Historias de Usuario - Sistema de Gestión Hospitalaria

📋 **Tabla de Contenidos**

Módulo de Gestión Hospitalaria

HU-001: Registrar Hospital en el Sistema

Como administrador del sistema **Quiero** registrar un nuevo hospital con su información básica **Para** comenzar a gestionar sus operaciones médicas y administrativas

Criterios de Aceptación:

- El hospital debe tener un nombre único (máx. 200 caracteres)
- La dirección debe ser completa y válida (máx. 300 caracteres)
- El teléfono debe estar en formato válido (máx. 20 caracteres)
- El sistema genera automáticamente un ID único para el hospital
- Los campos nombre, dirección y teléfono son obligatorios
- Se valida que ningún campo esté vacío o contenga solo espacios

Validaciones Técnicas:

// Validación de campos obligatorios

validarString(nombre, "El nombre del hospital no puede ser nulo ni vacío")

validarString(direccion, "La dirección no puede ser nula ni vacía")

validarString(telefono, "El teléfono no puede ser nulo ni vacío")

Resultado Esperado:

- Hospital registrado con ID único generado automáticamente

- Listas vacías inicializadas para departamentos y pacientes
- Relaciones bidireccionales preparadas para agregar entidades

---

HU-002: Consultar Información de Hospital

Como usuario del sistema **Quiero** consultar la información completa de un hospital **Para** conocer sus datos básicos, departamentos y pacientes registrados

Criterios de Aceptación:

- Mostrar nombre, dirección y teléfono del hospital
- Listar cantidad de departamentos asociados
- Listar cantidad de pacientes registrados
- Presentar información de forma clara y organizada

Consulta JPQL:

```
TypedQuery<Hospital> query = em.createQuery(
   "SELECT h FROM Hospital h WHERE h.id = :id", Hospital.class);
query.setParameter("id", hospitalId);
Hospital hospital = query.getSingleResult();
```

Módulo de Gestión de Departamentos

HU-003: Crear Departamento Médico

Como administrador hospitalario **Quiero** crear departamentos especializados dentro del hospital **Para** organizar la atención médica por especialidades

Criterios de Aceptación:

- El departamento debe tener un nombre descriptivo (máx. 150 caracteres)
- Debe estar asociado a una especialidad médica válida
- La especialidad determina qué tipo de médicos pueden trabajar allí
- El departamento debe estar vinculado a un hospital específico
- Las listas de médicos y salas se inicializan vacías

Especialidades Disponibles:

- CARDIOLOGIA, NEUROLOGIA, PEDIATRIA, TRAUMATOLOGIA
- GINECOLOGIA, UROLOGIA, OFTALMOLOGIA, DERMATOLOGIA

- PSIQUIATRIA, MEDICINA_GENERAL, CIRUGIA_GENERAL, ANESTESIOLOGIA

Relación Bidireccional:

```
hospital.agregarDepartamento(departamento);
// Automáticamente establece: departamento.hospital = hospital
```

HU-004: Asignar Médico a Departamento

Como jefe de departamento **Quiero** asignar médicos a mi departamento **Para** conformar el equipo médico especializado

Criterios de Aceptación:

- El médico debe tener la misma especialidad que el departamento
- No se permite agregar el mismo médico dos veces
- La relación bidireccional se mantiene automáticamente
- El médico queda asociado al departamento y al hospital

Validación de Especialidad:

```
// La especialidad del médico debe coincidir con el departamento
if (!medico.getEspecialidad().equals(departamento.getEspecialidad())) {
   throw new IllegalStateException("Especialidad incompatible");
}
```

Módulo de Gestión de Salas

HU-005: Crear Sala Médica en Departamento

Como administrador de departamento **Quiero** crear salas médicas dentro de mi departamento **Para** disponer de espacios físicos para atender pacientes

Criterios de Aceptación:

- La sala debe tener un número identificador único (máx. 20 caracteres)
- Debe especificarse el tipo de sala (máx. 100 caracteres)
- La sala pertenece exclusivamente a un departamento
- Tipos comunes: "Consultorio", "Quirófano", "Emergencias", "Sala de Observación"
- El número de sala no puede duplicarse en el sistema

