

Rúbrica de Evaluación - Sistema de Gestión Hospitalaria con JPA/Hibernate

Curso: Programación y Paradigmas Orientado a Objetos **Institución:** Universidad Tecnológica Nacional (UTN) - Argentina **Complejidad:** Alta (15-20 horas estimadas) **Puntaje Total:** 100 puntos

Objetivo del Proyecto

Desarrollar un **Sistema de Gestión Hospitalaria** utilizando **JPA/Hibernate** que demuestre el dominio de conceptos avanzados de persistencia, relaciones entre entidades, y patrones de diseño orientados a objetos. El sistema debe gestionar pacientes, médicos, departamentos, salas, historias clínicas y citas médicas.

Estructura del Proyecto Requerida

```
src/
├── main/
│   ├── java/org/example/
│   │   ├── entidades/      # 9 entidades JPA + 3 enums + 1 embeddable
│   │   │   ├── Persona.java # Clase abstracta con @MappedSuperclass
│   │   │   ├── Medico.java  # Hereda de Persona
│   │   │   ├── Paciente.java # Hereda de Persona
│   │   │   ├── Hospital.java # Aggregate Root
│   │   │   ├── Departamento.java
│   │   │   ├── Sala.java
│   │   │   ├── Cita.java
│   │   │   ├── HistoriaClinica.java
│   │   │   ├── Matricula.java # @Embeddable (Value Object)
│   │   │   ├── TipoSangre.java # enum
│   │   │   ├── EspecialidadMedica.java # enum
│   │   │   └── EstadoCita.java # enum
│   │   ├── servicio/      # Opcional: lógica de negocio
│   │   └── Main.java       # Demostración completa
│   └── resources/
```

```
|   └─ META-INF/  
|       └─ persistence.xml # Configuración JPA  
└─ build.gradle           # Gestión de dependencias
```

Base de datos: H2 file-based en ./data/hospidb.mv.db (se crea automáticamente)

Criterios de Evaluación (100 puntos)

1. Implementación de Entidades JPA del Dominio (25 puntos)

Debes implementar **correctamente** las siguientes entidades con sus anotaciones JPA:

1.1 Clase Abstracta Persona (4 pts)

- Usar `@MappedSuperclass` (patrón Template Method)
- Atributos protegidos: nombre, apellido, dni, fechaNacimiento, tipoSangre
- Métodos: `getNombreCompleto()`, `getEdad()`
- **Validación de DNI:** regex `\d{7,8}` en método `validarDni()`
- Usar `@SuperBuilder` de Lombok para herencia

1.2 Entidad Paciente (4 pts)

- Usar `@Entity` y heredar de `Persona`
- Atributos adicionales: telefono, direccion
- Relación `@OneToOne` con `HistoriaClinica` (creada automáticamente en constructor)
- Relación `@ManyToOne` con `Hospital`
- Relación `@OneToMany` con `List<Cita>`
- Constructor protegido que crea la historia clínica automáticamente
- Usar `@SuperBuilder`

1.3 Entidad Medico (4 pts)

- Usar `@Entity` y heredar de `Persona`
- Atributo `@Embedded` `Matricula` (validación formato MP-\d{4,6})
- Atributo especialidad (enum `EspecialidadMedica`)
- Relación `@ManyToOne` con `Departamento`
- Relación `@OneToMany` con `List<Cita>`
- **CRÍTICO:** Constructor protegido que inicializa citas = `new ArrayList<>()`

```
protected Medico(MedicoBuilder<?, ?> builder) {
```

```

super(builder);

this.citas = new ArrayList<>(); // Obligatorio
}

```

- Usar @SuperBuilder

1.4 Entidad Hospital (3 pts)

- Usar @Entity con @Id @GeneratedValue
- Atributos: nombre, direccion, telefono (final)
- Relación @OneToMany(cascade=ALL, orphanRemoval=true) con List<Departamento>
- Relación @OneToMany(cascade=ALL, orphanRemoval=true) con List<Paciente>
- Métodos helper: agregarDepartamento(), agregarPaciente() que establecen relaciones bidireccionales
- **Patrón Aggregate Root**

