

# **Sistema de Gestión de Licencias de Conducir**

Metodologías Ágiles  
Scrum

Universidad Tecnológica Nacional,  
Facultad Regional Santa Fe

Profesor:  
Ledesma, Rodrigo

Integrantes:  
Brunas, Kevin  
Debona, Marcos  
Gaitán, Gonzalo  
Paggi, Santino

4 de julio de 2025.

# ÍNDICE

<b>PLANIFICACIÓN.....</b>	<b>2</b>
HISTORIAS REFINADAS.....	2
ESTIMACIÓN DE ESFUERZO Y PRIORIDAD.....	6
HISTORIAS DE USUARIO CON SUS PRIORIDADES Y ESTIMACIÓN DEL ESFUERZO EN STORY POINTS.....	7
DEFINICIÓN DE VELOCIDAD, ALCANCE Y TAMAÑO DEL SPRINT.....	7
PRODUCT BACKLOG.....	8
FECHA DE INICIO DEL DESARROLLO.....	8
<b>PLANIFICACIÓN SPRINT 1.....</b>	<b>9</b>
ESTIMACIÓN DE ESFUERZO, PLANIFICACIÓN Y SPRINT BACKLOG.....	9
DIVISIÓN EN TAREAS.....	9
TAREAS DETALLADAS.....	10
HISTORIA 1: EMITIR LICENCIA.....	10
HISTORIA 2: CALCULAR VIGENCIA DE LICENCIA.....	22
HISTORIA 3: CALCULAR COSTO DE LICENCIA.....	25
HISTORIA 4: IMPRIMIR LICENCIA.....	28
HISTORIA 5: RENOVAR LICENCIA.....	32
HISTORIA 6: LISTADO DE LICENCIAS QUE HAN EXPIRADO.....	37
HISTORIA 7: DAR DE ALTA UN TITULAR.....	42
GRÁFICO BURN DOWN.....	47
DESCRIPCIÓN DE CADA DAILY MEETING.....	48
REVISIÓN DEL SPRINT.....	49
RETROSPECTIVA DEL SPRINT.....	49
<b>PLANIFICACIÓN SPRINT 2.....</b>	<b>50</b>
ESTIMACIÓN DE ESFUERZO, PLANIFICACIÓN Y SPRINT BACKLOG.....	50
DIVISIÓN EN TAREAS.....	50
TAREAS DETALLADAS.....	51
HISTORIA 8: EMITIR UNA COPIA.....	51
HISTORIA 9: LISTADO DE LICENCIAS VIGENTES POR CRITERIOS.....	52
HISTORIA 10: MODIFICAR LOS DATOS DE UN TITULAR EXISTENTE.....	57
HISTORIA 11: DAR DE ALTA UN USUARIO.....	60
HISTORIA 12: MODIFICAR LOS DATOS DE UN USUARIO.....	65
HISTORIA 13: INICIAR SESIÓN.....	67
HISTORIA 14: CERRAR SESIÓN.....	72
GRÁFICO BURN DOWN.....	72
DESCRIPCIÓN DE CADA DAILY MEETING.....	73
REVISIÓN DEL SPRINT.....	74
RETROSPECTIVA DEL SPRINT.....	74
<b>CONCLUSIONES.....</b>	<b>75</b>

## PLANIFICACIÓN

### HISTORIAS REFINADAS

Historia de Usuario															
<b>Nombre:</b> Emitir Licencia	<b>Número:</b> 1														
<b>Prioridad:</b> Alta	<b>Story Points:</b> 7														
Descripción															
<p>Yo como usuario, necesito crear licencias y especificar en la misma, si fuera necesario, las observaciones del titular, de forma que quede registrado quién llevó a cabo el trámite y la fecha del mismo. La edad mínima para la obtención de una licencia depende de la clase de la misma:</p> <ul style="list-style-type: none"> <li>• Clases C, D y E: 21 años.</li> <li>• Restantes Clases: 17 años.</li> </ul> <p>Los conductores con licencia de conductor de las clases C, D y E son considerados conductores profesionales. Para que se les pueda emitir una licencia de este tipo, debe habérselas otorgado una licencia de clase B al menos un año antes. Además, no se puede otorgar licencia profesional por primera vez a personas de más de 65 años.</p> <table border="1"> <tr> <td>Clase A</td><td>Ciclomotores, motocicletas y triciclos motorizados.</td></tr> <tr> <td>Clase B</td><td>Automóviles y camionetas con acoplado.</td></tr> <tr> <td>Clase C</td><td>Camiones sin acoplado y los comprendidos en la clase B.</td></tr> <tr> <td>Clase D</td><td>Servicio de transporte de pasajeros, emergencia, seguridad y los comprendidos en la clase B o C, según el caso.</td></tr> <tr> <td>Clase E</td><td>Camiones articulados o con acoplado, maquinaria especial no agrícola y los comprendidos en la clase B y C.</td></tr> <tr> <td>Clase F</td><td>Automotores especialmente adaptados para discapacitados.</td></tr> <tr> <td>Clase G</td><td>Tractores agrícolas y maquinaria especial agrícola.</td></tr> </table>		Clase A	Ciclomotores, motocicletas y triciclos motorizados.	Clase B	Automóviles y camionetas con acoplado.	Clase C	Camiones sin acoplado y los comprendidos en la clase B.	Clase D	Servicio de transporte de pasajeros, emergencia, seguridad y los comprendidos en la clase B o C, según el caso.	Clase E	Camiones articulados o con acoplado, maquinaria especial no agrícola y los comprendidos en la clase B y C.	Clase F	Automotores especialmente adaptados para discapacitados.	Clase G	Tractores agrícolas y maquinaria especial agrícola.
Clase A	Ciclomotores, motocicletas y triciclos motorizados.														
Clase B	Automóviles y camionetas con acoplado.														
Clase C	Camiones sin acoplado y los comprendidos en la clase B.														
Clase D	Servicio de transporte de pasajeros, emergencia, seguridad y los comprendidos en la clase B o C, según el caso.														
Clase E	Camiones articulados o con acoplado, maquinaria especial no agrícola y los comprendidos en la clase B y C.														
Clase F	Automotores especialmente adaptados para discapacitados.														
Clase G	Tractores agrícolas y maquinaria especial agrícola.														

Historia de Usuario	
<b>Nombre:</b> Calcular vigencia de licencia	<b>Número:</b> 2
<b>Prioridad:</b> Alta	<b>Story Points:</b> 4
Descripción	
<p>Yo como usuario, necesito poder calcular la vigencia de la licencia de acuerdo a las vigencias proporcionadas y a la fecha de nacimiento del titular. Las vigencias son:</p> <ul style="list-style-type: none"> <li>• Menores de 21 años: 1 año la primera vez y 3 años las siguientes.</li> <li>• Hasta 46 años: 5 años.</li> <li>• Hasta 60 años: 4 años.</li> <li>• Hasta 70 años: 3 años.</li> <li>• Mayores de 70 años: 1 año.</li> </ul> <p>El día y mes de la fecha de vencimiento deben coincidir con el día y mes de la fecha de nacimiento del titular, respectivamente.</p>	

**Historia de Usuario****Nombre:** Calcular costo de licencia**Número:** 3**Prioridad:** Media**Story Points:** 4**Descripción**

Yo como usuario, necesito poder obtener el costo de la licencia de acuerdo a la clase de la misma y su vigencia en años, tal como lo especifica la tabla correspondiente y sin estar sujeto a modificaciones por parte de un operador. Además, el costo en concepto de gastos administrativos es de \$8. Tabla de costos y vigencias:

Clase	Vigencia del permiso			
	5 años	4 años	3 años	1 año
A	\$40	\$30	\$25	\$20
B	\$40	\$30	\$25	\$20
C	\$47	\$35	\$30	\$23
D	\$50	\$40	\$35	\$28
E	\$59	\$44	\$39	\$29
F	\$40	\$30	\$25	\$20
G	\$40	\$30	\$25	\$20

**Historia de Usuario****Nombre:** Imprimir licencia**Número:** 4**Prioridad:** Alta**Story Points:** 12**Descripción**

Yo como usuario, necesito imprimir la licencia y un comprobante, con la finalidad de que el interesado abone el costo del trámite en caja.

**Historia de Usuario****Nombre:** Renovar licencia**Número:** 5**Prioridad:** Media**Story Points:** 4**Descripción**

Yo como usuario, necesito poder renovar una licencia, con la finalidad de extender su duración por expiración de su fecha de vigencia o por modificar los datos del titular de la misma. El costo de renovación por modificación de datos es el mismo que el de emisión del carnet.

**Historia de Usuario****Nombre:** Listado de licencias que han expirado**Número:** 6**Prioridad:** Media**Story Points:** 6**Descripción**

Yo como usuario, necesito poder emitir el listado de licencias expiradas, con la finalidad de poder consultar las mismas en rangos de fechas específicas, ordenadas de la última vencida hasta la más antigua. La misma debe mostrar el nombre del titular, la fecha de creación y la de expiración.

**Historia de Usuario****Nombre:** Dar de alta un titular**Número:** 7**Prioridad:** Alta**Story Points:** 4**Descripción**

Yo como usuario, necesito poder dar de alta un titular ingresando sus datos, con la finalidad de que el mismo obtenga su licencia de conducir. Los datos requeridos del titular son:

- N° de documento.
- Nombre.
- Apellido.
- Fecha de nacimiento.
- Dirección.
- Clase/s solicitada/s (A-G)
- Grupo sanguíneo y factor RH.
- Donante de órganos (SÍ/NO).

**Historia de Usuario****Nombre:** Emitir una copia**Número:** 8**Prioridad:** Baja**Story Points:** 4**Descripción**

Yo como usuario, necesito poder emitir una copia de la licencia, con la finalidad de cubrir las necesidades de los titulares. La licencia debe tener el mismo formato original y contener los datos de la licencia perdida. Tiene un costo de \$50 sin importar los tipos y cantidad de licencias que se renueven.

**Historia de Usuario****Nombre:** Listado de licencias vigentes por criterios**Número:** 9**Prioridad:** Baja**Story Points:** 8**Descripción**

Yo como usuario, necesito poder emitir el listado de licencias vigentes filtradas por criterios, tales como por nombre y apellido, grupo sanguíneo y factor RH y por cuyos titulares son donantes de órganos, con la finalidad de consultar las mismas de acuerdo a estos criterios específicos. Además, las licencias no vigentes deben almacenarse a modo de historial, de tal manera que sea posible auditar el sistema.

**Historia de Usuario****Nombre:** Modificar los datos de un titular existente**Número:** 10**Prioridad:** Baja**Story Points:** 4**Descripción**

Yo como usuario, necesito poder modificar todos los datos personales de un titular existente, con la finalidad de actualizar su información personal.

**Historia de Usuario****Nombre:** Dar de alta un usuario**Número:** 11**Prioridad:** Baja**Story Points:** 4**Descripción**

Yo como administrador, necesito poder dar de alta un usuario con la finalidad de que pueda ingresar al sistema y usar las funcionalidades que le corresponden. El usuario tendrá los siguientes atributos: nombre, apellido, DNI, email, dirección, fecha de nacimiento, teléfono.

**Historia de Usuario****Nombre:** Modificar los datos de un usuario**Número:** 12**Prioridad:** Baja**Story Points:** 3**Descripción**

Yo como administrador, necesito poder modificar los datos de los usuarios encargados de operar el sistema. El administrador puede modificar todos los atributos del usuario, excepto su contraseña.

**Historia de Usuario****Nombre:** Iniciar sesión**Número:** 13**Prioridad:** Baja**Story Points:** 4**Descripción**

Yo como usuario, debo poder iniciar sesión especificando DNI y contraseña, de modo que pueda aprovechar las utilidades del sistema.

**Historia de Usuario****Nombre:** Cerrar sesión**Número:** 14**Prioridad:** Baja**Story Points:** 2**Descripción**

Yo como usuario, debo poder cerrar sesión con la finalidad de proteger la integridad de mi perfil y evitar que otros registren actividades en el sistema en mi nombre.

**ESTIMACIÓN DE ESFUERZO Y PRIORIDAD**

Considerando el dominio de aplicación del trabajo, se determinó utilizar Story Points como horas de trabajo ideales, estableciendo que 1 Story Point equivale a 1 hora. La estimación del esfuerzo para cada historia fue realizada por el Scrum Team mediante brainstorming. Durante este proceso, cada miembro presentó su estimación inicial y, luego de analizar y mediar las diferencias encontradas, se llegó a un consenso que reflejaba la percepción colectiva del esfuerzo necesario. Al contar todos los miembros del equipo con experiencia previa similar, no se presentaron diferencias significativas al momento de estimar las historias de usuario, lo que permitió alcanzar rápidamente un acuerdo.

Por ejemplo, en la estimación de la Historia de Usuario 1 (Emitir Licencia), cada miembro presentó su estimación inicial basándose en su entendimiento de los requisitos y experiencia previa. Como resultado, se obtuvo:

- Kevin Brunas: estimó 6 story points.
- Marcos Debona: estimó 8 story points.
- Gonzalo Gaitán: estimó 7 story points.
- Santino Paggi: estimó 6 story points.

Como fuimos el mismo grupo de trabajo tanto para la materia de Desarrollo de Software y Diseño de Sistemas, todos teníamos básicamente la misma experiencia previa, por lo tanto, las diferencias fueron mediadas por charlas, considerando los temas centrales de esa historia (creación de entidades, servicios, controladores, repositorios, preparación del frontend, validación de campos y testing).

De esta manera, gracias al diálogo, el equipo llegó a un consenso de que 7 story points era la cantidad más conveniente para la historia.

Por otro lado, la prioridad de las historias de usuario fue establecida a través de una reunión exhaustiva con el Product Owner, donde se evaluaron las habilidades de desarrollo del Scrum Team, se analizó la experiencia previa del Product Owner en el desarrollo de este tipo de sistemas, y se consideró el valor de negocio que cada funcionalidad aportaría al producto final. Esta colaboración entre el Product Owner y el equipo de desarrollo aseguró que las prioridades reflejarán tanto las necesidades del negocio como las capacidades técnicas del equipo.

**HISTORIAS DE USUARIO CON SUS PRIORIDADES Y ESTIMACIÓN DEL ESFUERZO EN STORY POINTS**

<u>N°</u>	<u>Resumen</u>	<u>Prioridad</u>	<u>Story Points</u>
1	Emitir licencia	Alta	7
2	Calcular vigencia de licencia	Alta	4
3	Calcular costo de licencia	Media	4
4	Imprimir licencia	Alta	12
5	Renovar licencia	Media	4
6	Listado de licencias que han expirado	Media	6
7	Dar de alta un titular	Alta	4
8	Emitir una copia	Baja	4
9	Listado de licencias vigentes por criterios	Baja	8
10	Modificar los datos de un titular existente	Baja	4
11	Dar de alta un usuario	Baja	4
12	Modificar los datos de un usuario	Baja	3
13	Iniciar sesión	Baja	4
14	Cerrar sesión	Baja	2

**DEFINICIÓN DE VELOCIDAD, ALCANCE Y TAMAÑO DEL SPRINT**

En base al tiempo disponible y a las recomendaciones del Scrum Master, el equipo acordó que la duración del sprint sea de 2 semanas: el primero del 5 al 19 de Junio y el segundo del 20 de Junio al 4 de Julio.

Con respecto a la velocidad, realizamos una puesta en común sobre la cantidad de horas por sprint que puede aportar cada integrante. Llegamos a la conclusión de que el sprint no puede superar los 45 story points.

Una vez definidos estos dos valores, en la reunión de planificación al comienzo de cada sprint, se establecerá el alcance de cada uno. Finalmente, el primer sprint se dedicó a implementar las historias de usuario de prioridad alta y media y el segundo a las de baja prioridad.