Factory Method:

```
Sala sala = departamento.crearSala("CARD-101", "Consultorio");
// Automáticamente vincula la sala al departamento
```

HU-006: Consultar Disponibilidad de Sala

Como coordinador de citas **Quiero** verificar la disponibilidad de una sala **Para** programar nuevas citas médicas

Criterios de Aceptación:

- Mostrar todas las citas programadas en la sala
- Indicar horarios ocupados y disponibles
- Considerar buffer de 2 horas entre citas
- Permitir filtrar por fecha específica

Validación de Disponibilidad:

```
// Buffer de 2 horas entre citas en la misma sala
private boolean esSalaDisponible(Sala sala, LocalDateTime fechaHora) {
   for (Cita citaExistente : citasSala) {
      if (Math.abs(citaExistente.getFechaHora().compareTo(fechaHora)) < 2) {
         return false;
      }
   }
   return true;
}
```

Módulo de Gestión de Médicos

HU-007: Registrar Médico con Matrícula Profesional

Como director médico **Quiero** registrar médicos con su matrícula profesional **Para** acreditar profesionales autorizados para ejercer

Criterios de Aceptación:

- Registrar datos personales heredados de Persona (nombre, apellido, DNI, fecha nacimiento, tipo sangre)
- DNI debe tener formato argentino: 7-8 dígitos numéricos
- Matrícula profesional con formato "MP-XXXX" (4-6 dígitos)
- La matrícula es única en el sistema
- Especialidad médica obligatoria
- Tipo de sangre obligatorio para emergencias

Formato de Validación:

```
// DNI: 7-8 dígitos
if (!dni.matches("\\d{7,8}")) {
   throw new IllegalArgumentException("DNI inválido");
}

// Matrícula: MP- seguido de 4-6 dígitos
if (!numero.matches("MP-\\d{4,6}")) {
   throw new IllegalArgumentException("Matrícula inválida");
}
```

Ejemplo de Registro:

```
Medico cardiologo = Medico.builder()
   .nombre("Carlos")
   .apellido("González")
   .dni("12345678")
   .fechaNacimiento(LocalDate.of(1975, 5, 15))
   .tipoSangre(TipoSangre.A_POSITIVO)
   .numeroMatricula("MP-12345")
   .especialidad(EspecialidadMedica.CARDIOLOGIA)
   .build();
```

HU-008: Consultar Agenda de Médico

Como médico **Quiero** consultar mi agenda de citas **Para** conocer mis pacientes programados

Criterios de Aceptación:

- Mostrar todas las citas del médico ordenadas por fecha
- Incluir información del paciente, sala y horario
- Indicar el estado de cada cita (PROGRAMADA, EN_CURSO, COMPLETADA, CANCELADA)
- Permitir filtrar por rango de fechas

Consulta:

List<Cita> citasMedico = citaManager.getCitasPorMedico(medico);

// Retorna lista inmutable de citas ordenadas

HU-009: Actualizar Disponibilidad de Médico

Como médico **Quiero** que el sistema valide mi disponibilidad automáticamente **Para** evitar sobrecarga de citas

Criterios de Aceptación:

- Buffer mínimo de 2 horas entre citas consecutivas
- Validación automática al programar nueva cita
- Mensaje claro de error si no hay disponibilidad
- Considerar tiempo de consulta y descanso entre pacientes

Módulo de Gestión de Pacientes

HU-010: Registrar Paciente en el Hospital

Como recepcionista **Quiero** registrar un nuevo paciente **Para** que pueda recibir atención médica

Criterios de Aceptación:

- Registrar datos personales (nombre, apellido, DNI, fecha nacimiento, tipo sangre)
- DNI formato argentino: 7-8 dígitos numéricos
- Teléfono de contacto obligatorio (máx. 20 caracteres)
- Dirección completa obligatoria (máx. 300 caracteres)
- Historia clínica creada AUTOMÁTICAMENTE al registrar paciente