1.5 Entidad Departamento (3 pts)

- Usar @Entity con @Id @GeneratedValue
- Atributos: nombre, especialidad (enum, final)
- Relación @ManyToOne con Hospital
- Relación @OneToMany(mappedBy="departamento") con List<Medico>
- Relación @OneToMany(mappedBy="departamento") con List<Sala>
- Método agregarMedico() que valida especialidad compatible

1.6 Entidad Sala (2 pts)

- Usar @Entity con @Id @GeneratedValue
- Atributos: numero (String único), tipo
- Relación @ManyToOne con Departamento (final)
- Relación @OneToMany con List<Cita>

1.7 Entidad HistoriaClinica (2.5 pts)

- Usar @Entity con @Id @GeneratedValue
- Atributo numeroHistoria generado automáticamente (formato HC-{DNI}-{timestamp})
- Relación @OneToOne con Paciente (unique constraint)
- Atributo fechaCreacion
- @ElementCollection para List<String> diagnosticos, tratamientos, alergias
- Métodos: agregarDiagnostico(), agregarTratamiento(), agregarAlergia()

1.8 Entidad Cita (3 pts)

- Usar @Entity con @Id @GeneratedValue
- Relaciones @ManyToOne(cascade={PERSIST,MERGE}) con Paciente, Medico, Sala
- Atributos: fechaHora (LocalDateTime), costo (BigDecimal, final), estado (enum EstadoCita, mutable), observaciones (String, mutable)
- Métodos helper addCita() en entidades relacionadas para mantener bidireccionalidad

1.9 Value Object Matricula (1.5 pts)

- Usar @Embeddable (patrón Value Object)
- Validación en constructor: formato MP-\d{4,6} con regex
- Usado con @Embedded en Medico

1.10 Enums (2.5 pts)

- **TipoSangre** (1 pt): 8 tipos (A_POSITIVO, A_NEGATIVO, B_POSITIVO, B_NEGATIVO, AB_POSITIVO, AB_NEGATIVO, O_POSITIVO, O_NEGATIVO) con método getDescripcion()
- **EspecialidadMedica** (1 pt): 12 especialidades (CARDIOLOGIA, NEUROLOGIA, PEDIATRIA, TRAUMATOLOGIA, GINECOLOGIA, UROLOGIA, OFTALMOLOGIA, DERMATOLOGIA, PSIQUIATRIA, MEDICINA_GENERAL, CIRUGIA_GENERAL, ANESTESIOLOGIA) con método getDescripcion()
- **EstadoCita** (0.5 pt): 5 estados (PROGRAMADA, EN_CURSO, COMPLETADA, CANCELADA, NO_ASISTIO)

2. Configuración JPA y Persistencia (20 puntos)

2.1 Archivo persistence.xml (3 pts)

- Ubicación correcta: src/main/resources/META-INF/persistence.xml
- <persistence-unit name="hospital-persistence-unit">

2.2 Configuración de Hibernate (2 pts)

```
<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
```

2.3 Configuración de H2 Database (3 pts)

```
<property name="jakarta.persistence.jdbc.url" value="jdbc:h2:file:./data/hospidb"/>  
<property name="jakarta.persistence.jdbc.driver" value="org.h2.Driver"/>
```

2.4 Propiedades de Hibernate (4 pts)

```
<property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
<property name="hibernate.hbm2ddl.auto" value="update"/> <!-- o create-drop -->
<property name="hibernate.show_sql" value="true"/>
<property name="hibernate.format_sql" value="true"/>
```

2.5 EntityManagerFactory (2 pts)

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("hospital-persistence-unit");
```

2.6 Operaciones CRUD con EntityManager (2 pts)

- Usar em.persist(), em.merge(), em.find(), em.createQuery()

2.7 Gestión de Transacciones (4 pts)

