Programa de Ingeniería de Sistemas y Computación
Universidad del Quindío

Curso: Programación Avanzada
Guía: 04
Título: Modelado del Dominio I – Entidades y Value Objects
Duración estimada: 120 minutos
Docente: Christian Andrés Candela

MODELADO DEL DOMINIO I: FUNDAMENTOS DE DDD

OBJETIVO

En las guías anteriores entendimos:

- qué problema vamos a resolver,
- por qué el dominio es el centro del sistema,
- y cómo preparar un proyecto base para crecer correctamente.

En esta guía comenzamos a **modelar el dominio**.

El objetivo **no es aprender teoría por memorizar**, sino aprender a:

- identificar los conceptos importantes del problema,
- diferenciarlos correctamente,
- y expresarlos de forma clara en el código.

> En esta guía el código **representa ideas**, no infraestructura.

CONTEXTUALIZACIÓN TEÓRICA

¿Qué es el dominio?

El **dominio** es el área de conocimiento o actividades en el cual esta inmerso el problema real que queremos resolver con software. Incluye:

- las reglas que deben cumplirse
- los conceptos importantes del problema
- las decisiones que el sistema puede o no puede tomar

En este curso, el dominio es la **gestión de solicitudes académicas**.

¿Cómo identificar los elementos importantes del problema?

Una forma sencilla de empezar es preguntarse:

- ¿Qué cosas existen en este problema?
- ¿Qué cosas cambian con el tiempo?
- ¿Qué cosas tienen reglas?

Las respuestas a estas preguntas nos llevan a identificar los **conceptos clave del dominio**.

Entidades

Una **entidad** es un objeto del dominio que:

- tiene identidad propia
- puede cambiar su estado
- se distingue de otros objetos similares

Ejemplo: una *Solicitud* es una entidad, porque cada solicitud es distinta, aunque tenga los mismos datos que otra.

Value Objects

Un **Value Object** es un objeto que:

- no tiene identidad propia
- se define únicamente por sus valores
- suele ser inmutable

Ejemplos: prioridad, tipo de solicitud, clasificación.

Entidad vs Value Object

Aspecto	Entidad	Value Object
Identidad	Sí	No
Mutabilidad	Puede cambiar	Inmutable
Igualdad	Por identidad	Por valores
Ejemplo	Solicitud, Usuario	Prioridad, Email

Lenguaje del dominio (lenguaje compartido)

El **lenguaje de dominio** es el conjunto de palabras y expresiones que usan las personas involucradas en el problema para hablar de él y entenderse entre sí.

Incluye:

- conceptos clave del problema,
- acciones que se realizan,
- estados por los que pasan las cosas,
- y reglas implícitas del negocio.

Este lenguaje debe ser **compartido** entre:

- estudiantes,
- docentes,
- usuarios del sistema,
- y el código fuente.

El objetivo es que todos hablen **el mismo idioma**, sin traducciones mentales entre lo que se dice y lo que se programa.

Referencia: Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.

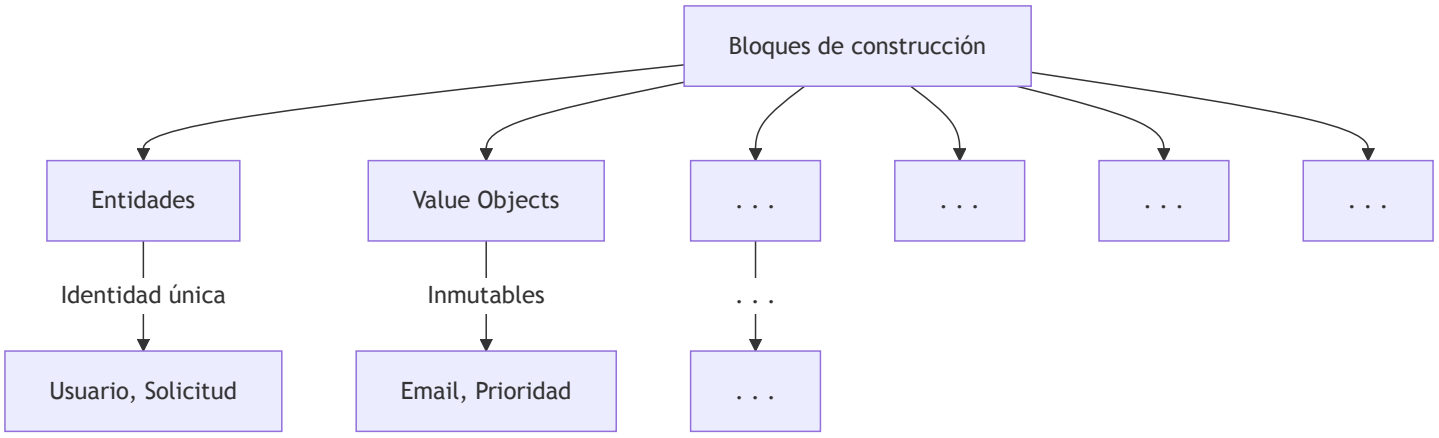
Modelo de Dominio

Representación abstracta de los conceptos, reglas y relaciones del dominio de negocio.