**PRODUCT BACKLOG**

14 actividades			70 Story Points
<u>N°</u>	<u>Resumen</u>	<u>Prioridad</u>	<u>Story Points</u>
1	Emitir licencia	Alta	7
2	Calcular vigencia de licencia	Alta	4
3	Calcular costo de licencia	Media	4
4	Imprimir licencia	Alta	12
5	Renovar licencia	Media	4
6	Listado de licencias que han expirado	Media	6
7	Dar de alta un titular	Alta	4
8	Emitir una copia	Baja	4
9	Listado de licencias vigentes por criterios	Baja	8
10	Modificar los datos de un titular existente	Baja	4
11	Dar de alta un usuario	Baja	4
12	Modificar los datos de un usuario	Baja	3
13	Iniciar sesión	Baja	4
14	Cerrar sesión	Baja	2

**FECHA DE INICIO DEL DESARROLLO**

- Sprint 1: del 5/06 al 19/06 inclusive. Se buscará que la primera etapa de desarrollo tenga mayores story points a completar debido a la naturaleza del cuatrimestre, pues al finalizar el mismo hay mayores ocupaciones, por lo que para el equipo de desarrollo es conveniente trabajar más en un principio. Luego comentaremos sobre los efectos que causó esta decisión.
- Sprint 2: del 20/06 al 04/07 inclusive. Se buscará que la segunda etapa de desarrollo tenga menos carga horaria y de esfuerzo (respetando aún así la prioridad de las historias), pues se estará en el fin de cuatrimestre, la diversidad de ocupaciones aumentan.

## PLANIFICACIÓN SPRINT 1

### ESTIMACIÓN DE ESFUERZO, PLANIFICACIÓN Y SPRINT BACKLOG

Sprint 1 - Backlog:

7 actividades			41 Story Points
<u>N°</u>	<u>Historia de usuario</u>	<u>Prioridad</u>	<u>Story Points</u>
1	Emitir licencia	Alta	7
2	Calcular vigencia de licencia	Alta	4
3	Calcular costo de licencia	Media	4
4	Imprimir licencia	Alta	12
5	Renovar licencia	Media	4
6	Listado de licencias que han expirado	Media	6
7	Dar de alta un titular	Alta	4

### DIVISIÓN EN TAREAS

Teniendo en cuenta que cada task point (abreviado como *tp*) tiene una duración de 30min, determinamos la siguiente división de tareas:

<u>N°</u>	<u>Historia de usuario</u>	<u>N°</u>	<u>Tareas</u>
1	Emitir licencia	1.1	Realizar testing
		1.2	Validar campos para el formulario
		1.3	Crear controladores
		1.4	Diseñar interfaz
		1.5	Crear entidades
		1.6	Crear repositorios
		1.7	Crear servicios
2	Calcular vigencia de licencia	2.1	Implementar cálculo de vigencia
		2.2	Realizar testing de vigencia
3	Calcular costo de licencia	3.1	Implementar cálculo de costo
		3.2	Realizar testing de cálculo de costo
4	Imprimir licencia	4.1	Llamar a interfaz de impresión del navegador
		4.2	Diseñar licencia para imprimir

		4.3	Realizar testing en imprimir licencia
		4.4	Diseñar comprobante para imprimir
5	Renovar licencia	5.1	Realizar testing de renovar
		5.2	Diseñar interfaz para renovar
		5.3	Calcular el costo de renovación
		5.4	Persistir los datos de la licencia renovada
		5.5	Obtener la nueva fecha de expiración
		5.6	Validar y modificar los datos de la licencia seleccionada para renovar
6	Listado de licencias que han expirado	6.1	Obtener el listado
		6.2	Diseñar interfaz de usuario
		6.3	Realizar testing de licencias expiradas
7	Dar de alta un titular	7.1	Diseñar interfaz para dar de alta
		7.2	Realizar testing
		7.3	Crear el controlador
		7.4	Crear el servicio y repositorio
		7.5	Agregar validación de campos para el formulario

## TAREAS DETALLADAS

### HISTORIA 1: EMITIR LICENCIA

#### TAREA 1.1: REALIZAR TESTING

Descripción				
Desarrollar pruebas unitarias completas usando JUnit. Incluir casos de prueba para validaciones de edad, requisitos por clase de licencia, y escenarios de error.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	4	3	18/06	18/06
Código				
<u>Test edad menor a 17 años para clase B:</u> <pre>@Test void testEdadMenorA17ParaClaseB_lanzaExcepcion() {     Titular titular = new Titular();     titular.setNumeroDocumento("12345678");     titular.setFechaNacimiento(LocalDate.now().minusYears(16));      when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular); }</pre>				

```
LicenciaFormDTO form = buildForm("12345678", "B");
```

```
Exception ex = assertThrows(IllegalArgumentException.class,
    () -> licenciaService.emitirLicencia(form));
assertTrue(ex.getMessage().contains("al menos 17 años"));
```

```
}
```

#### Test edad menor a 21 años para clase profesional C:

@Test

```
void testEdadMenorA21ParaClaseC_lanzaExcepcion() {
```

```
    Titular titular = new Titular();
    titular.setNumeroDocumento("12345678");
    titular.setFechaNacimiento(LocalDate.now().minusYears(20));
```

```
    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);
```

```
    LicenciaFormDTO form = buildForm("12345678", "C");
```

```
    Exception ex = assertThrows(IllegalArgumentException.class,
        () -> licenciaService.emitirLicencia(form));
    assertTrue(ex.getMessage().contains("al menos 21 años"));
```

```
}
```

#### Test clase profesional sin licencia B previa:

@Test

```
void testClaseProfesionalSinLicenciaB_lanzaExcepcion() {
```

```
    Titular titular = new Titular();
    titular.setNumeroDocumento("12345678");
    titular.setFechaNacimiento(LocalDate.now().minusYears(30));
```

```
    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);
```

```
    when(licenciaActivaRepository.findByTitular_NumeroDocumento("12345678"))
        .thenReturn(null);
```

```
    when(licenciaExpiradaRepository.findByTitular_NumeroDocumento("12345678"))
        .thenReturn(Collections.emptyList());
```

```
    LicenciaFormDTO form = buildForm("12345678", "D");
```

```
    Exception ex = assertThrows(IllegalArgumentException.class,
        () -> licenciaService.emitirLicencia(form));
    assertTrue(ex.getMessage().contains("licencia B"));
```

```
}
```

#### Test clase profesional sin antigüedad suficiente en licencia B:

@Test

```
void testClaseProfesionalSinAntigüedadLicenciaB_lanzaExcepcion() {
```

```
    Titular titular = new Titular();
    titular.setNumeroDocumento("12345678");
    titular.setFechaNacimiento(LocalDate.now().minusYears(30));
```

```
    LicenciaExpirada licB = new LicenciaExpirada();
    licB.setClases("B");
    licB.setFechaEmision(LocalDate.now().minusMonths(6));
```

```
    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);
```

```

when(licenciaActivaRepository.findByTitular_NumeroDocumento("12345678"))
    .thenReturn(null);
when(licenciaExpiradaRepository.findByTitular_NumeroDocumento("12345678"))
    .thenReturn(List.of(licB));

```

```

LicenciaFormDTO form = buildForm("12345678", "C");

```

```

Exception ex = assertThrows(IllegalArgumentException.class,
    () -> licenciaService.emitirLicencia(form));
assertTrue(ex.getMessage().contains("al menos 1 año de antigüedad"));

```

```

}

```

Test clase profesional mayor a 65 sin licencia previa:

@Test

```

void testClaseProfesionalMayorA65SinLicenciaPrevia_lanzaExcepcion() {

```

```

    Titular titular = new Titular();
    titular.setNumeroDocumento("12345678");
    titular.setFechaNacimiento(LocalDate.now().minusYears(66));

```

```

    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);
    when(licenciaActivaRepository.findByTitular_NumeroDocumento("12345678")).thenReturn(null);
    when(licenciaExpiradaRepository.findByTitular_NumeroDocumento("12345678"))
        .thenReturn(Collections.emptyList());

```

```

    LicenciaFormDTO form = buildForm("12345678", "D");

```

```

    Exception ex = assertThrows(IllegalArgumentException.class,
        () -> licenciaService.emitirLicencia(form));
    assertTrue(ex.getMessage().contains("Edad máxima para primera licencia profesional"));

```

```

}

```

Test titular no existe:

@Test

```

void testTitularNoExiste_lanzaExcepcion() {

```

```

    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(null);
    LicenciaFormDTO form = buildForm("12345678", "B");
    Exception ex = assertThrows(TitularNoEncontradoException.class,
        () -> licenciaService.emitirLicencia(form));
    assertTrue(ex.getMessage().contains("No se encontró un titular"));

```

```

}

```

Test emisión exitosa de licencia clase B:

@Test

```

void testEmitirLicenciaClaseB_ok() {

```

```

    Titular titular = new Titular();
    titular.setNumeroDocumento("12345678");
    titular.setFechaNacimiento(LocalDate.now().minusYears(30));

```

```

    Usuario usuario = new Usuario();
    usuario.setNumeroDocumento("11999888");

```

```

    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);
    when(usuarioRepository.findByNumeroDocumento(any())).thenReturn(usuario);
    when(licenciaActivaRepository.findByTitular_NumeroDocumento("12345678"))
        .thenReturn(null);
    when(licenciaExpiradaRepository.findByTitular_NumeroDocumento("12345678"))
        .thenReturn(Collections.emptyList());

```

```

when(licenciaEmitidaMapper.entityToDto(any())).thenReturn(new LicenciaEmitidaDTO());

LicenciaFormDTO form = buildForm("12345678", "B");

assertDoesNotThrow(() -> licenciaService.emitirLicencia(form));
}

```

Test emisión exitosa de licencia clase C con licencia B previa:

@Test

```

void testEmitirLicenciaClaseCConLicenciaB_ok() {
    Titular titular = new Titular();
    titular.setNumeroDocumento("12345678");
    titular.setFechaNacimiento(LocalDate.now().minusYears(30));

    Usuario usuario = new Usuario();
    usuario.setNumeroDocumento("11999888");

    LicenciaExpirada licB = new LicenciaExpirada();
    licB.setClases("B");
    licB.setFechaEmision(LocalDate.now().minusYears(2));

    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);
    when(usuarioRepository.findByNumeroDocumento(any())).thenReturn(usuario);
    when(licenciaActivaRepository.findByTitular_NumeroDocumento("12345678"))
        .thenReturn(null);
    when(licenciaExpiradaRepository.findByTitular_NumeroDocumento("12345678"))
        .thenReturn(List.of(licB));
    when(licenciaEmitidaMapper.entityToDto(any())).thenReturn(new LicenciaEmitidaDTO());

    LicenciaFormDTO form = buildForm("12345678", "C");

    assertDoesNotThrow(() -> licenciaService.emitirLicencia(form));
}

```

## TAREA 1.2: VALIDAR CAMPOS PARA EL FORMULARIO

Descripción				
Implementar las validaciones en el backend para todos los campos del formulario. Validaciones incluyen: verificación de edad mínima según clase de licencia (21 años para C, D, E; 17 años para el resto), validación de formato de documento, verificación de que conductores profesionales (C, D, E) tengan licencia B previa de al menos 1 año, restricción de edad máxima de 65 años para primera licencia profesional, y validación de campos obligatorios.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	2	3	18/06	18/06
Código				
<pre> private void validarFormularioLicencia(LicenciaFormDTO form) {     // Verifica que el documento del titular no sea nulo o vacío     if (form.getDocumentoTitular() == null    form.getDocumentoTitular().isBlank())         throw new IllegalArgumentException("El documento del titular es obligatorio");     // Verifica que se haya especificado al menos una clase </pre>				

```

if (form.getClases() == null || form.getClases().isBlank())
    throw new IllegalArgumentException("Debe especificar al menos una clase");
// Verifica que el documento tenga entre 7 y 9 dígitos numéricos
if (!form.getDocumentoTitular().matches("\\d{7,9}"))
    throw new IllegalArgumentException("El documento debe tener entre 7 y 9 dígitos numéricos");

// Busca el titular en la base de datos
Titular titular = titularRepository.findByNumeroDocumento(form.getDocumentoTitular());
if (titular == null)
    throw new TitularNoEncontradoException("No se encontró un titular con el número de documento: " +
form.getDocumentoTitular());

int edad = titular.calcularEdad();
String[] clases = form.getClases().trim().split("\\s+");

for (String clase : clases) {
    boolean esProfesional = clase.equals("C") || clase.equals("D") || clase.equals("E");
    if (esProfesional) {
        // Para clases profesionales, debe tener al menos 21 años
        if (edad < 21) throw new IllegalArgumentException("Debe tener al menos 21 años para clase " +
clase);

        // Verifica si ya tuvo licencia profesional antes y la edad máxima para primera vez
        boolean tieneLicenciaProfesional = false;

        LicenciaActiva activa =
licenciaActivaRepository.findByTitular_NumeroDocumento(titular.getNumeroDocumento());
        if (activa != null && activa.getClases() != null && activa.getClases().matches(".*[CDE].*")) {
            tieneLicenciaProfesional = true;
        } else {
            var expiradas =
licenciaExpiradaRepository.findByTitular_NumeroDocumento(titular.getNumeroDocumento());
            tieneLicenciaProfesional = expiradas.stream().anyMatch(l -> l.getClases() != null &&
l.getClases().matches(".*[CDE].*"));
        }
        if (!tieneLicenciaProfesional && edad > 65)
            throw new IllegalArgumentException("Edad máxima para primera licencia profesional (" + clase +
") es 65 años.");

        // Verifica que tenga licencia B y al menos 1 año de antigüedad
        boolean tieneLicenciaB = false;
        LocalDate fechaEmisionB = null;

        LicenciaActiva licenciaB =
licenciaActivaRepository.findByTitular_NumeroDocumento(titular.getNumeroDocumento());
        if (licenciaB != null && licenciaB.getClases() != null && licenciaB.getClases().contains("B")) {
            tieneLicenciaB = true;
            fechaEmisionB = licenciaB.getFechaEmision();
        } else {
            var expiradas =
licenciaExpiradaRepository.findByTitular_NumeroDocumento(titular.getNumeroDocumento());
            for (var l : expiradas) {
                if (l.getClases() != null && l.getClases().contains("B")) {
                    tieneLicenciaB = true;
                    if (fechaEmisionB == null || l.getFechaEmision().isBefore(fechaEmisionB)) {
                        fechaEmisionB = l.getFechaEmision();
                    }
                }
            }
        }
    }
}

```

```

    }
  }
  if (!tieneLicenciaB)
    throw new IllegalArgumentException("Debe poseer licencia B para tramitar clase " + clase);
  if (fechaEmisionB == null || fechaEmisionB.isAfter(LocalDate.now().minusYears(1)))
    throw new IllegalArgumentException("Debe tener al menos 1 año de antigüedad con licencia B
para clase " + clase);
  } else {
    // Para clases no profesionales, debe tener al menos 17 años
    if (edad < 17) throw new IllegalArgumentException("Debe tener al menos 17 años para clase " +
clase);
  }
}
}
}

```

### TAREA 1.3: CREAR CONTROLADORES

Descripción				
<p>Implementar los controladores REST necesarios para gestionar las operaciones relacionadas con la emisión de licencias, para que el sistema pueda recibir y responder correctamente a las peticiones HTTP del frontend. Se debe:</p> <ul style="list-style-type: none"> <li>• Obtener titular por id.</li> <li>• Emitir una nueva licencia.</li> <li>• Obtener la información necesaria para completar el formulario de emisión.</li> <li>• Manejar errores y devolver códigos de estado HTTP.</li> </ul>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	2	3	14/06	15/06
Código				
<p><u>TitularController - Obtener titular por ID</u></p> <pre> @RestController @RequestMapping("/api/titulares") public class TitularController {     private final TitularService titularService;      public TitularController(TitularService titularService) {         this.titularService = titularService;     }      @GetMapping("/id/{numeroDocumento}")     public TitularClasesDTO getTitularByNumeroDocumento(@PathVariable String numeroDocumento) {         return titularService.getTitularConClases(numeroDocumento);     } } </pre> <p><u>LicenciaController - Emisión de licencias</u></p> <pre> @RestController @RequestMapping("/api/licencias") public class LicenciaController {     private final LicenciaService licenciaService; </pre>				



```

public LicenciaController(LicenciaService licenciaService) {
    this.licenciaService = licenciaService;
}

@PostMapping()
public ResponseEntity<LicenciaEmitidaDTO> emitirLicencia(@RequestBody LicenciaFormDTO
licenciaform) {
    LicenciaEmitidaDTO licenciaCreada = licenciaService.emitirLicencia(licenciaform);
    return new ResponseEntity<>(licenciaCreada, HttpStatus.OK);
}
}

```

#### Manejo de excepciones

@ControllerAdvice

```
public class GlobalExceptionHandler {
```

```
    @ExceptionHandler(TitularNoEncontradoException.class)
```

```
    public ResponseEntity<String> handleTitularNoEncontrado(TitularNoEncontradoException ex) {
        return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT_FOUND);
    }

```

```
    @ExceptionHandler(IllegalArgumentException.class)
```

```
    public ResponseEntity<String> handleIllegalArgumentException(IllegalArgumentException ex) {
        return new ResponseEntity<>(ex.getMessage(), HttpStatus.BAD_REQUEST);
    }

```

```
}
```

TAREA 1.4: DISEÑAR INTERFAZ**Descripción**

Desarrollar el formulario de emisión de licencias. Incluir los campos tipo y número de documento, apellido y nombre, fecha de nacimiento, dirección, clase/s solicitada/s (A-G), grupo sanguíneo y factor RH, donante de órganos (SI/NO), limitaciones u observaciones. Implementar los campos de entrada con sus respectivos tipos (text, date, select, radio buttons, textarea). Configurar el estado del formulario y los eventos de cambio. Asegurar que el formulario capture correctamente todos los datos del titular según los requerimientos.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	8	15	10/06	14/06

**Interfaz**
TAREA 1.5: CREAR ENTIDADES**Descripción**

Desarrollar las entidades necesarias para que el sistema pueda gestionar la emisión de licencias de conducir.