- Paciente asignado a un hospital específico

Auto-generación de Historia Clínica:

```
protected Paciente(PacienteBuilder<?, ?> builder) {
   super(builder);
   this.telefono = validarString(builder.telefono, "...");
   this.direccion = validarString(builder.direccion, "...");
   this.citas = new ArrayList<>();

   // ⚠️ CRÍTICO: Historia clínica auto-generada
   this.historiaClinica = HistoriaClinica.builder()
      .paciente(this)
      .build();
}
```

HU-011: Asignar Paciente a Hospital

Como administrador **Quiero** asignar pacientes a hospitales específicos **Para** mantener organizado el registro de pacientes por institución

Criterios de Aceptación:

- Establecer relación bidireccional paciente-hospital
- Permitir transferencia entre hospitales
- Actualizar automáticamente la lista de pacientes del hospital
- Mantener consistencia en la base de datos

Relación Bidireccional:

```
hospital.agregarPaciente(paciente);
// Automáticamente: paciente.hospital = hospital
// Y: hospital.pacientes.add(paciente)
```

HU-012: Consultar Información de Paciente

Como personal médico **Quiero** consultar la información completa de un paciente **Para** conocer sus datos personales y médicos

Criterios de Aceptación:

- Mostrar datos personales completos
- Edad calculada automáticamente desde fecha de nacimiento
- Tipo de sangre visible para emergencias
- Acceso rápido a historia clínica
- Lista de citas programadas y pasadas

Módulo de Historia Clínica

HU-013: Acceder a Historia Clínica de Paciente

Como médico **Quiero** acceder a la historia clínica del paciente **Para** conocer su historial médico antes de la consulta

Criterios de Aceptación:

- Cada paciente tiene UNA ÚNICA historia clínica (constraint UNIQUE)
- Historia clínica creada automáticamente al registrar paciente
- Número de historia auto-generado: HC-{DNI}-{timestamp}
- Fecha de creación registrada para auditoría
- Acceso inmediato a través de paciente.getHistoriaClinica()

⚠️ **Regla Crítica:**

```
// ✅ CORRECTO: Usar historia auto-generada
HistoriaClinica historia = paciente.getHistoriaClinica();

// ❌ INCORRECTO: Crear nueva historia (viola UNIQUE constraint)
HistoriaClinica nueva = HistoriaClinica.builder()
  .paciente(paciente)  // ERROR!
  .build();
```

HU-014: Registrar Diagnóstico Médico

Como médico **Quiero** agregar diagnósticos a la historia clínica **Para** documentar las condiciones médicas del paciente

Criterios de Aceptación:

- Permitir agregar múltiples diagnósticos (máx. 500 caracteres c/u)
- Diagnósticos almacenados en tabla separada (@ElementCollection)
- Validar que el diagnóstico no esté vacío
- Registro acumulativo (no se eliminan diagnósticos previos)
- Mantener orden cronológico de entrada

Implementación:

historia.agregarDiagnostico("Hipertensión arterial");

historia.agregarDiagnostico("Diabetes tipo 2");

// Almacenados en tabla 'diagnosticos'

HU-015: Registrar Tratamiento Prescrito

Como médico **Quiero** registrar tratamientos prescritos **Para** documentar el plan terapéutico del paciente

Criterios de Aceptación:

- Permitir agregar múltiples tratamientos (máx. 500 caracteres c/u)
- Incluir medicamentos, dosis, terapias y procedimientos
- Tratamientos almacenados en tabla separada
- Validar que el tratamiento no esté vacío
- Historial completo de todos los tratamientos

Ejemplos de Tratamientos:

historia.agregarTratamiento("Enalapril 10mg cada 12 horas");

historia.agregarTratamiento("Dieta baja en sodio");

historia.agregarTratamiento("Fisioterapia 3 veces por semana");

HU-016: Registrar Alergias del Paciente

Como médico **Quiero** registrar las alergias conocidas del paciente **Para** evitar prescribir medicamentos o tratamientos peligrosos

Criterios de Aceptación:

- Permitir agregar múltiples alergias (máx. 200 caracteres c/u)
- Información crítica para seguridad del paciente
- Alergias almacenadas en tabla separada
- Validar que la alergia no esté vacía
- DEBE consultarse antes de prescribir medicamentos

