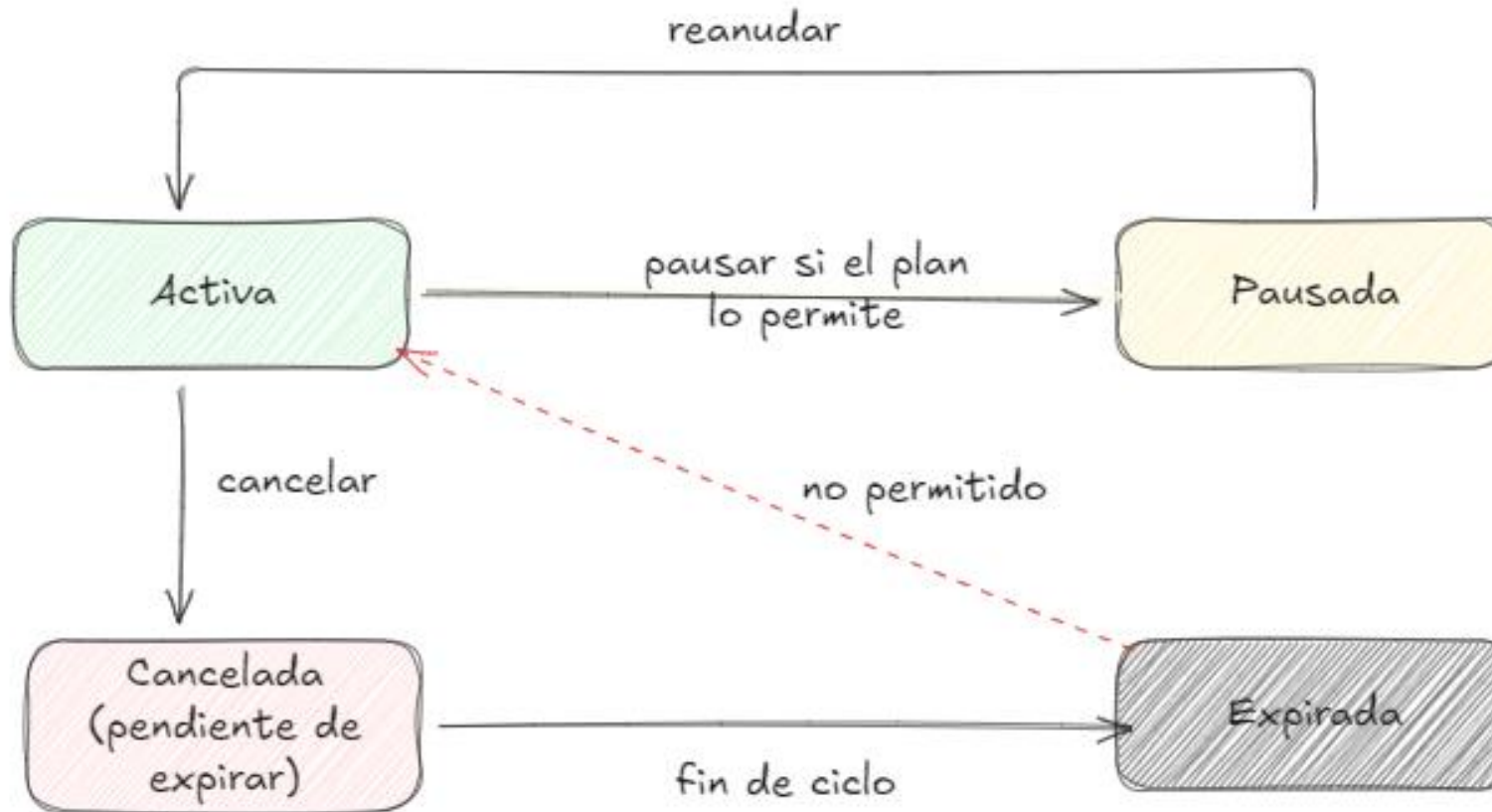
- Como **alumno**, quiero **contratar una suscripción** eligiendo un plan, para tener acceso a los servicios asociados a ese plan.
- Como **alumno**, quiero **cambiar el plan de mi suscripción**, para adaptar el servicio a mis necesidades.
- Como **alumno**, quiero **pausar temporalmente mi suscripción**, para no consumir el servicio durante un periodo de tiempo.
- Como **alumno**, quiero **cancelar mi suscripción**, para dejar de renovarla automáticamente, manteniendo el acceso hasta final de ciclo.
- Como **alumno**, quiero **consultar el estado y el historial de mi suscripción**, para saber en qué situación se encuentra y qué cambios se han realizado.