```
em.getTransaction().begin();
try {
    // Operaciones de persistencia
    em.persist(entidad);
    em.getTransaction().commit();
} catch (Exception e) {
    em.getTransaction().rollback();
    throw e;
}
```

3. Relaciones JPA y Cascading (15 puntos)

3.1 Relaciones @OneToMany con mappedBy (3 pts)

- Hospital.departamentos mappedBy="hospital"
- Departamento.medicos mappedBy="departamento"
- Paciente.citas mappedBy="paciente"
- Etc.

3.2 Relaciones @ManyToOne (3 pts)

- Departamento.hospital
- Medico.departamento
- Paciente.hospital
- Cita.paciente, Cita.medico, Cita.sala

3.3 Relación @OneToOne bidireccional (2 pts)

- Paciente ↔ HistoriaClinica con mappedBy y unique constraint

3.4 CascadeType.ALL (3 pts)

- En relaciones padre-hijo: Hospital → Departamento, Hospital → Paciente

3.5 orphanRemoval=true (2 pts)

- En relaciones de composición: Hospital.departamentos, Hospital.pacientes

3.6 Métodos Helper Bidireccionales (2 pts)

```
public void agregarDepartamento(Departamento dept) {
    this.departamentos.add(dept);
    dept.setHospital(this); // Sincronización bidireccional
}
```

4. Lombok con JPA (@SuperBuilder) (15 puntos)

4.1 @SuperBuilder en Persona (3 pts)

- Clase abstracta con @SuperBuilder para permitir builders en jerarquía

4.2 @SuperBuilder en Medico y Paciente (3 pts)

```
Medico medico = Medico.builder()
    .nombre("Juan")
    .apellido("Pérez")
    .especialidad(CARDIOLOGIA)
    .build();
```

4.3 @NoArgsConstructor protegido (2.5 pts)

```
@NoArgsConstructor(access = AccessLevel.PROTECTED)
```

4.4 @Getter y @Setter apropiados (2.5 pts)

- @Setter(AccessLevel.NONE) en campos inmutables

4.5 Constructor Protegido Personalizado (4 pts) - CRÍTICO

```
protected Medico(MedicoBuilder<?, ?> builder) {  
    super(builder);  
    this.citas = new ArrayList<>(); // Obligatorio - @Builder.Default no funciona con  
    @SuperBuilder  
}
```

⚠ Sin esto, obtendrás NullPointerException al agregar citas

4.6 @Builder en entidades simples (2 pts)

- Hospital, Departamento, Sala, HistoriaClinica, Cita

5. Validaciones y Reglas de Negocio (10 puntos)

5.1 Validación de DNI (2.5 pts)

```
private void validarDni(String dni) {  
    if (!dni.matches("\\d{7,8}")) {  
        throw new IllegalArgumentException("DNI inválido");  
    }  
}
```

5.2 Validación de Matrícula (2.5 pts)

```
public Matricula(String numero) {  
    if (!numero.matches("MP-\\d{4,6}")) {  
        throw new IllegalArgumentException("Formato de matrícula inválido");  
    }  
    this.numero = numero;  
}
```

5.3 Validaciones con Objects.requireNonNull() (2 pts)

```
this.nombre = Objects.requireNonNull(nombre, "Nombre no puede ser null");
```

5.4 Validación de especialidad compatible (1.5 pts)

```
public void agregarMedico(Medico medico) {  
    if (!medico.getEspecialidad().equals(this.especialidad)) {  
        throw new IllegalArgumentException("Especialidad incompatible");  
    }  
    // ... agregar médico  
}
```

5.5 Validación de strings no vacíos (1.5 pts)

```
private void validarString(String valor, String nombreCampo) {  
    if (valor == null || valor.trim().isEmpty()) {  
        throw new IllegalArgumentException(nombreCampo + " no puede estar vacío");  
    }  
}
```

6. Consultas JPQL y TypedQuery (10 puntos)

6.1 Uso de TypedQuery<T> (3 pts)

```
TypedQuery<Hospital> query = em.createQuery("SELECT h FROM Hospital h", Hospital.class);
```

6.2 Consultas básicas SELECT FROM (2 pts)

```
TypedQuery<Medico> query = em.createQuery("SELECT m FROM Medico m", Medico.class);  
List<Medico> medicos = query.getResultList();
```

6.3 Consultas con WHERE y parámetros (3 pts)


```
TypedQuery<Medico> query = em.createQuery(
    "SELECT m FROM Medico m WHERE m.especialidad = :esp", Medico.class);
query.setParameter("esp", EspecialidadMedica.CARDIOLOGIA);
List<Medico> medicos = query.getResultList();
```

6.4 Consultas con ORDER BY (1 pt)