⚠️ **Seguridad del Paciente:**

historia.agregarAlergia("Penicilina");

historia.agregarAlergia("Ibuprofeno");

historia.agregarAlergia("Maní");

// Consultar antes de prescribir

List<String> alergias = historia.getAlergias();

// Verificar contra medicamento a prescribir

HU-017: Consultar Historial Completo

Como médico especialista **Quiero** consultar el historial médico completo del paciente **Para** tomar decisiones informadas sobre el tratamiento

Criterios de Aceptación:

- Mostrar todos los diagnósticos históricos
- Listar todos los tratamientos prescritos
- Destacar alergias conocidas (CRÍTICO)
- Mostrar fecha de creación de la historia
- Presentar información ordenada cronológicamente

Módulo de Gestión de Citas

HU-018: Programar Cita Médica

Como recepcionista **Quiero** programar una cita médica **Para** asignar atención al paciente con un médico específico

Criterios de Aceptación:

- Vincular paciente, médico y sala específicos
- Validar que la fecha/hora sea futura (no en el pasado)
- Costo debe ser mayor a cero (BigDecimal para precisión)
- Especialidad del médico debe coincidir con departamento de la sala
- Buffer de 2 horas entre citas del mismo médico
- Buffer de 2 horas entre citas en la misma sala
- Estado inicial: PROGRAMADA
- Observaciones opcionales (máx. 1000 caracteres)

Validaciones de Negocio:

```
// 1. Validación temporal
if (fechaHora.isBefore(LocalDateTime.now())) {
   throw new CitaException("No se puede programar cita en el pasado");
}

// 2. Validación de costo
if (costo.compareTo(BigDecimal.ZERO) <= 0) {
   throw new CitaException("El costo debe ser mayor que cero");
}

// 3. Validación de especialidad
if (!medico.getEspecialidad().equals(sala.getDepartamento().getEspecialidad())) {
   throw new CitaException("Especialidad del médico no coincide con departamento");
}

// 4. Validación de disponibilidad médico (buffer 2 horas)
if (!esMedicoDisponible(medico, fechaHora)) {
   throw new CitaException("Médico no disponible");
}
```

```
// 5. Validación de disponibilidad sala (buffer 2 horas)
if (!esSalaDisponible(sala, fechaHora)) {
   throw new CitaException("Sala no disponible");
}
```

HU-019: Actualizar Estado de Cita

Como personal médico **Quiero** actualizar el estado de las citas **Para** reflejar su progreso y resultado

Criterios de Aceptación:

- Estados disponibles: PROGRAMADA, EN_CURSO, COMPLETADA, CANCELADA, NO_ASISTIO
- Flujo normal: PROGRAMADA → EN_CURSO → COMPLETADA
- Permitir cancelación: PROGRAMADA → CANCELADA
- Registrar no asistencia: PROGRAMADA → NO_ASISTIO
- El estado no puede ser nulo
- Actualización persistida en base de datos

Flujo de Estados:

```
// Al iniciar consulta
cita.setEstado(EstadoCita.EN_CURSO);

// Al finalizar exitosamente
cita.setEstado(EstadoCita.COMPLETADA);

// Si el paciente no asiste
cita.setEstado(EstadoCita.NO_ASISTIO);

// Si se cancela
cita.setEstado(EstadoCita.CANCELADA);
```

HU-020: Agregar Observaciones a Cita

Como médico **Quiero** agregar observaciones a la cita **Para** documentar notas importantes sobre la consulta

Criterios de Aceptación:

- Observaciones de hasta 1000 caracteres
- Información sobre síntomas, hallazgos o instrucciones
- Actualizable en cualquier momento
- Las comas se reemplazan por punto y coma en CSV

Ejemplos de Observaciones:

cita.setObservaciones("Paciente presenta mejoría significativa");

cita.setObservaciones("Control post-operatorio, retirar puntos en 7 días");

cita.setObservaciones("Paciente con antecedentes de hipertensión");

HU-021: Cancelar Cita Médica

Como paciente o recepcionista **Quiero** cancelar una cita programada **Para** liberar el espacio cuando el paciente no pueda asistir