Modelo del dominio



- ¿Dónde se garantiza que un estudiante no tenga **dos suscripciones activas**?
- ¿Quién decide si una **transición de estado es válida**?
- ¿Qué ocurre si dos historias de usuario modifican suscripciones a la vez?

Diagrama de estados



¿Quién garantiza que estas transacciones se respeten en el sistema?

Modelo del dominio



- No hay nada que impida tener dos suscripciones activas.
- El estado es solo un dato: cualquiera podría cambiarlo.
- Las reglas de negocio no tienen un lugar claro.

Agregados

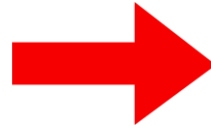


- No estamos añadiendo nada nuevo al dominio.
Estamos decidiendo **quién es responsable de que no se rompa**.
 - Todas las reglas que afectan a una suscripción deben pasar por ella.
 - Nadie cambia el estado “desde fuera”.
 - La suscripción protege sus propias invariantes.

Agregados

Opción A

```
suscripcion.setEstado (EXPIRADA);  
suscripcion.setPlan (nuevoPlan);
```



Opción B

```
suscripcion.cancelar (fecha);  
suscripcion.pausar (plan, fecha);  
suscripcion.cambiarPlan (  
    nuevoPlan,  
    fecha  
);
```

El agregado expone *operaciones del dominio*, no *setters* ni *getters*.

¿Qué hemos resuelto hasta ahora?

Coherencia de una suscripción individual

- Una suscripción controla sus **estados y transiciones**.
- No se pueden realizar **cambios inválidos de estado**.
- *Cancelar mantiene el acceso hasta fin de ciclo.*
- *Cambiar de plan entra en vigor en el siguiente ciclo.*
- *Pausar solo es posible si el plan lo permite.*

¿Qué NO hemos resuelto hasta ahora?

Coherencia entre varias suscripciones

- *Un estudiante no puede tener más de una suscripción activa.*

¿Por qué no encaja en la Suscripción?

- No es responsabilidad de **una única suscripción**.
- Requiere conocer el estado de **otras suscripciones** del estudiante.
- Forzarla dentro del agregado rompería su responsabilidad.

Servicio de dominio

Un servicio de dominio encapsula **reglas del negocio** que no pertenecen claramente a una sola entidad.

- El **servicio** decide *si* algo se puede hacer.
- El **agregado** decide *cómo* se hace.

Servicios de dominio

```
public class ServicioSuscripciones {  
  
    public Suscripcion contratar (EstudianteId estudianteId, PlanId planId) {  
  
        // 1) Consulto el estado global (varias suscripciones)  
        boolean yaTieneActiva = repo.existeActivaPara (estudianteId);  
  
        if (yaTieneActiva) {  
            throw new ReglaDeNegocio ("Ya existe una suscripción activa");  
        }  
  
        // 2) Creo el agregado y aplico reglas internas  
        Suscripcion suscripcion = Suscripcion.crearNueva (estudianteId, planId);  
  
        return suscripcion;  
    }  
}
```

Ideas

- El servicio no toca estado por *setters*: delega en el agregado.
- El repositorio aparece para mirar.

Servicios de dominio

```
public class ServicioSuscripciones {  
  
    public void cambiarPlan (Estudiante estudianteId, Plan nuevoPlanId) {  
  
        Suscripcion activa = repo.obtenerActivaDe (estudianteId);  
  
        if (activa == null) {  
            throw new ReglaDeNegocio ("No existe una suscripción activa");  
        }  
  
        activa.solicitarCambioDePlan (nuevoPlanId);  
    }  
}
```

Ideas

- El **servicio** busca la suscripción activa y **decide** si la operación tiene sentido.
- La **suscripción** valida su estado y aplica las reglas.
- **NOTA**
Recordad que cambiar de plan no es inmediato (siguiente ciclo).

¿Qué hemos visto?

En este ejemplo hemos visto como usar algunos conceptos de DDD para fijar historias de usuario, decisiones del modelo, agregados y servicios de dominio.

- Dominio → situaciones → historias
- Modelo → problemas → decisiones
- Agregado → reglas internas
- Servicio → reglas transversales