Entidades:

- Titular, con atributos: id, número documento, nombre, apellido, fecha de nacimiento, domicilio, grupo factor (grupo sanguíneo y factor RH combinados) y donante órganos.
- Licencia (clase abstracta), con atributos: número, observaciones, clases, fecha de emisión, fecha de vencimiento.
- LicenciaActiva, hereda de Licencia, representa la licencia vigente actual del titular.
- LicenciaExpirada, hereda de Licencia, representa las licencias que han vencido o sido reemplazadas.
- Usuario, como el empleado administrativo, con atributos: id, número documento, nombre, apellido,

email, rol (USER o ADMIN), contraseña.

#### Relaciones entre entidades:

- Un titular tiene una única licencia activa y una licencia activa pertenece a un único titular (uno a uno bidireccional).
- Un titular puede tener múltiples licencias expiradas y cada licencia expirada pertenece a un único titular (uno a muchos bidireccional).
- Un usuario puede tramitar muchas licencias y cada licencia tiene un solo usuario que la tramitó (uno a muchos desde Usuario, muchos a uno desde Licencia).

#### Consideraciones:

- Las clases de licencia (A, B, C, D, E, F, G) se almacenan como string en el atributo clases de la licencia.
- Cada licencia (activa o expirada) tiene sus propias fechas de emisión y vencimiento.
- Cuando una licencia vence, se convierte en LicenciaExpirada y se puede crear una nueva LicenciaActiva.
- El titular mantiene un historial de todas sus licencias a través de la relación con LicenciaExpirada.
- Los números de documento son únicos tanto para titulares como para usuarios.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	6	5	05/06	07/06

#### Código

##### Entidad Titular:

@Entity

@Getter

@Setter

public class Titular {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(unique = true)

private String numeroDocumento;

private String nombre;

private String apellido;

@DateTimeFormat(pattern = "yyyy-MM-dd")

private LocalDate fechaNacimiento;

private String domicilio;

private String grupoFactor;

private boolean donanteOrganos;

@OneToOne(mappedBy = "titular", cascade = CascadeType.ALL)

private LicenciaActiva licenciaActiva;

@OneToMany(mappedBy = "titular", cascade = CascadeType.ALL)

private List<LicenciaExpirada> licenciasExpiradas;

public int calcularEdad() {

LocalDate hoy = LocalDate.now();

int edad = hoy.getYear() - this.fechaNacimiento.getYear();

```

        if (hoy.getMonthValue() < this.fechaNacimiento.getMonthValue() ||
            (hoy.getMonthValue() == this.fechaNacimiento.getMonthValue() &&
             hoy.getDayOfMonth() < this.fechaNacimiento.getDayOfMonth())) {
            edad--;
        }
        return edad;
    }
}

```

#### Entidad Licencia:

```

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Licencia {
    @Id @GeneratedValue
    private Long numero;
    private String observaciones;
    private String clases;
    private LocalDate fechaEmision;
    private LocalDate fechaVencimiento;

    @ManyToOne
    @JoinColumn(name = "usuario_id")
    private Usuario usuario;
}

```

#### Entidad LicenciaActiva:

```

@Entity
@Getter
@Setter
public class LicenciaActiva extends Licencia {
    @OneToOne
    @JoinColumn(name = "titular_id")
    private Titular titular;
}

```

#### Entidad LicenciaExpirada:

```

@Entity
@Getter
@Setter
public class LicenciaExpirada extends Licencia {
    @ManyToOne
    @JoinColumn(name = "titular_id")
    private Titular titular;
}

```

#### Entidad Usuario:

```

@Entity
@Getter
@Setter
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true)
    private String numeroDocumento;
}

```

```

private String nombre;
private String apellido;
private String email;

@OneToMany(mappedBy = "usuario")
private List<Licencia> licenciasTramitadas;
}

```

### TAREA 1.6: CREAR REPOSITORIOS

#### Descripción

Implementar los repositorios para el acceso a datos con Spring Data JPA, para que el sistema pueda gestionar las entidades requeridas para la emisión de licencias. Se debe:

- Implementar repositorios para las entidades Titular, LicenciaActiva, LicenciaExpirada y Usuario.
- Verificar si un titular ya posee una licencia vigente.
- Buscar titulares existentes por número de documento.
- Buscar usuarios por número de documento para identificar quién tramita la licencia.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	1	06/06	06/06

#### Código

##### TitularRepository

```

public interface TitularRepository extends JpaRepository<Titular, Long> {
    Titular findByNumeroDocumento(String numeroDocumento);
}

```

##### LicenciaActivaRepository

```

public interface LicenciaActivaRepository extends JpaRepository<LicenciaActiva, Long> {
    LicenciaActiva findByTitular_NumeroDocumento(String titular_numeroDocumento);
}

```

##### LicenciaExpiradaRepository

```

public interface LicenciaExpiradaRepository extends JpaRepository<LicenciaExpirada, Long> {
    List<LicenciaExpirada> findByTitular_NumeroDocumento(String titular_numeroDocumento);
}

```

##### UsuarioRepository

```

public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
    Usuario findByNumeroDocumento(String numeroDocumento);
}

```

TAREA 1.7: CREAR SERVICIOS**Descripción**

Implementar la lógica de negocio. Incluir métodos para validar datos del titular, verificar requisitos por clase de licencia, generar número de licencia único, y coordinar la creación de titular y licencia.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	6	8	10/06	11/06

**Código**

@Service

```
public class LicenciaServiceImpl implements LicenciaService {
```

```
    private final LicenciaActivaRepository licenciaActivaRepository;
    private final LicenciaExpiradaRepository licenciaExpiradaRepository;
    private final TitularRepository titularRepository;
    private final UsuarioRepository usuarioRepository;
    private final LicenciaActivaMapper licenciaActivaMapper;
    private final LicenciaEmitidaMapper licenciaEmitidaMapper;
    private final LicenciaExpiradaMapper licenciaExpiradaMapper;
```

```
    private static final double GASTOS_ADMINISTRATIVOS = 8.0;
```

```
    public LicenciaServiceImpl(
        LicenciaActivaRepository licenciaActivaRepository,
        LicenciaExpiradaRepository licenciaExpiradaRepository,
        TitularRepository titularRepository,
        UsuarioRepository usuarioRepository,
        LicenciaActivaMapper licenciaActivaMapper,
        LicenciaEmitidaMapper licenciaEmitidaMapper,
        LicenciaExpiradaMapper licenciaExpiradaMapper) {
        this.licenciaActivaRepository = licenciaActivaRepository;
        this.licenciaExpiradaRepository = licenciaExpiradaRepository;
        this.titularRepository = titularRepository;
        this.usuarioRepository = usuarioRepository;
        this.licenciaActivaMapper = licenciaActivaMapper;
        this.licenciaEmitidaMapper = licenciaEmitidaMapper;
        this.licenciaExpiradaMapper = licenciaExpiradaMapper;
    }
```

@Transactional

@Override

```
public LicenciaEmitidaDTO emitirLicencia(LicenciaFormDTO licenciaFormDTO) {
    validarFormularioLicencia(licenciaFormDTO); // Valida los datos del formulario
```

```
    String documentoUsuario = obtenerDocumentoUsuarioAutenticado(); // Obtiene el documento del usuario autenticado
```

```
    // Busca el titular por documento, lanza excepción si no existe
```

```
    Titular titular = titularRepository.findByNumeroDocumento(licenciaFormDTO.getDocumentoTitular());
```

```
    if (titular == null)
```

```
        throw new TitularNoEncontradoException("No se encontró un titular con el número de documento: " +
            licenciaFormDTO.getDocumentoTitular());
```

```
// Busca el usuario por documento, lanza excepción si no existe
Usuario usuario = usuarioRepository.findByNumeroDocumento(documentoUsuario);
if (usuario == null)
    throw new UsuarioNoEncontradoException("No se encontró un usuario con el número de documento:
" + documentoUsuario);

LocalDate fechaEmision = LocalDate.now(); // Fecha de emisión actual
LocalDate fechaVencimiento = calcularFechaVencimiento(titular); // Calcula la fecha de vencimiento

// Expira la licencia anterior (si existe) y crea la nueva licencia activa
LicenciaActiva nuevaLicencia = expirarYCrearNuevaLicenciaActiva(titular, usuario, licenciaFormDTO,
fechaEmision, fechaVencimiento);

LicenciaEmitidaDTO dto = licenciaEmitidaMapper.entityToDto(nuevaLicencia);
dto.setDocumentoTitular(titular.getNumeroDocumento());
dto.setDocumentoUsuario(usuario.getNumeroDocumento());
return dto;
}
```

## HISTORIA 2: CALCULAR VIGENCIA DE LICENCIA

### TAREA 2.1: IMPLEMENTAR CÁLCULO DE VIGENCIA

#### Descripción

El día y mes de vencimiento deben coincidir con la fecha de nacimiento del titular. La fecha de inicio debe ser la fecha actual del sistema. La vigencia de la licencia se calcula según las reglas: menores de 21 años (1 año primera vez, 3 años siguientes), hasta 46 años (5 años), hasta 60 años (4 años), hasta 70 años (3 años), mayores de 70 años (1 año).

Si la fecha de trámite se encuentra a más de un mes antes del cumpleaños del titular se descuenta un año de la vigencia. Esto significa que existe una ventana de tiempo sin penalización que abarca desde un mes antes hasta la fecha de nacimiento del titular. Fuera de esta ventana, la vigencia se reduce en un año completo.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	4	4	17/06	17/06

#### Código

```
private int calcularVigencia(Titular titular) {
    int edad = titular.calcularEdad(); // Obtiene la edad del titular

    // Determina si el titular tuvo alguna licencia (activa o expirada)
    boolean tuvoLicencia = false;
    if (licenciaActivaRepository.findByTitular_NumeroDocumento(titular.getNumeroDocumento()) != null) {
        tuvoLicencia = true;
    } else {
        if
        (!licenciaExpiradaRepository.findByTitular_NumeroDocumento(titular.getNumeroDocumento()).isEmpty()) {
            tuvoLicencia = true;
        }
    }

    // Define la vigencia según la edad y si tuvo licencia previa
```

```

if (edad < 21) {
    return tuvoLicencia ? 3 : 1; // Menor de 21: 3 años si tuvo licencia, 1 si no
} else if (edad <= 46) {
    return 5; // 21 a 46 años: 5 años
} else if (edad <= 60) {
    return 4; // 47 a 60 años: 4 años
} else if (edad <= 70) {
    return 3; // 61 a 70 años: 3 años
} else {
    return 1; // Más de 70 años: 1 año
}
}

```

## TAREA 2.2: REALIZAR TESTING DE VIGENCIA

Descripción				
<p>Crear pruebas unitarias para el cálculo de vigencia. Incluir casos de prueba para cada rango de edad, verificación de fechas de vencimiento correctas (día y mes coincidentes con nacimiento), y casos límite en los cambios de rango de edad. Validar que la fecha de inicio siempre sea la fecha actual del sistema.</p>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	3	3	19/06	19/06
Código				
<pre> @Test void testVigenciaMenor21_sinLicencia() throws Exception {     // Menor de 21 años, sin licencia previa: vigencia debe ser 1 año     LocalDate hoy = LocalDate.now();     LocalDate nacimiento = hoy.minusYears(20);     Titular t = buildTitular(nacimiento);      when(licenciaActivaRepository.findByTitular_NumeroDocumento(any())).thenReturn(null);      when(licenciaExpiradaRepository.findByTitular_NumeroDocumento(any())).thenReturn(Collections.emptyList());      LocalDate vencimiento = invocarCalcularFechaVencimiento(t);      assertEquals(nacimiento.getMonth(), vencimiento.getMonth()); // El mes coincide con el nacimiento     assertEquals(nacimiento.getDayOfMonth(), vencimiento.getDayOfMonth()); // El día también     assertEquals(1, vencimiento.getYear() - hoy.getYear()); // Vigencia de 1 año     assertTrue(vencimiento.isAfter(hoy)); // La fecha de vencimiento es futura }  @Test void testVigenciaMenor21_conLicenciaPrevia() throws Exception {     // Menor de 21 años, con licencia previa: vigencia debe ser 3 años     LocalDate hoy = LocalDate.now();     LocalDate nacimiento = hoy.minusYears(19);     Titular t = buildTitular(nacimiento); </pre>				



```

when(licenciaActivaRepository.findByTitular_NumeroDocumento(any())).thenReturn(mock(LicenciaActiva.class));

    LocalDate vencimiento = invocarCalcularFechaVencimiento(t);

    assertEquals(nacimiento.getMonth(), vencimiento.getMonth());
    assertEquals(nacimiento.getDayOfMonth(), vencimiento.getDayOfMonth());
    assertEquals(3, vencimiento.getYear() - hoy.getYear()); // Vigencia de 3 años
    assertTrue(vencimiento.isAfter(hoy));
}

@Test
void testVigencia21a46() throws Exception {
    // Entre 21 y 46 años: vigencia debe ser 5 años
    LocalDate hoy = LocalDate.now();
    LocalDate nacimiento = hoy.minusYears(30);
    Titular t = buildTitular(nacimiento);

    when(licenciaActivaRepository.findByTitular_NumeroDocumento(any())).thenReturn(null);

    when(licenciaExpiradaRepository.findByTitular_NumeroDocumento(any())).thenReturn(Collections.emptyList());

    LocalDate vencimiento = invocarCalcularFechaVencimiento(t);

    assertEquals(nacimiento.getMonth(), vencimiento.getMonth());
    assertEquals(nacimiento.getDayOfMonth(), vencimiento.getDayOfMonth());
    assertEquals(5, vencimiento.getYear() - hoy.getYear()); // Vigencia de 5 años
    assertTrue(vencimiento.isAfter(hoy));
}

@Test
void testVigenciaMayor70() throws Exception {
    // Mayor de 70 años: vigencia debe ser 1 año
    LocalDate hoy = LocalDate.now();
    LocalDate nacimiento = hoy.minusYears(75);
    Titular t = buildTitular(nacimiento);

    when(licenciaActivaRepository.findByTitular_NumeroDocumento(any())).thenReturn(null);

    when(licenciaExpiradaRepository.findByTitular_NumeroDocumento(any())).thenReturn(Collections.emptyList());

    LocalDate vencimiento = invocarCalcularFechaVencimiento(t);

    assertEquals(nacimiento.getMonth(), vencimiento.getMonth());
    assertEquals(nacimiento.getDayOfMonth(), vencimiento.getDayOfMonth());
    assertEquals(1, vencimiento.getYear() - hoy.getYear()); // Vigencia de 1 año
    assertTrue(vencimiento.isAfter(hoy));
}

```

**HISTORIA 3: CALCULAR COSTO DE LICENCIA****TAREA 3.1: IMPLEMENTAR CÁLCULO DE COSTO****Descripción**

Desarrollar el sistema de cálculo de costos basado en la tabla de precios por clase y vigencia. Implementar la matriz de costos completa para la emisión de licencias. La matriz de costos debe considerar las siguientes clases y vigencias:

- Clase A/B/G: 5 años: \$ 40, 4 años: \$ 30, 3 años: \$ 25, 1 año: \$ 20
- Clase C: 5 años: \$ 47, 4 años: \$ 35, 3 años: \$ 30, 1 año: \$ 23
- Clase D: 5 años: \$ 50, 4 años: \$ 40, 3 años: \$ 35, 1 año: \$ 28
- Clase E: 5 años: \$ 59, 4 años: \$ 44, 3 años: \$ 39, 1 año: \$ 29
- Clase F: 5 años: \$ 40, 4 años: \$ 30, 3 años: \$ 25, 1 año: \$ 20

Además, el sistema debe agregar automáticamente \$8 en concepto de gastos administrativos a todos los cálculos de costo. Se debe asegurar que los costos resultantes no puedan ser modificados directamente por un operador.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	2	1	17/06	17/06

**Código**

```
// Calcula el costo total de emisión según las clases y la vigencia del titular
private double calcularCostoEmision(String clases, Titular titular) {
    int vigencia = calcularVigencia(titular); // Determina la vigencia según la edad y antecedentes
    double total = 0.0;

    for (String clase : clases.trim().split("\\s+")) {
        total += costoPorClaseYVigencia(clase, vigencia);
    }
    return total;
}

// Devuelve el costo de una clase específica según la vigencia
private double costoPorClaseYVigencia(String clase, int vigencia) {
    switch (clase) {
        case "A":
        case "B":
        case "G":
        case "F":
            switch (vigencia) {
                case 5: return 40.0;
                case 4: return 30.0;
                case 3: return 25.0;
                case 1: return 20.0;
            }
            break;
        case "C":
            switch (vigencia) {
                case 5: return 47.0;
                case 4: return 35.0;
                case 3: return 30.0;
                case 1: return 23.0;
            }
    }
}
```

```

        break;
    case "D":
        switch (vigencia) {
            case 5: return 50.0;
            case 4: return 40.0;
            case 3: return 35.0;
            case 1: return 28.0;
        }
        break;
    case "E":
        switch (vigencia) {
            case 5: return 59.0;
            case 4: return 44.0;
            case 3: return 39.0;
            case 1: return 29.0;
        }
        break;
    }
    throw new IllegalArgumentException("Clase o vigencia no válida");
}

```

### TAREA 3.2: REALIZAR TESTING DE CÁLCULO DE COSTO

Descripción				
<p>Crear pruebas unitarias para verificar el cálculo correcto de costos para todas las combinaciones de clase y vigencia. Incluir verificación de que los gastos administrativos se agreguen correctamente, y que el sistema no permita modificaciones manuales de precios. Probar casos límite y validar la integridad de la tabla de precios.</p>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	2	2	19/06	19/06
Código				
<pre> @Test void testCostosPorClaseYVigencia() throws Exception {     // Simula un titular de 30 años (vigencia 5)     Titular titular = new Titular();     titular.setFechaNacimiento(LocalDate.now().minusYears(30));     // Tabla de precios esperada     assertEquals(40.0, costoPorClaseYVigencia("A", 5));     assertEquals(30.0, costoPorClaseYVigencia("B", 4));     assertEquals(25.0, costoPorClaseYVigencia("G", 3));     assertEquals(20.0, costoPorClaseYVigencia("F", 1));     assertEquals(47.0, costoPorClaseYVigencia("C", 5));     assertEquals(35.0, costoPorClaseYVigencia("C", 4));     assertEquals(30.0, costoPorClaseYVigencia("C", 3));     assertEquals(23.0, costoPorClaseYVigencia("C", 1));     assertEquals(50.0, costoPorClaseYVigencia("D", 5));     assertEquals(40.0, costoPorClaseYVigencia("D", 4));     assertEquals(35.0, costoPorClaseYVigencia("D", 3));     assertEquals(28.0, costoPorClaseYVigencia("D", 1));     assertEquals(59.0, costoPorClaseYVigencia("E", 5)); </pre>				

```
assertEquals(44.0, costoPorClaseYVigencia("E", 4));
assertEquals(39.0, costoPorClaseYVigencia("E", 3));
assertEquals(29.0, costoPorClaseYVigencia("E", 1));
}

@Test
void testCostosInvalidosLanzanExcepcion() throws Exception {
    assertThrows(Exception.class, () -> costoPorClaseYVigencia("Z", 5));
    assertThrows(Exception.class, () -> costoPorClaseYVigencia("A", 2));
    assertThrows(Exception.class, () -> costoPorClaseYVigencia("C", 2));
}


@Test
void testCostoEmisionMultiplesClases() throws Exception {
    Titular titular = new Titular();
    titular.setFechaNacimiento(LocalDate.now().minusYears(30)); // vigencia 5
    double costo = calcularCosto("B C", titular);
    assertEquals(40.0 + 47.0, costo);
}
```

## HISTORIA 4: IMPRIMIR LICENCIA

## TAREA 4.1: LLAMAR A INTERFAZ DE IMPRESIÓN DEL NAVEGADOR

Descripción				
Implementar la funcionalidad de impresión utilizando la API de impresión del navegador. Configurar opciones de impresión apropiadas para licencias y comprobantes, manejar diferentes tamaños de papel, y asegurar que la calidad de impresión sea adecuada. Incluir vista previa antes de imprimir.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Kevin Brunas	3	4	17/06	17/06

Interfaz	
<div><div>Licencia Nacional de Conducir Santa Fe - Santa Fe</div><div><div><div>N° Licencia: 21000000</div><div>Apellido: Díaz</div><div>Domicilio: Avenida Siempre Viva 1</div><div>Otorgamiento: 28 de junio de 2023</div><div>Donante de Órganos: Si</div><div>Observaciones: Apretaba mucho el embrague</div></div><div><div>Clases: B</div><div>Nombre: Carla</div><div>Fecha de Nac.: 28 de junio de 2004</div><div>Vencimiento: 28 de junio de 2028</div><div>Grupo y Factor: O+</div></div></div><div><div>SEGURO VIAL</div><div>Ministerio de Transporte República Argentina</div></div></div>	<div><div>Imprimir</div><div>2 hojas de papel</div><div><div>Destino</div><div> Microsoft Print to PDF</div></div><div><div>Páginas</div><div>Todos</div></div><div><div>Diseño</div><div>Vertical</div></div><div><div>Color</div><div>Color</div></div><div>Más opciones de configuración</div><div><div>Imprimir</div><div>Cancelar</div></div></div>

**Comprobante de Pago****Nombre:** Juan Pérez**DNI:** 12345678**Tipo de trámite:** Licencia Clase A**Detalle de Costos:**

- Trámite: \$40
- Gastos administrativos: \$8

**Total:** \$48**Fecha de emisión:** 03 de julio de 2025

Conserve este comprobante para presentarlo al momento del pago.


payment/receipt/12345678

1/2

**Imprimir**

2 hojas de papel

Destino

 Microsoft Print to PDF ▾

Páginas

Todos ▾

Diseño

Vertical ▾

Color

Color ▾

Más opciones de configuración ▾

Imprimir

Cancelar

## TAREA 4.2: DISEÑAR LICENCIA PARA IMPRIMIR

## Descripción

Diseñar el comprobante de pago que se entregará al interesado para abonar en caja. Debe incluir: datos del titular, tipo de trámite, costo detallado (incluyendo gastos administrativos), número de comprobante único, fecha de emisión.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Kevin Brunas	8	6	15/06	16/06

## Interfaz

**Menú**

- Nueva Licencia
- Renovar Licencia
- Licencias Vigentes
- Licencias a Expirar
- Registrar Titular
- Modificar Titular
- Registrar Usuario
- Modificar Usuario

**Licencia Nacional de Conducir**  
Santa Fe - Santa Fe

**Nº Licencia:** 21000000  
**Apellido:** Diaz  
**Domicilio:** Avenida Siempre Viva 1  
**Otorgamiento:** 28 de junio de 2023  
**Donante de Órganos:** Si  
**Observaciones:** Apretaba mucho el embrague

**Clases:** B  
**Nombre:** Carla  
**Fecha de Nac.:** 28 de junio de 2004  
**Vencimiento:** 28 de junio de 2028  
**Grupo y Factor:** O+

SEGUIDAD VIAL

Ministerio de Transporte  
República Argentina

Imprimir licencia

Cerrar sesión

## TAREA 4.3: REALIZAR TESTING EN IMPRIMIR LICENCIA

## Descripción

Crear pruebas para verificar la generación de documentos imprimibles, validar que todos los datos coincidan con los datos del titular, se muestren correctamente y que el formato sea consistente.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Kevin Brunas	5	5	18/06	18/06

## Interfaz

Se realizó el testing de manera manual comprobando nombres con el mayor largo admitido por el formulario, se probó con caracteres invisibles, valores no posibles y demás caracteres perjudiciales.

## Licencia Nacional de Conducir

Santa Fe - Santa Fe



N° Licencia: 11111111

Apellido: Lorem ipsum dolor sit amet, consectetur adipiscing

Domicilio: Lorem ipsum dolor sit amet, consectetur adipiscing

Otorgamiento: 04 de julio de 2025

Donante de Órganos: Si

Observaciones: Lorem ipsum dolor sit amet, consectetur adipiscing elit nam.

Clases: A B F

Nombre: Lorem ipsum dolor sit amet, consectetur adipiscing

Fecha de Nac.: 11 de noviembre de 2001

Vencimiento: 11 de noviembre de 2029

Grupo y Factor: A+



SEGURIDAD VIAL

Ministerio de Transporte  
República Argentina

Imprimir licencia

## TAREA 4.4: DISEÑAR COMPROBANTE PARA IMPRIMIR

## Descripción

Diseñar el comprobante de pago que se entregará al interesado para abonar en caja. Debe incluir: datos del titular, tipo de trámite, costo detallado (incluyendo gastos administrativos), número de comprobante único, fecha de emisión.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Kevin Brunas	8	5	16/06	16/06

## Interfaz

## Menú

Nueva Licencia  
Renovar Licencia  
Licencias Vigentes  
Licencias a Expirar  
Registrar Titular  
Modificar Titular  
Registrar Usuario  
Modificar Usuario

Cerrar sesión

## Comprobante de Pago

Nombre: Juan Pérez

DNI: 12345678

Tipo de trámite: Licencia Clase A

## Detalle de Costos:

- Trámite: \$40
- Gastos administrativos: \$8

Total: \$48

Fecha de emisión: 04 de julio de 2025

Conserve este comprobante para presentarlo al momento del pago.

Imprimir comprobante



**HISTORIA 5: RENOVAR LICENCIA****TAREA 5.1: REALIZAR TESTING DE RENOVAR LICENCIA**

Descripción				
<p>Crear pruebas unitarias para incluir casos de renovación por vencimiento, renovación por modificación de datos, validaciones de datos, cálculo de costos y fechas, y persistencia correcta. Probar escenarios de error y casos límite.</p>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	3	2	18/06	19/06
Código				
<p><u>Tests de validación de motivos y precondiciones</u></p> <pre>@Test void testRenovarLicencia_conMotivoInvalido_lanzaExcepcion() {     LicenciaFormDTO form = buildForm("12345678", "B");     Exception ex = assertThrows(IllegalArgumentException.class, () -&gt;         licenciaService.renovarLicencia(form, "extravio")     );     assertTrue(ex.getMessage().contains("'vencimiento' o 'modificacion'")); }</pre> <pre>@Test void testRenovarLicencia_titularNoExiste_lanzaExcepcion() {     LicenciaFormDTO form = buildForm("12345678", "B");     when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(null);     Exception ex = assertThrows(RuntimeException.class, () -&gt;         licenciaService.renovarLicencia(form, "vencimiento")     );     assertTrue(ex.getMessage().contains("No se encontró un titular")); }</pre> <p><u>Tests de validación de licencia actual</u></p> <pre>@Test void testRenovarLicencia_licenciaActivaNoExiste_lanzaExcepcion() {     LicenciaFormDTO form = buildForm("12345678", "B");     Titular titular = new Titular();     titular.setNumeroDocumento("12345678");     when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);     Exception ex = assertThrows(RuntimeException.class, () -&gt;         licenciaService.renovarLicencia(form, "vencimiento")     );     assertTrue(ex.getMessage().contains("No hay licencia activa")); }</pre> <pre>@Test void testRenovarLicencia_licenciaNoVencida_lanzaExcepcion() {     LicenciaFormDTO form = buildForm("12345678", "B");     Titular titular = new Titular();     titular.setNumeroDocumento("12345678");     LicenciaActiva activa = new LicenciaActiva();     activa.setFechaVencimiento(LocalDate.now().plusDays(5));     titular.setLicenciaActiva(activa); }</pre>				

```

when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);
Exception ex = assertThrows(IllegalArgumentException.class, () ->
    licenciaService.renovarLicencia(form, "vencimiento")
);
assertTrue(ex.getMessage().contains("aún no está vencida"));
}

```

#### Test de renovación exitosa

@Test

```

void testRenovarLicencia_ok() {
    LicenciaFormDTO form = buildForm("12345678", "B");
    Titular titular = new Titular();
    titular.setNumeroDocumento("12345678");
    titular.setFechaNacimiento(LocalDate.now().minusYears(30));

    Usuario usuario = new Usuario();
    usuario.setNumeroDocumento("11999888");

    LicenciaActiva activa = new LicenciaActiva();
    activa.setFechaVencimiento(LocalDate.now().minusDays(1));
    activa.setUsuario(usuario);
    activa.setClases("B");
    activa.setFechaEmision(LocalDate.now().minusYears(5));
    activa.setObservaciones("Antigua");

    titular.setLicenciaActiva(activa);

    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(titular);
    when(usuarioRepository.findByNumeroDocumento(any())).thenReturn(usuario);
    when(licenciaEmitidaMapper.entityToDto(any())).thenReturn(new LicenciaEmitidaDTO());

    assertDoesNotThrow(() -> licenciaService.renovarLicencia(form, "vencimiento"));
}

```

TAREA 5.2: DISEÑAR INTERFAZ PARA RENOVAR LICENCIA**Descripción**

Crear la interfaz para el proceso de renovación de licencias. Incluir un campo para buscar al titular, mostrar todos sus atributos y los de su licencia actual. Incluir una lista desplegable para seleccionar las clases a renovar. La interfaz debe distinguir entre renovación por vencimiento y renovación por modificación de datos. Redirigir a la ventana de impresión de licencia.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	4	3	17/06	18/06

**Interfaz**
TAREA 5.3: CALCULAR EL COSTO DE RENOVACIÓN**Descripción**

Implementar la lógica para calcular el costo de renovación. Para renovaciones por vencimiento, aplicar costos según nueva vigencia. Para renovaciones por modificación de datos, aplicar el mismo costo que la emisión original. Reutilizar la lógica de cálculo de costos ya implementada.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	2	1	16/06	16/06

**Código**

```
@Override
@Transactional
public LicenciaEmitidaDTO renovarLicencia(LicenciaFormDTO licenciaFormDTO, String motivo) {
    // Validaciones y lógica específica de renovación
    // Reutilización de la lógica de cálculo de costos
}
```

```
double costoRenovacion = calcularCostoEmision(licenciaFormDTO.getClases(), titular);
// Creación de nueva licencia
return dto;
}
```

#### TAREA 5.4: PERSISTIR LOS DATOS DE LA LICENCIA RENOVADA

##### Descripción

Desarrollar la lógica para actualizar la licencia en la base de datos. Mantener historial de la licencia anterior, actualizar datos de la licencia actual, y registrar la transacción de renovación con usuario y fecha correspondientes.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	3	2	17/06	18/06