Criterios de Aceptación:

- Solo se pueden cancelar citas en estado PROGRAMADA
- Cambiar estado a CANCELADA
- Liberar disponibilidad del médico y sala
- Permitir reagendar si es necesario
- Registrar observaciones del motivo de cancelación

HU-022: Consultar Citas por Paciente

Como personal administrativo **Quiero** consultar todas las citas de un paciente **Para** revisar su historial de atenciones

Criterios de Aceptación:

- Listar todas las citas (pasadas, presentes, futuras)
- Mostrar fecha, hora, médico y estado
- Incluir observaciones de cada cita
- Ordenar por fecha (más recientes primero)
- Lista inmutable para proteger integridad

Consulta:

List<Cita> citasPaciente = citaManager.getCitasPorPaciente(paciente);

// Retorna Collections.unmodifiableList()

HU-023: Consultar Citas por Médico

Como coordinador médico **Quiero** consultar las citas de un médico **Para** gestionar su agenda y carga de trabajo

Criterios de Aceptación:

- Listar todas las citas del médico
- Filtrar por estado si es necesario
- Mostrar paciente, sala y horario
- Permitir visualización por día/semana/mes
- Identificar espacios disponibles

HU-024: Consultar Citas por Sala

Como administrador de recursos **Quiero** consultar la ocupación de las salas **Para** optimizar el uso de espacios físicos

Criterios de Aceptación:

- Listar todas las citas por sala
- Mostrar horarios ocupados
- Identificar tiempos de limpieza/preparación
- Calcular tasa de ocupación
- Generar reportes de uso

Módulo de Consultas y Reportes

HU-025: Generar Estadísticas por Especialidad

Como director médico **Quiero** consultar cantidad de médicos por especialidad **Para** analizar la cobertura de servicios

Criterios de Aceptación:

- Contar médicos agrupados por especialidad
- Mostrar solo especialidades con médicos activos

- Presentar datos en formato tabla o gráfico
- Permitir exportación de datos

Consulta JPQL:

```
TypedQuery<Long> query = em.createQuery(
   "SELECT COUNT(m) FROM Medico m WHERE m.especialidad = :esp",
   Long.class);
query.setParameter("esp", EspecialidadMedica.CARDIOLOGIA);
Long count = query.getSingleResult();
```

HU-026: Generar Reporte de Citas por Estado

Como administrador **Quiero** consultar estadísticas de citas por estado **Para** analizar la eficiencia del servicio

Criterios de Aceptación:

- Contar citas agrupadas por estado
- Mostrar: PROGRAMADAS, COMPLETADAS, CANCELADAS, NO_ASISTIO, EN_CURSO
- Calcular porcentaje de cada estado
- Identificar tendencias y problemas
- Generar reporte en formato legible

Análisis de Estados:

```
for (EstadoCita estado : EstadoCita.values()) {
   Long count = em.createQuery(
      "SELECT COUNT(c) FROM Cita c WHERE c.estado = :estado",
      Long.class)
      .setParameter("estado", estado)
      .getSingleResult();

   if (count > 0) {
      System.out.println(estado + ": " + count);
```

```
    }
}
```

HU-027: Consultar Pacientes con Alergias

Como farmacéutico o médico **Quiero** identificar pacientes con alergias registradas **Para** prevenir reacciones adversas

Criterios de Aceptación:

- Listar pacientes que tienen alergias documentadas
- Mostrar tipo y cantidad de alergias
- Acceso rápido a lista completa de alergias por paciente
- Destacar información crítica para seguridad

Consulta JPQL:

```
TypedQuery<Paciente> query = em.createQuery(
    "SELECT DISTINCT p FROM Paciente p " +
    "JOIN p.historiaClinica h " +
    "WHERE SIZE(h.alergias) > 0",
    Paciente.class
);
List<Paciente> pacientesConAlergias = query.getResultList();
```

HU-028: Consultar Total de Recursos

Como administrador general **Quiero** consultar totales de recursos del hospital **Para** tener visión general del sistema

Criterios de Aceptación:

- Total de salas disponibles
- Total de pacientes registrados
- Total de médicos activos
- Total de citas programadas
- Presentar dashboard con KPIs