Beneficios

1. **Comunicación mejorada** entre desarrolladores y expertos del dominio
2. **Código más expresivo** que refleja el lenguaje del negocio
3. **Facilita el mantenimiento** al concentrar la lógica de negocio
4. **Reduce la brecha** entre análisis y código
5. **Código autodocumentado** mediante el lenguaje de dominio

Bloques de construcción



En esta guía nos enfocaremos en: Entidades y Value Objects

PARTE 1: IDENTIFICANDO CONCEPTOS DEL DOMINIO

Contexto del Dominio

Lee el siguiente escenario:

> Un estudiante registra una solicitud de homologación. Un coordinador clasifica la solicitud y le asigna prioridad MEDIA. Luego la asigna a un docente para su atención.

Ejercicio guiado

Identifica qué conceptos son **Entidades** y cuáles son **Value Objects**:

Término	Tipo	Definición
Solicitud		
Usuario		
Prioridad		
Estado de Solicitud		
Tipo de Solicitud		

Discútelo en grupo antes de responder.

> El resultado del análisis anterior es nuestro primer modelo de dominio.

Aplicación al proyecto final

Extienda este ejercicio al proyecto final identificando los conceptos del dominio y sus relaciones.

PARTE 2: REGLAS BÁSICAS EN EL DOMINIO

El dominio no es solo datos. También contiene **reglas de negocio** que definen qué se puede y qué no se puede hacer.

Ejemplos de reglas en este dominio:

- una solicitud cerrada no puede modificarse
- no se puede atender una solicitud sin responsable
- una solicitud debe tener un tipo válido

Estas reglas **deben vivir en el dominio**, no en controladores ni servicios técnicos.

Ejercicio: identificando reglas de negocio

Lee nuevamente el siguiente escenario:

> Un estudiante registra una solicitud de homologación. Un coordinador clasifica la solicitud y le asigna prioridad MEDIA. Luego la asigna a un docente para su atención.

En grupo, identifica **al menos 3 reglas de negocio** que se desprendan de este escenario.

Para cada regla, responde:

- ¿qué acción regula?
- ¿qué condición debe cumplirse?
- ¿qué pasaría si la regla no existiera?

> No escribas código todavía. El objetivo es **pensar como el dominio**, no codificar.

Aplicación al proyecto final

Extienda este ejercicio al proyecto final identificando las reglas de negocio del dominio.

PARTE 3: MATERIALIZACIÓN DEL DOMINIO EN JAVA

Hasta ahora has identificado **conceptos**, **entidades**, **value objects** y **reglas de negocio**. En esta actividad darás el siguiente paso: **materializar esos conceptos en código Java**, sin convertirlos todavía en una aplicación.

Debemos representar el dominio en código de forma clara, expresiva y coherente, usando únicamente Java puro.

> El objetivo no es que el sistema funcione, sino que el código **explique el dominio**.

Estructura base del proyecto

A partir del proyecto Spring Boot creado en la guía anterior, trabajarás únicamente en el siguiente paquete:

```
com.unquindio.solicitudes.domain
```

Este paquete será el **núcleo del sistema** y se mantendrá durante todo el semestre.

Dentro de `domain` puedes crear subpaquetes como:

- `entity`
- `valueobject`
- `exception` (opcional)

Requisitos de la actividad

Debes crear:

- **Al menos 2 Entidades** del dominio
- **Al menos 3 Value Objects**

Cada artefacto debe:

- usar nombres del lenguaje del dominio
- reflejar un concepto real del problema

Aplicación al proyecto final

Extienda este ejercicio al proyecto final materializando las entidades y value objects del dominio.

PRECAUCIONES Y RECOMENDACIONES

1. **No sobre-ingenierizar:** No convertir todo en Value Object. Usa criterio.
2. **Lenguaje ubicuo:** Usa los términos del dominio en clases, métodos y variables.
3. **Validar en constructores:** Los Value Objects deben validar en su creación.
4. **Inmutabilidad:** Prefiere `record` (Java 14+) para Value Objects inmutables.
5. **Tests:** Los conceptos del dominio deben tener tests unitarios.

EVALUACIÓN O RESULTADO

Al finalizar esta guía, los estudiantes deben:

1. ✅ Comprender los conceptos de Entidad y Value Object
2. ✅ Haber identificado al menos 3 entidades del Sistema de Triage
3. ✅ Haber creado al menos 2 Value Objects
4. ✅ Tener un modelo conceptual básico del dominio

PRÓXIMA ACTIVIDAD

En la **Guía 05 – Modelado del Dominio II** profundizaremos en:

- agregados
- reglas de consistencia
- servicios de dominio

> Se sugiere preparar un diagrama de clases conceptual del Sistema de Triage con las entidades y value objects identificados.

REFERENCIAS BIBLIOGRÁFICAS

1. Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
2. Vernon, V. (2013). *Implementing Domain-Driven Design*. Addison-Wesley.
3. Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
4. Nilsson, J. (2006). *Applying Domain-Driven Design and Patterns*. Addison-Wesley.
5. Martin Fowler - Anemic Domain Model: https://martinfowler.com/bliki/AnemicDomainModel.html
6. Spring Data JPA Documentation: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/