##### Código

###### Método de Persistencia y Actualización:

```
private LicenciaActiva expirarYCrearNuevaLicenciaActiva(Titular titular, Usuario usuario, LicenciaFormDTO
licenciaFormDTO, LocalDate fechaEmision, LocalDate fechaVencimiento) {
    LicenciaActiva licenciaAnterior = titular.getLicenciaActiva();
    if (licenciaAnterior != null) {
        LicenciaExpirada expirada = new LicenciaExpirada();
        expirada.setTitular(titular);
        expirada.setUsuario(licenciaAnterior.getUsuario());
        expirada.setObservaciones(licenciaAnterior.getObservaciones());
        expirada.setClases(licenciaAnterior.getClases());
        expirada.setFechaEmision(licenciaAnterior.getFechaEmision());
        expirada.setFechaVencimiento(licenciaAnterior.getFechaVencimiento());

        titular.setLicenciaActiva(null);
        licenciaAnterior.setTitular(null);

        licenciaExpiradaRepository.save(expirada);
        licenciaActivaRepository.delete(licenciaAnterior);
        licenciaActivaRepository.flush();
    }

    LicenciaActiva nuevaLicencia = new LicenciaActiva();
    nuevaLicencia.setTitular(titular);
    nuevaLicencia.setUsuario(usuario);
    nuevaLicencia.setObservaciones(licenciaFormDTO.getObservaciones());
    nuevaLicencia.setClases(licenciaFormDTO.getClases());
    nuevaLicencia.setFechaEmision(fechaEmision);
    nuevaLicencia.setFechaVencimiento(fechaVencimiento);

    titular.setLicenciaActiva(nuevaLicencia);

    licenciaActivaRepository.save(nuevaLicencia);
    titularRepository.save(titular);

    return nuevaLicencia;
}
```

### TAREA 5.5: OBTENER LA NUEVA FECHA DE EXPIRACIÓN

Descripción				
Reutilizar la lógica de cálculo de vigencia, considerando la edad actual del titular. Asegurar que la nueva fecha siga las mismas reglas que la emisión original.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	1	17/06	17/06
Código				
<pre>// Al emitir o renovar, se reutiliza la lógica de vigencia: LocalDate fechaVencimiento = calcularFechaVencimiento(titular); // Se usa la fecha calculada para crear la nueva licencia LicenciaActiva nuevaLicencia = expirarYCrearNuevaLicenciaActiva(titular, usuario, licenciaFormDTO, fechaEmision, fechaVencimiento);</pre>				

### TAREA 5.6: VALIDAR Y MODIFICAR LOS DATOS DE LA LICENCIA SELECCIONADA PARA RENOVAR

Descripción				
Implementar las validaciones necesarias para el proceso de renovación. Verificar que la licencia pueda ser renovada, validar nuevos datos si se modifican, y aplicar las mismas reglas de negocio que en la emisión. Manejar casos especiales como cambios de clase de licencia.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	3	4	18/06	18/06
Código				
<pre>@Override @Transactional public LicenciaEmitidaDTO renovarLicencia(LicenciaFormDTO licenciaFormDTO, String motivo) {     // Motivo de renovación     if (!"vencimiento".equalsIgnoreCase(motivo) &amp;&amp; !"modificacion".equalsIgnoreCase(motivo)) {         throw new IllegalArgumentException("El motivo debe ser 'vencimiento' o 'modificacion'");     }      // Motivo de renovación modificación no se puede implementar completamente en este Sprint     if ("vencimiento".equalsIgnoreCase(motivo)) {         throw new IllegalArgumentException("El motivo 'modificacion' no corresponde a este Sprint");     }      // Validación del titular y licencia existente     Titular titular = titularRepository.findByNumeroDocumento(licenciaFormDTO.getDocumentoTitular());     if (titular == null) {         throw new TitularNoEncontradoException("No se encontró un titular con el número de documento: " + licenciaFormDTO.getDocumentoTitular());     }      LicenciaActiva licenciaAnterior = titular.getLicenciaActiva();</pre>				

```

if (licenciaAnterior == null) {
    throw new LicenciaNoEncontradaException("No hay licencia activa para renovar");
}

// Renovación por vencimiento
if ("vencimiento".equalsIgnoreCase(motivo) &&
licenciaAnterior.getFechaVencimiento().isAfter(LocalDate.now())) {
    throw new IllegalArgumentException("La licencia aún no está vencida, no se puede renovar");
}

// Validación del formulario y reglas de negocio
validarFormularioLicencia(licenciaFormDTO);

// Proceso de renovación
String documentoUsuario = "11999888";
Usuario usuario = usuarioRepository.findByNumeroDocumento(documentoUsuario);
if (usuario == null) {
    throw new UsuarioNoEncontradoException("No se encontró un usuario con el número de documento:
" + documentoUsuario);
}

LocalDate fechaEmision = LocalDate.now();
LocalDate fechaVencimiento = calcularFechaVencimiento(titular);

LicenciaActiva nuevaLicencia = expirarYCrearNuevaLicenciaActiva(titular, usuario, licenciaFormDTO,
fechaEmision, fechaVencimiento);

LicenciaEmitidaDTO dto = licenciaEmitidaMapper.entityToDto(nuevaLicencia);
dto.setDocumentoTitular(titular.getNumeroDocumento());
dto.setDocumentoUsuario(usuario.getNumeroDocumento());
return dto;
}

```

## HISTORIA 6: LISTADO DE LICENCIAS QUE HAN EXPIRADO

### TAREA 6.1: OBTENER EL LISTADO DE LICENCIAS EXPIRADAS

Descripción				
Implementar consulta a la base de datos para obtener las licencias expiradas, permitiendo filtrar por rango de fechas y mostrando los resultados desde el más reciente al más antiguo.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	3	6	17/06	18/06
Código				
<pre> @Override public LicenciasVencidasDTO obtenerLicenciasVencidasEntre(LocalDate desde, LocalDate hasta) {     // Obtiene las licencias expiradas en el rango de fechas y las mapea a DTOs     List&lt;LicenciaExpiradaDTO&gt; expiradas = licenciaExpiradaRepository         .findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(desde, hasta)         .stream()         .map(licenciaExpiradaMapper::entityToDto) </pre>				

```
.collect(Collectors.toList());
```

```
// Obtiene las licencias activas que vencen en el rango de fechas y las mapea a DTOs
List<LicenciaActivaDTO> activasVencidas = licenciaActivaRepository
    .findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(desde, hasta)
    .stream()
    .map(licenciaActivaMapper::entityToDto)
    .collect(Collectors.toList());
```

```
// Crea el DTO de resultado y asigna las listas obtenidas
LicenciasVencidasDTO resultado = new LicenciasVencidasDTO();
resultado.setExpiradas(expiradas);
resultado.setActivasVencidas(activasVencidas);
```

```
// Devuelve el DTO con la información de licencias vencidas
return resultado;
```

```
}
```

## TAREA 6.2: DISEÑAR INTERFAZ DE USUARIO

### Descripción

Crear la interfaz para mostrar el listado de licencias expiradas. Se muestran todas las licencias expiradas o a expirar en cierto rango, que no debe ser mayor a una semana. Se muestra una lista de cada una de las licencias, junto a sus datos principales (nombre titular, fecha de creación, fecha expiración), y funcionalidad de ordenamiento por fecha de expiración.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	4	6	18/06	19/06

### Interfaz

**Menú**

- Nueva Licencia
- Renovar Licencia
- Licencias Vigentes
- Licencias a Expirar
- Registrar Titular
- Modificar Titular
- Registrar Usuario
- Modificar Usuario

**Licencias a Expirar**

Desde  Hasta

Titular	Clase/s	Fecha de emisión	Fecha de vencimiento ▼
12345678	D	2016-03-10	2021-03-10

## TAREA 6.3: REALIZAR TESTING DE LICENCIAS EXPIRADAS

Descripción				
Desarrollar pruebas para verificar la consulta correcta de licencias expiradas, funcionamiento del filtrado por fechas, ordenamiento correcto (de más reciente a más antigua expiración).				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	3	4	18/06	19/06
Código				
<pre> @Test void testObtenerLicenciasExpiradas_filtradoYOrden() {     LocalDate hoy = LocalDate.now();     LicenciaExpirada lic1 = new LicenciaExpirada();     lic1.setFechaVencimiento(hoy.minusDays(2));     Titular t1 = new Titular();     t1.setNumeroDocumento("1");     lic1.setTitular(t1);      LicenciaExpirada lic2 = new LicenciaExpirada();     lic2.setFechaVencimiento(hoy.minusDays(1));     Titular t2 = new Titular();     t2.setNumeroDocumento("2");     lic2.setTitular(t2);      LicenciaExpirada lic3 = new LicenciaExpirada();     lic3.setFechaVencimiento(hoy.minusDays(3));     Titular t3 = new Titular();     t3.setNumeroDocumento("3");     lic3.setTitular(t3);      when(licenciaExpiradaRepository.findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(         hoy.minusDays(5), hoy)).thenReturn(List.of(lic2, lic1, lic3));     when(licenciaExpiradaMapper.entityToDto(any())).thenAnswer(inv -&gt; {         LicenciaExpirada l = inv.getArgument(0);         var dto = new tp.agil.backend.dtos.LicenciaExpiradaDTO();         dto.setDocumentoTitular(l.getTitular().getNumeroDocumento());         dto.setFechaVencimientoLicencia(l.getFechaVencimiento());         return dto;     });      when(licenciaActivaRepository.findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(any(),         any()))         .thenReturn(Collections.emptyList());      var result = licenciaService.obtenerLicenciasVencidasEntre(hoy.minusDays(5), hoy);      assertEquals(3, result.getExpiradas().size());     assertEquals("2", result.getExpiradas().get(0).getDocumentoTitular()); // más reciente     assertEquals("1", result.getExpiradas().get(1).getDocumentoTitular());     assertEquals("3", result.getExpiradas().get(2).getDocumentoTitular()); // más antigua } </pre>				



@Test

```
void testObtenerLicenciasExpiradas_conActivasVencidasYExpiradas() {
```

```
    LocalDate hoy = LocalDate.now();
```

```
    // Expirada
```

```
    LicenciaExpirada exp1 = new LicenciaExpirada();
```

```
    exp1.setFechaVencimiento(hoy.minusDays(2));
```

```
    Titular t1 = new Titular();
```

```
    t1.setNumeroDocumento("1");
```

```
    exp1.setTitular(t1);
```

```
    // Activa vencida
```

```
    LicenciaActiva act1 = new LicenciaActiva();
```

```
    act1.setFechaVencimiento(hoy.minusDays(1));
```

```
    Titular t2 = new Titular();
```

```
    t2.setNumeroDocumento("10");
```

```
    act1.setTitular(t2);
```

```
when(licenciaExpiradaRepository.findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(hoy.minusDays(5), hoy))
```

```
    .thenReturn(List.of(exp1));
```

```
when(licenciaActivaRepository.findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(hoy.minusDays(5), hoy))
```

```
    .thenReturn(List.of(act1));
```

```
when(licenciaExpiradaMapper.entityToDto(any())).thenReturn(inv -> {
```

```
    LicenciaExpirada l = inv.getArgument(0);
```

```
    var dto = new tp.agil.backend.dtos.LicenciaExpiradaDTO();
```

```
    dto.setDocumentoTitular(l.getTitular().getNumeroDocumento());
```

```
    dto.setFechaVencimientoLicencia(l.getFechaVencimiento());
```

```
    return dto;
```

```
});
```

```
when(licenciaActivaMapper.entityToDto(any())).thenReturn(inv -> {
```

```
    LicenciaActiva l = inv.getArgument(0);
```

```
    var dto = new tp.agil.backend.dtos.LicenciaActivaDTO();
```

```
    dto.setDocumentoTitular(l.getTitular().getNumeroDocumento());
```

```
    dto.setFechaVencimientoLicencia(l.getFechaVencimiento());
```

```
    return dto;
```

```
});
```

```
var result = licenciaService.obtenerLicenciasVencidasEntre(hoy.minusDays(5), hoy);
```

```
assertEquals(1, result.getExpiradas().size());
```

```
assertEquals("1", result.getExpiradas().get(0).getDocumentoTitular());
```

```
assertEquals(1, result.getActivasVencidas().size());
```

```
assertEquals("10", result.getActivasVencidas().get(0).getDocumentoTitular());
```

```
}
```

@Test

```
void testObtenerLicenciasExpiradas_fronterasDeFechas() {
```

```
    LocalDate desde = LocalDate.of(2024, 1, 1);
```

```
    LocalDate hasta = LocalDate.of(2024, 1, 31);
```

```
    LicenciaExpirada lic = new LicenciaExpirada();
```

```
    lic.setFechaVencimiento(desde);
```

```
    Titular t = new Titular();
```

```
    t.setNumeroDocumento("5");
```

```
lic.setTitular(t);

when(licenciaExpiradaRepository.findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(desde,
hasta))
    .thenReturn(List.of(lic));
when(licenciaExpiradaMapper.entityToDto(any())).thenReturn(inv -> {
    LicenciaExpirada l = inv.getArgument(0);
    var dto = new tp.agil.backend.dtos.LicenciaExpiradaDTO();
    dto.setDocumentoTitular(l.getTitular().getNumeroDocumento());
    dto.setFechaVencimientoLicencia(l.getFechaVencimiento());
    return dto;
});

when(licenciaActivaRepository.findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(desde,
hasta))
    .thenReturn(Collections.emptyList());

var result = licenciaService.obtenerLicenciasVencidasEntre(desde, hasta);

assertEquals(1, result.getExpiradas().size());
assertEquals(desde, result.getExpiradas().get(0).getFechaVencimientoLicencia());
}
```

HISTORIA 7: DAR DE ALTA UN TITULAR

TAREA 7.1: DISEÑAR INTERFAZ DE ALTA DE TITULAR

Descripción				
<div>Desarrollar el componente React para el formulario de alta de titular. Implementar manejo de estado del formulario y eventos de validación en tiempo real. Configurar comunicación con el backend para el envío de datos. Los campos necesarios son:</div> <ul style="list-style-type: none"><li>Tipo y número de documento.</li><li>Apellido y Nombre.</li><li>Fecha de nacimiento.</li><li>Dirección.</li><li>Grupo sanguíneo y factor RH.</li><li>Donante de órganos (SI/NO).</li></ul>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	3	2	17/06	17/06
Interfaz				
<div><div><div>Menú</div><div><div>Nueva Licencia</div><div>Renovar Licencia</div><div>Licencias Vigentes</div><div>Licencias a Expirar</div><div>Registrar Titular</div><div>Modificar Titular</div><div>Registrar Usuario</div><div>Modificar Usuario</div></div><div>Cerrar sesión</div></div><div><div>Registrar Titular</div><div><div>Número de DNI</div><div>Ej: 12345678</div></div><div><div>Nombre</div><div></div></div><div><div>Apellido</div><div></div></div><div><div>Fecha de Nacimiento</div><div>mm / dd / yyyy</div></div><div><div>Domicilio</div><div></div></div><div><div>Grupo Sanguíneo y Factor RH</div><div>Selecciona un grupo sanguíneo</div></div><div><div><input type="checkbox"/> Donante de Órganos</div></div><div><div>Cancelar</div><div>Registrar</div></div></div></div>				

## TAREA 7.2: REALIZAR TESTING

## Descripción

Crear pruebas unitarias para el proceso completo de alta de titular. Incluir validación de datos, manejo de duplicados, persistencia correcta, y casos de error. Probar diferentes combinaciones de datos y validar que el sistema mantenga la integridad de los datos.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	3	4	19/06	19/06

## Código

Configuración del test

@BeforeEach

```
void setup() {
    ejemploDTO = new TitularDTO();
    ejemploDTO.setNumeroDocumento("12345678");
    ejemploDTO.setNombre("Juan");
    ejemploDTO.setApellido("Pérez");
    ejemploDTO.setFechaNacimiento(LocalDate.of(1990, 1, 1));
    ejemploDTO.setDomicilio("Calle Falsa 123");
    ejemploDTO.setGrupoFactor("0+");
    ejemploDTO.setDonanteOrganos(true);

    ejemploEntity = new Titular();
    ejemploEntity.setNumeroDocumento("12345678");
    ejemploEntity.setNombre("Juan");
    ejemploEntity.setApellido("Pérez");
    ejemploEntity.setFechaNacimiento(LocalDate.of(1990, 1, 1));
    ejemploEntity.setDomicilio("Calle Falsa 123");
    ejemploEntity.setGrupoFactor("0+");
    ejemploEntity.setDonanteOrganos(true);
}
```

Test Creación Exitosa

@Test

```
void testCrearTitular_ok() {
    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(null);
    when(titularMapper.dtoToEntity(sampleDTO)).thenReturn(sampleEntity);
    when(titularRepository.save(sampleEntity)).thenReturn(sampleEntity);
    when(titularMapper.entityToDto(sampleEntity)).thenReturn(sampleDTO);

    TitularDTO result = titularService.crearTitular(sampleDTO);

    assertEquals("12345678", result.getNumeroDocumento());
    verify(titularRepository).save(sampleEntity);
}
```