Consultas Agregadas:

```
Long totalSalas = em.createQuery(
   "SELECT COUNT(s) FROM Sala s", Long.class)
   .getSingleResult();

Long totalPacientes = em.createQuery(
   "SELECT COUNT(p) FROM Paciente p", Long.class)
   .getSingleResult();

Long totalMedicos = em.createQuery(
   "SELECT COUNT(m) FROM Medico m", Long.class)
   .getSingleResult();
```

Módulo de Persistencia

HU-029: Persistir Datos con JPA

Como sistema **Quiero** persistir datos usando JPA/Hibernate **Para** mantener la información en base de datos

Criterios de Aceptación:

- Configuración de persistence unit: "hospital-persistence-unit"
- Base de datos H2 file-based: ./data/hospidb
- Schema management: hibernate.hbm2ddl.auto=update
- SQL logging habilitado para debugging
- Estrategia IDENTITY para generación de IDs
- Cascade operations configuradas correctamente
- Orphan removal para mantener integridad

Configuración Persistence:

```
<persistence-unit name="hospital-persistence-unit">
   <properties>
      <property name="jakarta.persistence.jdbc.url"
```

```
            value="jdbc:h2:file:./data/hospidb"/>
        <property name="hibernate.hbm2ddl.auto" value="update"/>
        <property name="hibernate.show_sql" value="true"/>
        <property name="hibernate.format_sql" value="true"/>
    </properties>
</persistence-unit>
```

HU-030: Exportar Citas a CSV

Como administrador **Quiero** exportar las citas a formato CSV **Para** realizar backups o análisis externos

Criterios de Aceptación:

- Formato
  CSV: dniPaciente,dniMedico,numeroSala,fechaHora,costo,estado,observaciones
- Reemplazar comas por punto y coma en observaciones
- Generar archivo en ubicación especificada
- Mantener integridad de datos
- Permitir importación posterior

Serialización:

```
public String toCsvString() {
    return String.format("%s,%s,%s,%s,%s,%s,%s",
        paciente.getDni(),
        medico.getDni(),
        sala.getNumero(),
        fechaHora.toString(),
        costo.toString(),
        estado.name(),
        observaciones.replaceAll(",", ";")
    );
}
```

HU-031: Importar Citas desde CSV

Como administrador **Quiero** importar citas desde archivo CSV **Para** restaurar backups o migrar datos

Criterios de Aceptación:

- Leer archivo CSV línea por línea
- Validar formato de cada línea (7 campos)
- Resolver referencias a pacientes, médicos y salas mediante DNI/número
- Lanzar excepción si entidad no se encuentra
- Reconstruir índices en memoria después de carga
- Advertir que se limpian citas existentes

⚠️ **Advertencia Crítica:**

```
// LIMPIA TODAS las citas existentes antes de cargar
citas.clear();
citasPorPaciente.clear();
citasPorMedico.clear();
citasPorSala.clear();
// Luego carga desde CSV
```

HU-032: Gestionar Transacciones

Como sistema **Quiero** gestionar transacciones de base de datos correctamente **Para** mantener la integridad y consistencia de los datos

Criterios de Aceptación:

- Iniciar transacción antes de operaciones de escritura
- Commit si operaciones exitosas
- Rollback en caso de error
- Cerrar recursos (EntityManager) en finally
- Manejar excepciones apropiadamente

Patrón de Transacción:

```
EntityManager em = emf.createEntityManager();
```

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

try {
   // Operaciones de persistencia
   em.persist(entity);
   em.getTransaction().commit();

} catch (Exception e) {
   em.getTransaction().rollback();
   throw e;