```
TypedQuery<Cita> query = em.createQuery(
    "SELECT c FROM Cita c ORDER BY c.fechaHora", Cita.class);
```

6.5 Consultas de agregación COUNT (1 pt)

```
TypedQuery<Long> query = em.createQuery(
    "SELECT COUNT(c) FROM Cita c WHERE c.estado = :estado", Long.class);
query.setParameter("estado", EstadoCita.PROGRAMADA);
Long total = query.getSingleResult();
```

7. Programa Main - Demostración Completa (15 puntos)

7.1 Inicialización de EntityManager (2 pts)

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("hospital-persistence-unit");
EntityManager em = emf.createEntityManager();

try {
    // ... operaciones
} finally {
    em.close();
    emf.close();
}
```

7.2 Inicialización de datos (3 pts)

- Crear 1 Hospital
- Crear 3 Departamentos (Cardiología, Pediatría, Traumatología)

- Crear salas por departamento
- Crear 3 médicos especialistas
- Crear 3 pacientes con historias clínicas auto-generadas
- Usar builders de Lombok

7.3 Persistencia con cascading (3 pts)

```
em.getTransaction().begin();
```

```
em.persist(hospital); // Cascading persiste departamentos, pacientes, etc.
```

```
em.getTransaction().commit();
```

7.4 Programación de citas (2 pts)

- Crear al menos 3 citas médicas en fechas futuras
- Diferentes especialidades
- Persistir con `em.persist(cita)`
- Asignar estados y observaciones

7.5 Consultas JPQL (2 pts)

- Recuperar hospitales, médicos (filtrado por especialidad), pacientes, citas
- Mostrar resultados con `System.out.println()`

7.6 Actualización de datos (1.5 pts)

```
em.getTransaction().begin();
```

```
cita.setEstado(EstadoCita.COMPLETADA);
```

```
em.merge(cita);
```

```
em.getTransaction().commit();
```

7.7 Estadísticas con COUNT (1.5 pts)

- Médicos por especialidad
- Citas por estado
- Total de pacientes
- Total de salas

7.8 Mensaje de éxito (0.5 pts)

```
System.out.println("SISTEMA EJECUTADO EXITOSAMENTE");
```

8. Calidad de Código y Buenas Prácticas (5 puntos)

8.1 Nomenclatura (1 pt)

- Variables/métodos: camelCase
- Clases: PascalCase
- Constantes: UPPER_SNAKE_CASE

8.2 Organización de paquetes (0.5 pts)

- entidades/ para modelo JPA
- servicio/ para lógica opcional

8.3 Encapsulación (1 pt)

- Atributos privados/protected
- Collections.unmodifiableList() en getters de colecciones

8.4 Inmutabilidad (0.5 pts)

- Uso apropiado de final en atributos core

8.5 Configuración UTF-8 en build.gradle (0.5 pts)

```
tasks.withType(JavaCompile) {  
    options.encoding = 'UTF-8'  
}
```

8.6 Limpieza de código (0.5 pts)

- Sin Javadoc innecesario (código auto-documentado)
- Sin código comentado
- Sin imports sin usar

8.7 Dependencias en build.gradle (1 pt)

```
dependencies {  
    compileOnly 'org.projectlombok:lombok:1.18.38'  
    annotationProcessor 'org.projectlombok:lombok:1.18.38'  
    implementation 'jakarta.persistence:jakarta.persistence-api:3.1.0'
```

```
implementation 'org.hibernate:hibernate-core:6.4.4'

runtimeOnly 'com.h2database:h2:2.2.224'

implementation 'org.slf4j:slf4j-simple:2.0.9'

}
```

Penalizaciones Automáticas

Las siguientes penalizaciones se **restan del puntaje final**:

| Condición | Pe |
|--|----|
| Main.java no ejecuta completamente sin errores | -2 |
| Faltan 3 o más clases de entidades requeridas | -1 |
| persistence.xml no existe o está mal configurado | -1 |
| No usa EntityManager para operaciones CRUD | -1 |
| No implementa relaciones bidireccionales JPA | -1 |
| No usa Lombok @SuperBuilder en Persona→Medico/Paciente | -1 |
| No usa CascadeType.ALL en relaciones apropiadas | -5 |
| No usa TypedQuery (usa Query sin tipos) | -5 |
| No cierra EntityManager/EntityManagerFactory | -5 |