Test Duplicado

@Test

```
void testCrearTitular_duplicado_lanzaExcepcion() {
    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(sampleEntity);

    assertThrows(TitularExistenteException.class, () -> titularService.crearTitular(sampleDTO));
    verify(titularRepository, never()).save(any());
}
```

```
}
```

### Test Búsqueda por Documento

```
@Test
```

```
void testGetTitularByNumeroDocumento_ok() {
    when(titularRepository.findByNumeroDocumento("12345678")).thenReturn(sampleEntity);
    when(titularMapper.entityToDto(sampleEntity)).thenReturn(sampleDTO);

    TitularDTO result = titularService.getTitularByNumeroDocumento("12345678");

    assertEquals("12345678", result.getNumeroDocumento());
}
```

### Test Titular No Encontrado

```
@Test
```

```
void testGetTitularByNumeroDocumento_noExiste_lanzaExcepcion() {
    when(titularRepository.findByNumeroDocumento("999999999")).thenReturn(null);

    assertThrows(TitularNoEncontradoException.class,
        () -> titularService.getTitularByNumeroDocumento("999999999"));
}
```

### Test Clases Permitidas por Edad

```
@Test
```

```
void testGetTitularConClases_menorA17_retornaSoloA_B_F_G() {
    Titular menor = new Titular();
    menor.setNumeroDocumento("11111111");
    menor.setFechaNacimiento(LocalDate.now().minusYears(17).minusDays(1));

    when(titularRepository.findByNumeroDocumento("11111111")).thenReturn(menor);
    when(titularMapper.entityToDto(menor)).thenReturn(new TitularDTO());

    TitularClasesDTO dto = titularService.getTitularConClases("11111111");
    assertEquals("A B F G", dto.getClases());
}
```

### Test Clases Profesionales

```
@Test
```

```
void testGetTitularConClases_profesionalConAntiguedad_retornaTodas() {
    Titular titular = new Titular();
    titular.setNumeroDocumento("22222222");
    titular.setFechaNacimiento(LocalDate.now().minusYears(30));

    LicenciaExpirada exp = new LicenciaExpirada();
    exp.setClases("B");
    exp.setFechaEmision(LocalDate.now().minusYears(2));
    titular.setLicenciasExpiradas(List.of(exp));

    when(titularRepository.findByNumeroDocumento("22222222")).thenReturn(titular);
    when(titularMapper.entityToDto(titular)).thenReturn(new TitularDTO());

    TitularClasesDTO dto = titularService.getTitularConClases("22222222");
    assertTrue(dto.getClases().contains("C"));
    assertTrue(dto.getClases().contains("D"));
    assertTrue(dto.getClases().contains("E"));
}
```

## TAREA 7.3: CREAR EL CONTROLADOR

## Descripción

Se debe implementar un endpoint en TitularController con la ruta /api/titulares y método HTTP POST. Este endpoint recibirá un JSON con los campos definidos en TitularDTO (numeroDocumento, nombre, apellido, fechaNacimiento, direccion, grupoFactor, donanteOrganos), y devolverá una respuesta HTTP 200 OK junto con el DTO del titular creado. El controlador se encargará de recibir la petición, delegar la lógica al servicio y retornar el resultado al frontend.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	1	17/06	18/06

## Código

Controlador

```
@RestController
```

```
@RequestMapping("/api/titulares")
```

```
public class TitularController {
```

```
    private final TitularService titularService;
```

```
    public TitularController(TitularService titularService) {
```

```
        this.titularService = titularService;
```

```
    }
```

```
    @PostMapping()
```

```
    public ResponseEntity<TitularDTO> altaTitular(@RequestBody TitularDTO titularDTO) {
```

```
        TitularDTO titularCreado = titularService.crearTitular(titularDTO);
```

```
        return new ResponseEntity<>(titularCreado, HttpStatus.OK);
```

```
    }
```

```
}
```

DTO

```
public class TitularDTO {
```

```
    private String numeroDocumento;
```

```
    private String nombre;
```

```
    private String apellido;
```

```
    @DateTimeFormat(pattern = "yyyy-MM-dd")
```

```
    private LocalDate fechaNacimiento;
```

```
    private String domicilio;
```

```
    private String grupoFactor;
```

```
    private boolean donanteOrganos;
```

```
}
```

## TAREA 7.4: CREAR EL SERVICIO Y REPOSITORIO

## Descripción

Desarrollar el método crearTitular(TitularDTO titularDTO) en el servicio (TitularServiceImpl) y definir el repositorio (TitularRepository). El servicio debe validar que no exista previamente un titular con el mismo numeroDocumento recibido en el DTO, evitando duplicados. Si el titular no existe, se procede a mapear el DTO a entidad y guardar el nuevo titular en la base de datos usando el repositorio. Si ya existe, se debe manejar el caso creando un TitularExistenteException y devolver el error adecuado para que el frontend lo pueda manejar.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	2	2	17/06	18/06

## Código

Servicio

@Service

public class TitularServiceImpl implements TitularService {

private final TitularRepository titularRepository;

private final TitularMapper titularMapper;

public TitularServiceImpl(TitularRepository titularRepository, TitularMapper titularMapper) {

this.titularRepository = titularRepository;

this.titularMapper = titularMapper;

}

## @Override

public TitularDTO crearTitular(TitularDTO titularDTO) {

Titular existente = titularRepository.findByNumeroDocumento(titularDTO.getNumeroDocumento());

if (existente != null) {

throw new TitularExistenteException("Ya existe un titular con el número de documento

proporcionado.");

}

Titular titularGuardado = titularRepository.save(titularMapper.dtoToEntity(titularDTO));

return titularMapper.entityToDto(titularGuardado);

}

}

Repositorio

public interface TitularRepository extends JpaRepository&lt;Titular, Long&gt; {

Titular findByNumeroDocumento(String numeroDocumento);

}

Excepción

public class TitularExistenteException extends RuntimeException {

public TitularExistenteException(String message) {

super(message);

}

}

TAREA 7.5: AGREGAR VALIDACIÓN DE CAMPOS PARA EL FORMULARIO

Descripción				
Implementar validaciones completas en frontend. Incluir validación de formato de documento, validación de fecha de nacimiento, longitud de campos y selección válida de grupo sanguíneo y factor RH.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	1	1	17/06	17/06
Interfaz				
<div><div><div>Menú</div><div><div>Nueva Licencia</div><div>Renovar Licencia</div><div>Licencias Vigentes</div><div>Licencias a Expirar</div><div>Registrar Titular</div><div>Modificar Titular</div><div>Registrar Usuario</div><div>Modificar Usuario</div></div><div>Cerrar sesión</div></div><div><div>Registrar Titular</div><div><div>Número de DNI</div><div>23456789</div></div><div><div>Nombre</div><div></div><div>El nombre es obligatorio</div></div><div><div>Apellido</div><div></div><div>El apellido es obligatorio</div></div><div><div>Fecha de Nacimiento</div><div>07 / 17 / 1973</div></div><div><div>Domicilio</div><div>Pedro de Vega 114</div></div><div><div>Grupo Sanguíneo y Factor RH</div><div>A+</div></div><div><div><input type="checkbox"/> Donante de Órganos</div></div><div><div>Cancelar</div><div>Registrar</div></div></div></div>				

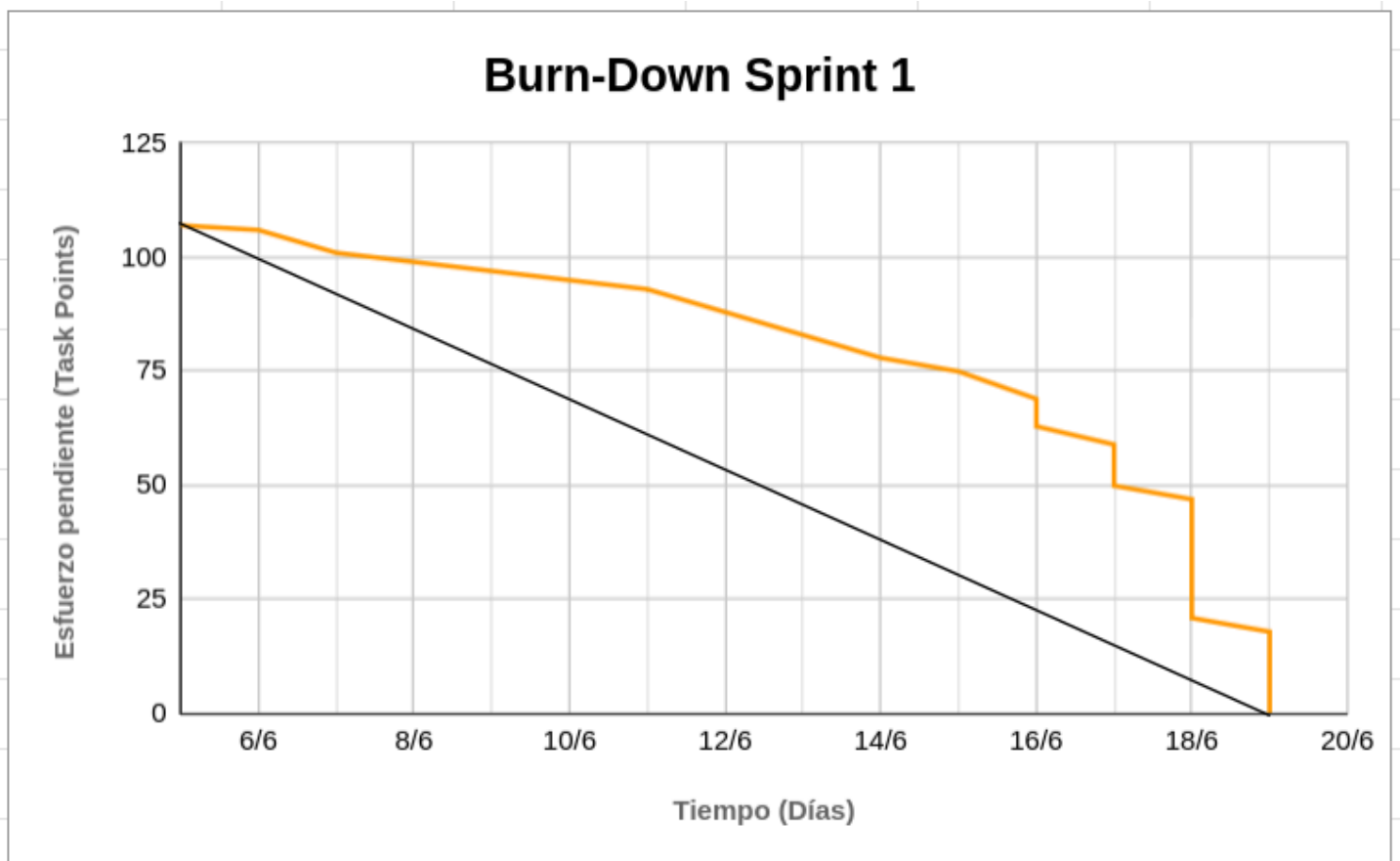
GRÁFICO BURN DOWN

Decidimos realizar los gráficos burn down en base a las tareas y no a las historias de usuario debido a que las primeras reflejan de manera más fiel a la realidad cómo fue la evolución de los sprints.

FECHA	TASK POINTS REAL RESTANTE	TASK POINTS IDEAL RESTANTE
05/06	107	100
06/06	106	93
07/06	101	86
08/06	101	78
09/06	101	71
10/06	101	64
11/06	93	57
12/06	93	50



13/06	93	43
14/06	78	36
15/06	75	29
16/06	63	21
17/06	50	14
18/06	21	7
19/06	0	0



Como podemos ver en el gráfico, el flujo real de trabajo siempre estuvo por encima del estimado. Este es un punto a tener en cuenta, ya que por más que el trabajo fue realizado de manera completa a tiempo, esto no es lo ideal, ya que el margen estrecho con respecto a las fechas puede hacer que se cometan errores no deseados.

### DESCRIPCIÓN DE CADA DAILY MEETING

#### DAILY MEETING 12-06

Esta Daily Meeting se realizó de forma presencial, justo después de la charla de Argeniss en la facultad. Fue una sesión en la que se discutieron diversas cuestiones, como:

- Definición de Endpoints: Se puso en común y se discutió el formato de los endpoints para cada operación requerida por el frontend.
- Puesta en común sobre la disposición y el orden de los campos en el frontend

- Cada miembro del equipo compartió cómo le estaba resultando la implementación de sus tareas. En general, identificamos un patrón: cada uno sabía lo que tenía que hacer individualmente, pero surgían dudas importantes respecto a la relación y la integración con los demás componentes del sistema.
- Se realizó una discusión de las próximas tareas a realizar en cada caso, intentando despejar dudas inmediatas y planificar los siguientes pasos.

#### DAILY MEETING 17-06

Esta Daily Meeting se llevó a cabo de forma presencial y se abordaron los siguientes puntos:

- Se planteó una duda sobre cómo debían funcionar las licencias. La pregunta fue si una licencia podía tener múltiples clases con distintos vencimientos, o si tenía un único vencimiento que se renovaba para todas las clases al renovar la de menor vencimiento. Acordamos que esta consulta se elevaría al Product Owner para obtener una definición clara y no avanzar con ambigüedades.
- Otro punto clave fue la posibilidad de modificar tareas que todavía no habíamos empezado. La idea era consultar al profesor (Scrum Master) si podíamos hacerlo, para poder reescribir y detallar mejor las tareas pendientes.
- Criterios y convenciones para los DTOs y los JSON que se utilizarían en la comunicación entre frontend y backend.

#### DAILY MEETING 19-06

Esta Daily Meeting fue realizada para ultimar detalles y prepararnos para la revisión. Revisamos el progreso final de todas las historias:

- Confirmamos que los cálculos, interfaces y testeos estaban prácticamente cerrados. Lo que quedaba eran ajustes menores.
- Se plantearon las últimas cosas a solucionar como algunas validaciones de campos. Nos enfocamos en intentar resolver esos detalles lo antes posible para que todo quedara listo para la revisión.
- El objetivo era constatar que cada parte del sistema estuviera integrada y funcionando perfectamente para la revisión. No había impedimentos mayores, solo el ajuste de esos últimos detalles.

#### REVISIÓN DEL SPRINT

- Se mostró el sistema desarrollado al Product Owner. Durante la presentación, se corrió el sistema validando las implementaciones hechas hasta el momento. El feedback que obtuvimos fue positivo ya que cumplía con las expectativas. Además, el PO testeo ciertas funcionalidades básicas propias del sprint, que funcionaban correctamente.
- Además del feedback de la funcionalidad del sistema, el PO mencionó ciertos puntos a considerar como:
  - La importancia de desarrollar correctamente las tareas, no introducir modificaciones externas pero sí internas durante el desarrollo del sprint.
  - Nos sugirió distribuir las tareas de una misma historia entre varios desarrolladores, evitando que una sola persona las haga.
  - Visto bueno de la lógica de negocio y funcionamiento de lo involucrado en las historias desarrolladas en el sprint.

#### RETROSPECTIVA DEL SPRINT

- Como grupo, nos dimos cuenta de la importancia que involucra una planificación detallada.
- Muchas veces nos sucedía que no sabíamos cómo conectar las distintas partes del sistema, ya fuera en el frontend, el backend o entre ambos.
- Esta confusión se dio por una falta de detalle en la descripción de nuestras historias y tareas. Esta carencia se fundamenta en una planificación del sprint rápida y breve, basada solamente en definir los story points sin desempeñarnos correctamente en detallar la planificación. Nos faltó especificar aspectos como datos, métodos, DTOs, controllers, entidades y la relación entre ellas.
- En la retrospectiva, hablamos de esmerarnos más a la hora de refinar las historias y detallar las tareas en los próximos sprints.