} finally {
   em.close();
}
```

Matriz de Trazabilidad

ID Historia	Módulo	Entidad Principal	Prioridad	Complejidad
HU-001	Hospital	Hospital	Alta	Baja
HU-002	Hospital	Hospital	Media	Baja
HU-003	Departamento	Departamento	Alta	Media
HU-004	Departamento	Medico, Departamento	Alta	Media
HU-005	Sala	Sala	Alta	Media
HU-006	Sala	Sala, Cita	Media	Alta
HU-007	Médico	Medico, Matricula	Alta	Media
HU-008	Médico	Medico, Cita	Media	Baja
HU-009	Médico	Medico, Cita	Alta	Alta

ID Historia	Módulo	Entidad Principal	Prioridad	Complejidad
HU-010	Paciente	Paciente	Alta	Media
HU-011	Paciente	Paciente, Hospital	Media	Baja
HU-012	Paciente	Paciente	Media	Baja
HU-013	Historia	HistoriaClinica	Alta	Media
HU-014	Historia	HistoriaClinica	Alta	Baja
HU-015	Historia	HistoriaClinica	Alta	Baja
HU-016	Historia	HistoriaClinica	Crítica	Baja
HU-017	Historia	HistoriaClinica	Media	Media
HU-018	Cita	Cita, CitaManager	Crítica	Alta
HU-019	Cita	Cita	Alta	Baja
HU-020	Cita	Cita	Media	Baja
HU-021	Cita	Cita	Alta	Media
HU-022	Cita	Cita, Paciente	Media	Baja
HU-023	Cita	Cita, Medico	Media	Baja
HU-024	Cita	Cita, Sala	Media	Baja
HU-025	Reportes	Medico	Media	Media
HU-026	Reportes	Cita	Media	Media
HU-027	Reportes	Paciente, HistoriaClinica	Alta	Media
HU-028	Reportes	Varias	Baja	Baja
HU-029	Persistencia	Todas	Crítica	Alta

ID Historia	Módulo	Entidad Principal	Prioridad	Complejidad
HU-030	Persistencia	Cita	Media	Media
HU-031	Persistencia	Cita	Media	Alta
HU-032	Persistencia	Todas	Crítica	Media

Reglas de Negocio Críticas

RN-001: Validación de DNI Argentino

- Formato: 7-8 dígitos numéricos
- Sin guiones, puntos ni espacios
- Único en el sistema

RN-002: Validación de Matrícula Profesional

- Formato: "MP-" + 4 a 6 dígitos
- Única en el sistema
- Obligatoria para médicos

RN-003: Historia Clínica Única

- **UNA ÚNICA** historia por paciente
- Auto-generada en constructor de Paciente
- NUNCA crear manualmente
- Constraint UNIQUE en paciente_id

RN-004: Validación de Citas

- Fecha/hora futura obligatoria
- Costo > 0
- Especialidad médico = Especialidad departamento de sala
- Buffer 2 horas entre citas del mismo médico
- Buffer 2 horas entre citas en misma sala

RN-005: Relaciones Bidireccionales

- Siempre usar métodos helper (agregarXxx, addXxx)
- NUNCA usar setters directamente
- Mantiene consistencia automáticamente

RN-006: Estados de Cita

- Estado inicial: PROGRAMADA
- Flujos válidos:
    - PROGRAMADA → EN_CURSO → COMPLETADA
    - PROGRAMADA → CANCELADA
    - PROGRAMADA → NO_ASISTIO

RN-007: Seguridad del Paciente

- Alergias DEBEN consultarse antes de prescribir
- Tipo de sangre visible para emergencias
- Historia clínica accesible para médicos

Glosario de Términos

Aggregate Root: Entidad principal que controla el ciclo de vida de un grupo de entidades relacionadas (ej: Hospital)

Value Object: Objeto sin identidad propia, identificado por su valor (ej: Matricula)

Bidirectional Relationship: Relación que se mantiene desde ambos lados automáticamente

Cascade Operations: Operaciones que se propagan automáticamente a entidades relacionadas

Orphan Removal: Eliminación automática de entidades sin padre

@ElementCollection: Colección de valores simples almacenados en tabla separada

Buffer de Citas: Tiempo mínimo entre citas (2 horas) para limpieza y preparación

JPQL: Java Persistence Query Language - Lenguaje de consultas orientado a objetos

Constraint UNIQUE: Restricción de base de datos que garantiza valores únicos

SuperBuilder: Patrón Lombok que permite herencia de builders

Versión: 1.0 **Fecha**: Octubre 2025 **Equipo**: Los Cortez Development Team