Condiciones de Falla Obligatoria

Si **NO CUMPLES** con estas condiciones, tu nota **NO PUEDE SUPERAR** el porcentaje indicado:


| Condición | Nota I |
|---|--------|
| Persona NO es clase abstracta con @MappedSuperclass | 40% |
| NO existe persistence.xml o no tiene <persistence-unit> | 45% |
| Main.java NO usa EntityManager | 50% |

| Condición | Nota |
|-------------------------------------|------|
| Menos de 7 entidades tienen @Entity | 50% |

Comandos para Compilar y Ejecutar


Compilar el proyecto


 `./gradlew build` # Linux/Mac

 `gradlew.bat build` # Windows

Debe mostrar: BUILD SUCCESSFUL

Ejecutar el programa

 `./gradlew run` # Linux/Mac

 `gradlew.bat run` # Windows

Debe mostrar: SISTEMA EJECUTADO EXITOSAMENTE

Limpiar y resetear base de datos

 `./gradlew clean`

`rm -rf data` # Linux/Mac

`rmdir /s /q data` # Windows

Escala de Calificación

| Rango | Calificación | Descripción |
|--------|---------------|--|
| 90-100 | Exce lente | Implementación completa y correcta. Todas las entidades JPA con anotaciones apropiadas bidireccionales, cascading. Main ejecuta sin errores. Código limpio con Lombok @SuperBu |



| Rango | Calificación | Descripción |
|-------|---------------------|---|
| 80-89 | Muy Bueno | Implementación mayormente correcta con pequeños desajustes no críticos. Todas las entidades implementadas. Main ejecuta correctamente con JPQL funcional. |
| 70-79 | Bueno | Implementación funcional de entidades principales. Algunas relaciones o cascading faltan. funcionalidad básica. persistence.xml configurado pero con omisiones. |
| 60-69 | Aprobado | Implementación básica que cumple requisitos mínimos. Entidades core con @Entity pero incompletas. Main ejecuta con funcionalidad limitada. |
| <60 | Insuficiente | Implementación incompleta. Entidades faltantes o sin anotaciones JPA correctas. Relaciones implementadas. Main no ejecuta correctamente o sin uso de EntityManager. |

Patrones de Diseño Requeridos

1. **Aggregate Root Pattern:** Hospital con cascade = CascadeType.ALL
2. **Value Object Pattern:** Matricula con @Embeddable
3. **Template Method Pattern:** Persona abstracta con @MappedSuperclass
4. **Builder Pattern:** Lombok @Builder y @SuperBuilder

Checklist de Verificación

Antes de entregar, verifica que:

- **13 clases totales:** 9 entidades + 3 enums + 1 embeddable
- Persona es abstracta con @MappedSuperclass
- Medico y Paciente usan @SuperBuilder con constructor protegido que inicializa colecciones
- persistence.xml existe en src/main/resources/META-INF/
- Todas las relaciones JPA son bidireccionales con métodos helper
- Main.java usa EntityManager y ejecuta sin errores
- Consultas JPQL usan TypedQuery<T>
-  ./gradlew build muestra BUILD SUCCESSFUL
-  ./gradlew run muestra SISTEMA EJECUTADO EXITOSAMENTE
- Base de datos H2 se crea en ./data/hospidb.mv.db

Tecnologías y Versiones

- **Java:** 8+
- **JPA:** Jakarta Persistence API 3.1.0
- **Hibernate ORM:** 6.4.4
- **H2 Database:** 2.2.x
- **Lombok:** 1.18.38
- **Build Tool:** Gradle

Recursos Adicionales

- **README.md:** Documentación completa con ejemplos
- **HISTORIAS_USUARIO.md:** 32 historias de usuario con criterios de aceptación
- **CLAUDE.md:** Guía técnica del proyecto

¡Buena suerte! 🚀