## PLANIFICACIÓN SPRINT 2

### ESTIMACIÓN DE ESFUERZO, PLANIFICACIÓN Y SPRINT BACKLOG

Sprint 2 - Backlog:

7 actividades			29 Story Points
<u>N°</u>	<u>Historia de usuario</u>	<u>Prioridad</u>	<u>Story Points</u>
8	Emitir una copia	Baja	4
9	Listado de licencias vigentes por criterios	Baja	8
10	Modificar los datos de un titular existente	Baja	4
11	Dar de alta un usuario	Baja	4
12	Modificar los datos de un usuario	Baja	3
13	Iniciar sesión	Baja	4
14	Cerrar sesión	Baja	2

### DIVISIÓN EN TAREAS

Teniendo en cuenta que cada task point (abreviado como *tp*) tiene una duración de 30min, determinamos la siguiente división de tareas:

<u>N°</u>	<u>Historia de usuario</u>	<u>N°</u>	<u>Tareas</u>
8	Emitir una copia	8.1	Añadir botón "Emitir Copia" en la UI
9	Listado de licencias vigentes por criterios	9.1	Añadir endpoint de búsqueda filtrada en LicenciaController
		9.2	Implementar testing para el servicio de búsqueda
		9.3	Mostrar resultados de la búsqueda
		9.4	Diseñar búsqueda de licencias
		9.5	Habilitar LicenciaActivaRepository para consultas dinámicas
		9.6	Implementar lógica de filtrado en LicenciaServiceImpl
10	Modificar los datos de un titular existente	10.1	Añadir endpoint de modificación en TitularController
		10.2	Implementar la lógica de modificación en TitularServiceImpl
		10.3	Finalizar implementación de renovación de licencia por modificación de datos del titular
		10.4	Crear interfaz de búsqueda y modificación de titular

11	Dar de alta un usuario	11.1	Ampliar la entidad Usuario
		11.2	Crear formulario para alta de usuarios
		11.3	Crear UsuarioController para alta de usuarios
		11.4	Crear UsuarioServiceImpl
		11.5	Crear DTOs y Mappers para Usuario
12	Modificar los datos de un usuario	12.1	Crear endpoint para modificación de usuario
		12.2	Implementar Service y Repository para modificar usuarios
		12.3	Crear formulario de modificación de usuarios
13	Iniciar sesión	13.1	Implementar UserDetailsService
		13.2	Configurar Spring Security
		13.3	Crear la página de login
		13.4	Crear AuthController para Login/Logout
14	Cerrar sesión	14.1	Añadir botón de Logout

## TAREAS DETALLADAS

### HISTORIA 8: EMITIR UNA COPIA

#### TAREA 8.1: AÑADIR BOTÓN “EMITIR COPIA” EN LA UI

Descripción								
En el listado de licencias vigentes debe crearse un botón de “emitir copia” para cada licencia listada que redirigirá a la página de imprimir licencia.								
Persona asignada	TP estimados	TP reales	Inicio	Fin				
Kevin Brunas	3	2	03/07	03/07				
Interfaz								
Buscar Licencias Vigentes								
Nombre	Apellido	Grupo sanguíneo	¿Donante?	Buscar				
Nombre	Apellido	DNI	Grupo	Donante	Clases	Fecha de Emisión	Fecha de Vencimiento	Acción
Juan	Pérez	12345678	A+	Sí	A	2025-07-03	2030-05-15	<a href="#">Imprimir</a>
Roberto	Flores	20000000	B-	No	B	2025-06-28	2030-06-28	<a href="#">Imprimir</a>
Carla	Díaz	21000000	O+	Sí	B	2023-06-28	2028-06-28	<a href="#">Imprimir</a>

**HISTORIA 9: LISTADO DE LICENCIAS VIGENTES POR CRITERIOS****TAREA 9.1: AÑADIR ENDPOINT DE BÚSQUEDA FILTRADA EN LICENCIACONTROLLER**

Descripción				
En LicenciaController.java, añadir un nuevo endpoint para la búsqueda.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	1	1	28/06	28/06
Código				
<pre>@GetMapping("/filtros") public ResponseEntity&lt;List&lt;LicenciaActivaDTO&gt;&gt; getLicenciasVigentesPorCriterios(     @RequestParam(required = false) String nombre,     @RequestParam(required = false) String apellido,     @RequestParam(required = false) String grupoFactor,     @RequestParam(required = false) Boolean esDonante ) {     List&lt;LicenciaActivaDTO&gt; licencias = licenciaService.buscarLicenciasPorCriterios(nombre, apellido,     grupoFactor, esDonante);     return new ResponseEntity&lt;&gt;(licencias, HttpStatus.OK); }</pre>				

**TAREA 9.2: IMPLEMENTAR TESTING PARA EL SERVICIO DE BÚSQUEDA**

Descripción				
Crear tests unitarios que verifiquen que el filtrado dinámico funciona correctamente para distintas combinaciones de criterios.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	2	3	02/07	02/07
Código				
<p><u>Test de búsqueda filtrando solo por nombre del titular</u></p> <pre>@Test void testBuscarLicenciasPorCriterios_soloNombre() {     LicenciaActiva lic = new LicenciaActiva();     Titular titular = new Titular();     titular.setNombre("Juan");     lic.setTitular(titular);      when(licenciaActivaRepository.findAll(org.mockito.ArgumentMatchers.&lt;Specification&lt;LicenciaActiva&gt;&gt;any()))     .thenReturn(List.of(lic));     when(licenciaActivaMapper.entityToDto(any())).thenReturn(inv -&gt; {         LicenciaActiva l = inv.getArgument(0);         LicenciaActivaDTO dto = new LicenciaActivaDTO();         dto.setNombreTitular(l.getTitular().getNombre());         return dto;     }); }</pre>				

```
});

var result = licenciaService.buscarLicenciasPorCriterios("Juan", null, null, null);

assertEquals(1, result.size());
assertEquals("Juan", result.get(0).getNombreTitular());
}
```

#### Test de filtrado sólo por nombre y apellido del titular

@Test

```
void testBuscarLicenciasPorCriterios_nombreYApellido() {
    LicenciaActiva lic = new LicenciaActiva();
    Titular titular = new Titular();
    titular.setNombre("Ana");
    titular.setApellido("García");
    lic.setTitular(titular);
```

```
when(licenciaActivaRepository.findAll(org.mockito.ArgumentMatchers.<Specification<LicenciaActiva>>any()
))
    .thenReturn(List.of(lic));
when(licenciaActivaMapper.entityToDto(any())).thenReturn(inv -> {
    LicenciaActiva l = inv.getArgument(0);
    LicenciaActivaDTO dto = new LicenciaActivaDTO();
    dto.setNombreTitular(l.getTitular().getNombre());
    dto.setApellidoTitular(l.getTitular().getApellido());
    return dto;
});

var result = licenciaService.buscarLicenciasPorCriterios("Ana", "García", null, null);

assertEquals(1, result.size());
assertEquals("Ana", result.get(0).getNombreTitular());
assertEquals("García", result.get(0).getApellidoTitular());
}
```

#### Test de búsqueda filtrando por grupo factor sanguíneo y condición de donante

@Test

```
void testBuscarLicenciasPorCriterios_grupoFactorYDonante() {
    LicenciaActiva lic = new LicenciaActiva();
    Titular titular = new Titular();
    titular.setGrupoFactor("A+");
    titular.setDonanteOrganos(true);
    lic.setTitular(titular);
```

```
when(licenciaActivaRepository.findAll(org.mockito.ArgumentMatchers.<Specification<LicenciaActiva>>any()
))
    .thenReturn(List.of(lic));
when(licenciaActivaMapper.entityToDto(any())).thenReturn(inv -> {
    LicenciaActiva l = inv.getArgument(0);
    LicenciaActivaDTO dto = new LicenciaActivaDTO();
    dto.setGrupoFactor(l.getTitular().getGrupoFactor());
    dto.setDonanteOrganos(l.getTitular().isDonanteOrganos() ? "Si" : "No");
    return dto;
});
```

```
var result = licenciaService.buscarLicenciasPorCriterios(null, null, "A+", true);
```

```
assertEquals(1, result.size());
assertEquals("A+", result.get(0).getGrupoFactor());
assertEquals("Si", result.get(0).getDonanteOrganos());
```

```
}
```

#### Test de búsqueda sin resultados cuando no hay coincidencias

```
@Test
```

```
void testBuscarLicenciasPorCriterios_sinResultados() {
```

```
when(licenciaActivaRepository.findAll(org.mockito.ArgumentMatchers.<Specification<LicenciaActiva>>any()
))
```

```
.thenReturn(Collections.emptyList());
```

```
var result = licenciaService.buscarLicenciasPorCriterios("NoExiste", null, null, null);
```

```
assertNotNull(result);
assertTrue(result.isEmpty());
```

```
}
```

#### Test de búsqueda cuando todos los criterios son nulos

```
@Test
```

```
void testBuscarLicenciasPorCriterios_todosNulos() {
```

```
    LicenciaActiva lic = new LicenciaActiva();
```

```
    Titular titular = new Titular();
```

```
    titular.setNombre("Pedro");
```

```
    lic.setTitular(titular);
```

```
when(licenciaActivaRepository.findAll(org.mockito.ArgumentMatchers.<Specification<LicenciaActiva>>any()
))
```

```
.thenReturn(List.of(lic));
```

```
when(licenciaActivaMapper.entityToDto(any())).thenReturn(inv -> {
```

```
    LicenciaActiva l = inv.getArgument(0);
```

```
    LicenciaActivaDTO dto = new LicenciaActivaDTO();
```

```
    dto.setNombreTitular(l.getTitular().getNombre());
```

```
    return dto;
```

```
});
```

```
var result = licenciaService.buscarLicenciasPorCriterios(null, null, null, null);
```

```
assertEquals(1, result.size());
assertEquals("Pedro", result.get(0).getNombreTitular());
```

```
}
```

TAREA 9.3: MOSTRAR RESULTADOS DE LA BÚSQUEDA

Descripción																																																																												
Mostrar las licencias activas devueltas por la API en una tabla, incluyendo la información relevante de cada licencia (Nombre, Apellido, DNI, Grupo y Factor, Donante, Clases, Fecha de Emisión y Fecha de vencimiento).																																																																												
Persona asignada	TP estimados	TP reales	Inicio	Fin																																																																								
Kevin Brunas	4	4	02/07	02/07																																																																								
Interfaz																																																																												
<div><div><div>Menú</div><div><div>Nueva Licencia</div><div>Renovar Licencia</div><div>Licencias Vigentes</div><div>Licencias a Expirar</div><div>Registrar Titular</div><div>Modificar Titular</div><div>Registrar Usuario</div><div>Modificar Usuario</div></div><div>Cerrar sesión</div></div><div><div>Buscar Licencias Vigentes</div><div><div><div>Nombre</div><div>Apellido</div><div>Grupo sanguíneo</div><div>¿Donante?</div><div>Buscar</div></div><table><tr><th>Nombre</th><th>Apellido</th><th>DNI</th><th>Grupo</th><th>Donante</th><th>Clases</th><th>Fecha de Emisión</th><th>Fecha de Venciminto</th><th>Acción</th></tr><tr><td>Juan</td><td>Pérez</td><td>12345678</td><td>A+</td><td>Sí</td><td>A</td><td>2025-07-04</td><td>2030-05-15</td><td><a href="#">Imprimir</a></td></tr><tr><td>Roberto</td><td>Flores</td><td>20000000</td><td>B-</td><td>No</td><td>B</td><td>2025-06-28</td><td>2030-06-28</td><td><a href="#">Imprimir</a></td></tr><tr><td>Carla</td><td>Díaz</td><td>21000000</td><td>O+</td><td>Sí</td><td>B</td><td>2023-06-28</td><td>2028-06-28</td><td><a href="#">Imprimir</a></td></tr><tr><td>Pedro</td><td>García</td><td>66000000</td><td>A-</td><td>Sí</td><td>B</td><td>2023-06-28</td><td>2028-06-28</td><td><a href="#">Imprimir</a></td></tr><tr><td>Carlos</td><td>Vega</td><td>35000000</td><td>O+</td><td>No</td><td>B</td><td>2023-01-01</td><td>2028-01-01</td><td><a href="#">Imprimir</a></td></tr><tr><td>Fernando</td><td>Ruiz</td><td>45000000</td><td>B+</td><td>No</td><td>B</td><td>2022-06-28</td><td>2027-06-28</td><td><a href="#">Imprimir</a></td></tr><tr><td>María</td><td>Vencida</td><td>60000000</td><td>O+</td><td>Sí</td><td>B</td><td>2020-06-28</td><td>2025-06-28</td><td><a href="#">Imprimir</a></td></tr></table></div></div></div>					Nombre	Apellido	DNI	Grupo	Donante	Clases	Fecha de Emisión	Fecha de Venciminto	Acción	Juan	Pérez	12345678	A+	Sí	A	2025-07-04	2030-05-15	<a href="#">Imprimir</a>	Roberto	Flores	20000000	B-	No	B	2025-06-28	2030-06-28	<a href="#">Imprimir</a>	Carla	Díaz	21000000	O+	Sí	B	2023-06-28	2028-06-28	<a href="#">Imprimir</a>	Pedro	García	66000000	A-	Sí	B	2023-06-28	2028-06-28	<a href="#">Imprimir</a>	Carlos	Vega	35000000	O+	No	B	2023-01-01	2028-01-01	<a href="#">Imprimir</a>	Fernando	Ruiz	45000000	B+	No	B	2022-06-28	2027-06-28	<a href="#">Imprimir</a>	María	Vencida	60000000	O+	Sí	B	2020-06-28	2025-06-28	<a href="#">Imprimir</a>
Nombre	Apellido	DNI	Grupo	Donante	Clases	Fecha de Emisión	Fecha de Venciminto	Acción																																																																				
Juan	Pérez	12345678	A+	Sí	A	2025-07-04	2030-05-15	<a href="#">Imprimir</a>																																																																				
Roberto	Flores	20000000	B-	No	B	2025-06-28	2030-06-28	<a href="#">Imprimir</a>																																																																				
Carla	Díaz	21000000	O+	Sí	B	2023-06-28	2028-06-28	<a href="#">Imprimir</a>																																																																				
Pedro	García	66000000	A-	Sí	B	2023-06-28	2028-06-28	<a href="#">Imprimir</a>																																																																				
Carlos	Vega	35000000	O+	No	B	2023-01-01	2028-01-01	<a href="#">Imprimir</a>																																																																				
Fernando	Ruiz	45000000	B+	No	B	2022-06-28	2027-06-28	<a href="#">Imprimir</a>																																																																				
María	Vencida	60000000	O+	Sí	B	2020-06-28	2025-06-28	<a href="#">Imprimir</a>																																																																				

TAREA 9.4: DISEÑAR BÚSQUEDA DE LICENCIAS

Descripción				
Diseñar interfaz de usuario con campos de entrada para el nombre, apellido, grupo sanguíneo (grupoFactor) y una lista desplegable para indicar si es donante.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Kevin Brunas	4	6	02/07	02/07
Interfaz				
<div><div>Buscar Licencias Vigentes</div><div><div>Nombre</div><div>Apellido</div><div>Grupo sanguíneo</div><div>¿Donante?</div><div>Buscar</div></div></div>				



### TAREA 9.5: HABILITAR LICENCIAACTIVAREPOSITORY PARA CONSULTAS DINÁMICAS

Descripción				
Modificar la interfaz LicenciaActivaRepository para que extienda de JpaSpecificationExecutor<LicenciaActiva>.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	1	1	28/06	28/06
Código				
<pre>public interface LicenciaActivaRepository extends JpaRepository&lt;LicenciaActiva, Long&gt;, JpaSpecificationExecutor&lt;LicenciaActiva&gt; {     LicenciaActiva findByTitular_NumeroDocumento(String titular_numeroDocumento);     List&lt;LicenciaActiva&gt; findByFechaVencimientoBetweenOrderByFechaVencimientoDesc(LocalDate desde,     LocalDate hasta); }</pre>				

### TAREA 9.6: IMPLEMENTAR LÓGICA DE FILTRADO EN LICENCIASERVICEIMPL

Descripción				
En LicenciaServiceImpl.java, implementar el método buscarLicenciasPorCriterios. Utilizar JPA Specifications para construir una consulta WHERE dinámica que una LicenciaActiva con Titular y filtre por los campos requeridos.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	4	6	28/06	29/06
Código				
<pre>@Override public List&lt;LicenciaActivaDTO&gt; buscarLicenciasPorCriterios(String nombre, String apellido, String grupoFactor, Boolean esDonante) {     // Comienza con una especificación vacía (sin filtros)     Specification&lt;LicenciaActiva&gt; spec = (root, query, cb) -&gt; cb.conjunction();      // Agrega filtro por nombre si corresponde     if (nombre != null &amp;&amp; !nombre.isEmpty()) {         spec = spec.and((root, query, cb) -&gt; cb.equal(root.get("titular").get("nombre"), nombre));     }     // Agrega filtro por apellido si corresponde     if (apellido != null &amp;&amp; !apellido.isEmpty()) {         spec = spec.and((root, query, cb) -&gt; cb.equal(root.get("titular").get("apellido"), apellido));     }     // Agrega filtro por grupo sanguíneo si corresponde     if (grupoFactor != null &amp;&amp; !grupoFactor.isEmpty()) {         spec = spec.and((root, query, cb) -&gt; cb.equal(root.get("titular").get("grupoFactor"), grupoFactor));     }     // Agrega filtro por donante si corresponde     if (esDonante != null) {         spec = spec.and((root, query, cb) -&gt; cb.equal(root.get("titular").get("donanteOrganos"), esDonante));     } }</pre>				

```

}

List<LicenciaActiva> licencias = licenciaActivaRepository.findAll(spec);
return licencias.stream()
    .map(licenciaActivaMapper::entityToDto)
    .collect(Collectors.toList());
}

```

## HISTORIA 10: MODIFICAR LOS DATOS DE UN TITULAR EXISTENTE

### TAREA 10.1: AÑADIR ENDPOINT DE MODIFICACIÓN EN TITULARCONTROLLER

Descripción				
En TitularController.java, añadir el método para PUT.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	1	1	29/06	29/06
Código				
<pre> @PutMapping("/{numeroDocumento}") public ResponseEntity&lt;TitularDTO&gt; modificarTitular(@PathVariable String numeroDocumento, @RequestBody TitularDTO titularDTO) {     TitularDTO titularActualizado = titularService.modificarTitular(titularDTO);     return new ResponseEntity&lt;&gt;(titularActualizado, HttpStatus.OK); } </pre>				

### TAREA 10.2: IMPLEMENTAR LA LÓGICA DE MODIFICACIÓN EN TITULARSERVICEIMPL

Descripción				
<p>En TitularServiceImpl.java, implementar el método modificarTitular. La lógica debe:</p> <ul style="list-style-type: none"> <li>• Buscar el Titular por numeroDocumento. Lanzar TitularNoEncontradoException si no existe.</li> <li>• Utilizar TitularMapper para actualizar los campos de la entidad con los del DTO.</li> <li>• Guardar la entidad actualizada con titularRepository.save().</li> </ul>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	2	2	29/06	29/06
Código				
<pre> @Override public TitularDTO modificarTitular(TitularDTO titularDTO) {     // Busca el titular por número de documento     Titular titular = titularRepository.findByNumeroDocumento(titularDTO.getNumeroDocumento());     if (titular == null) {         // Lanza excepción si no existe         throw new TitularNoEncontradoException("No se encontró un titular con el número de documento: " + titularDTO.getNumeroDocumento());     } } </pre>				

```
// Actualiza los datos del titular
titular.setNombre(titularDTO.getNombre());
titular.setApellido(titularDTO.getApellido());
titular.setFechaNacimiento(titularDTO.getFechaNacimiento());
titular.setDomicilio(titularDTO.getDomicilio());
titular.setGrupoFactor(titularDTO.getGrupoFactor());
titular.setDonanteOrganos(titularDTO.isDonanteOrganos());

// Guarda y retorna el titular actualizado como DTO
Titular actualizado = titularRepository.save(titular);
return titularMapper.entityToDto(actualizado);
}
```

### TAREA 10.3: FINALIZAR LA IMPLEMENTACIÓN DE RENOVACIÓN DE LICENCIA POR MODIFICACIÓN DE DATOS DEL TITULAR

#### Descripción

En TitularServiceImpl.java, completar la lógica para permitir la renovación de una licencia cuando el motivo sea la modificación de datos del titular. Esto implica permitir el motivo modificación en el endpoint y en el service de renovación, no exigir que la licencia esté vencida si el motivo es modificación. Además, asegurar que la nueva licencia se emita con los datos actualizados del titular y que la anterior quede expirada.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	2	02/07	02/07

#### Código

```
@Override
@Transactional
public LicenciaEmitidaDTO renovarLicencia(LicenciaFormDTO licenciaFormDTO, String motivo) {
    if (!"vencimiento".equalsIgnoreCase(motivo) && !"modificacion".equalsIgnoreCase(motivo)) {
        throw new IllegalArgumentException("El motivo debe ser 'vencimiento' o 'modificacion'");
    }

    Titular titular = titularRepository.findByNumeroDocumento(licenciaFormDTO.getDocumentoTitular());
    if (titular == null) {
        throw new TitularNoEncontradoException("No se encontró un titular con el número de documento: " +
            licenciaFormDTO.getDocumentoTitular());
    }

    LicenciaActiva licenciaAnterior = titular.getLicenciaActiva();
    if (licenciaAnterior == null) {
        throw new LicenciaNoEncontradaException("No hay licencia activa para renovar");
    }

    if ("vencimiento".equalsIgnoreCase(motivo) &&
        licenciaAnterior.getFechaVencimiento().isAfter(LocalDate.now())) {
        throw new IllegalArgumentException("La licencia aún no está vencida, no se puede renovar");
    }

    String documentoUsuario = obtenerDocumentoUsuarioAutenticado();
    Usuario usuario = usuarioRepository.findByNumeroDocumento(documentoUsuario);
```

```

if (usuario == null) {
    throw new UsuarioNoEncontradoException("No se encontró un usuario con el número de documento:
" +
    documentoUsuario);
}

LocalDate fechaEmision = LocalDate.now();
LocalDate fechaVencimiento = calcularFechaVencimiento(titular);

LicenciaActiva nuevaLicencia = expirarYCrearNuevaLicenciaActiva(titular, usuario, licenciaFormDTO,
    fechaEmision, fechaVencimiento);

LicenciaEmitidaDTO dto = licenciaEmitidaMapper.entityToDto(nuevaLicencia);
dto.setDocumentoTitular(titular.getNumeroDocumento());
dto.setDocumentoUsuario(usuario.getNumeroDocumento());
return dto;
}

```

#### TAREA 10.4: CREAR INTERFAZ DE BÚSQUEDA Y MODIFICACIÓN DE TITULAR

##### Descripción

Implementar formulario de modificación de titular. Refactorizar el componente de búsqueda de titular para que acepte como parámetros una función de búsqueda y un manejador. Reutilizar estructura del formulario de alta de titular, permitiendo que todos los campos sean modificables, salvo DNI. Implementar función con método PUT a /api/titulares/{dni}.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	3	3	02/07	02/07

##### Interfaz

**Menú**

- Nueva Licencia
- Renovar Licencia
- Licencias Vigentes
- Licencias a Expirar
- Registrar Titular
- Modificar Titular
- Registrar Usuario
- Modificar Usuario

Cerrar sesión

### Modificar Titular

DNI

12345678 Buscar

Nombre

Juan

Apellido

Pérez

Fecha de nacimiento

05/15/1990

Dirección

Calle Falsa 123

Grupo Sanguíneo

A+

☒ Donante de órganos

Borrar
Actualizar

**HISTORIA 11: DAR DE ALTA UN USUARIO****TAREA 11.1: AMPLIAR LA ENTIDAD USUARIO**

Descripción				
<p>En Usuario.java, añadir los campos necesarios para garantizar la seguridad. Esos campos son:</p> <ul style="list-style-type: none"> <li>Contraseña: String para guardar la contraseña hasheada en la BD.</li> <li>Rol: String ADMIN o USER para restringir el acceso a los endpoints.</li> </ul> <p>Además, la clase debe implementar UserDetails para realizar correctamente el proceso de autenticación.</p>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	1	30/06	30/06
Código				
<pre> @Getter @Setter @NoArgsConstructor @Entity public class Usuario implements UserDetails {     @Id     @GeneratedValue(strategy = GenerationType.IDENTITY)     private Long id;      @Column(unique = true)     private String numeroDocumento;      private String nombre;     private String apellido;     private String email;      @Column(nullable = false)     private String contrasena;      @Column(nullable = false)     private String rol; // ADMIN o USER      @OneToMany(mappedBy = "usuario")     private List&lt;Licencia&gt; licenciasTramitadas;      @Override     public Collection&lt;? extends GrantedAuthority&gt; getAuthorities() {         return List.of(new SimpleGrantedAuthority("ROLE_" + this.rol));     }      @Override     public String getPassword() {         return this.contrasena;     }      @Override     public String getUsername() {         return this.numeroDocumento;     } } </pre>				

```
@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return true;
}
}
```

TAREA 11.2: CREAR FORMULARIO PARA ALTA DE USUARIOS**Descripción**

Implementar formulario de alta de usuario, con los campos DNI, nombre, apellido, email, rol (no visible), contraseña y confirmación de contraseña. Crear schema para validar longitud de DNI igual a 8, nombre, apellido y email no nulos, contraseña de al menos 6 caracteres e igualdad entre contraseña y su confirmación. Crear servicio de usuario y una función con método POST a /api/usuarios.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	3	3	02/07	02/07

**Interfaz**
TAREA 11.3: CREAR USUARIOCONTROLLER PARA ALTA DE USUARIOS**Descripción**

Crear un nuevo controlador UsuarioController que contendrá el endpoint POST /api/usuarios para dar de alta un nuevo usuario administrativo. Recibirá un UsuarioFormDTO con los campos numeroDocumento, apellido, nombre, email, contrasena, rol.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	1	30/06	30/06

**Código**

```
Controlador
@RestController
@RequestMapping("/api/usuarios")
public class UsuarioController {
    private final UsuarioService usuarioService;
```

```

public UsuarioController(UsuarioService usuarioService) {
    this.usuarioService = usuarioService;
}

@PostMapping()
public ResponseEntity<UsuarioDTO> altaUsuario(@RequestBody UsuarioFormDTO usuarioFormDTO){
    UsuarioDTO usuarioCreado = usuarioService.crearUsuario(usuarioFormDTO);
    return new ResponseEntity<>(usuarioCreado, HttpStatus.OK);
}
}

DTO
public class UsuarioFormDTO {
    private String numeroDocumento;
    private String apellido;
    private String nombre;
    private String email;
    private String rol;
    private String contrasena;
}

```

#### TAREA 11.4: CREAR USUARIOSERVICEIMPL

Descripción				
<p>Crear el servicio que implemente UsuarioService y contenga la lógica para crear el usuario. Se debe usar un PasswordEncoder para hashear la contraseña, validar que no existe un usuario con ese numeroDocumento, persistir la entidad y devolver un UsuarioDTO con sus campos pero sin la contraseña.</p>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	3	3	30/06	30/06
Código				
<pre> @Service public class UsuarioServiceImpl implements UsuarioService {     private PasswordEncoder passwordEncoder;     private UsuarioRepository usuarioRepository;      public UsuarioServiceImpl(PasswordEncoder passwordEncoder, UsuarioRepository usuarioRepository) {         this.passwordEncoder = passwordEncoder;         this.usuarioRepository = usuarioRepository;     }      @Override     public UsuarioDTO crearUsuario(UsuarioFormDTO usuarioFormDTO) {         Usuario existente =         usuarioRepository.findByNumeroDocumento(usuarioFormDTO.getNumeroDocumento());         if (existente != null) throw new UsuarioExistenteException("Ya existe un usuario con el numero de         documento proporcionado");          Usuario guardado =         usuarioRepository.save(UsuarioMapper.usuarioFormDTOaUsuario(usuarioFormDTO, passwordEncoder)); </pre>				



```

    return UsuarioMapper.usuarioAusuarioDTO(guardado);
}
}

```

#### TAREA 11.5: CREAR DTOS PARA USUARIO Y USUARIOMAPPER

Descripción				
<p>Crear UsuarioMapper para convertir entre entidad y UsuarioDTO. Crear los DTOs:</p> <ul style="list-style-type: none"> <li>• UsuarioFormDTO: numeroDocumento, apellido, nombre, email, rol, contrasena.</li> <li>• UsuarioDTO: numeroDocumento, apellido, nombre, email, rol.</li> <li>• LoginRequestDTO: numeroDocumento, contrasena.</li> <li>• JwtResponseDTO: token, rol, numeroDocumento.</li> </ul>				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	1	30/06	30/06
Código				
<p><u>UsuarioMapper</u></p> <pre> public class UsuarioMapper {     public static Usuario usuarioFormDTOaUsuario(UsuarioFormDTO dto, PasswordEncoder passwordEncoder) {         Usuario nuevo = new Usuario();         nuevo.setNumeroDocumento(dto.getNumeroDocumento());         nuevo.setApellido(dto.getApellido());         nuevo.setNombre(dto.getNombre());         nuevo.setEmail(dto.getEmail());         nuevo.setRol(dto.getRol());         nuevo.setContrasena(passwordEncoder.encode(dto.getContrasena()));         return nuevo;     }      public static UsuarioDTO usuarioAusuarioDTO(Usuario usuario) {         UsuarioDTO dto = new UsuarioDTO();         dto.setNumeroDocumento(usuario.getNumeroDocumento());         dto.setApellido(usuario.getApellido());         dto.setNombre(usuario.getNombre());         dto.setEmail(usuario.getEmail());         dto.setRol(usuario.getRol());         return dto;     } } </pre> <p><u>UsuarioFormDTO</u></p> <p>@Getter @Setter</p> <pre> public class UsuarioFormDTO {     private String numeroDocumento;     private String apellido;     private String nombre;     private String email;     private String rol; } </pre>				

```

    private String contrasena;
}

UsuarioDTO
@Getter
@Setter
public class UsuarioDTO {
    private String numeroDocumento;
    private String apellido;
    private String nombre;
    private String email;
    private String rol;
}

LoginRequestDTO
@Getter
@Setter
public class LoginRequestDTO {
    private String numeroDocumento;
    private String contrasena;
}

JwtResponseDTO
@Getter
@Setter
public class JwtResponseDTO {
    private String numeroDocumento;
    private String rol;
    private String token;
}

```

## HISTORIA 12: MODIFICAR LOS DATOS DE UN USUARIO

### TAREA 12.1: CREAR ENDPOINT PARA MODIFICACIÓN DE USUARIO

Descripción				
Crear endpoint en UsuarioController.java (PUT /api/usuarios/{numeroDocumento}).				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	1	30/06	30/06
Código				
<pre> @PutMapping("/{numeroDocumento}") public ResponseEntity&lt;UsuarioDTO&gt; modificarUsuario(@RequestBody UsuarioFormDTO usuarioFormDTO){     UsuarioDTO usuarioModificado = usuarioService.actualizarUsuario(usuarioFormDTO);     return new ResponseEntity&lt;&gt;(usuarioModificado, HttpStatus.OK); } </pre>				

## TAREA 12.2: IMPLEMENTAR SERVICE Y REPOSITORY PARA MODIFICAR USUARIOS

## Descripción

Declarar el método Usuario findByNumeroDocumento(String numeroDocumento); en la interfaz UsuarioRepository.java.

Luego, en UsuarioServiceImpl.java, se debe implementar el método modificarUsuario. Este método recibirá el número de documento y un DTO con los datos actualizados, buscará al usuario con el método del repositorio y, si lo encuentra, mapeará los campos permitidos (nombre, apellido, email) a la entidad existente, excluyendo la contraseña.

Finalmente, persistir los cambios con usuarioRepository.save() y retornar un DTO con la información actualizada. Se debe lanzar una excepción UsuarioNoEncontradoException si el usuario no existe.

## Persona asignada

## TP estimados

## TP reales

## Inicio

## Fin

Gonzalo Gaitán

3

2

02/07

02/07

## Código

Repositorio:

```
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
    Usuario findByNumeroDocumento(String numeroDocumento);
}
```

Servicio:

## @Override

```
public UsuarioDTO actualizarUsuario(UsuarioFormDTO usuarioFormDTO) {
    // Busca el usuario por número de documento
    Usuario usuario =
    usuarioRepository.findByNumeroDocumento(usuarioFormDTO.getNumeroDocumento());
    if (usuario == null) {
        // Lanza excepción si no existe el usuario
        throw new UsuarioNoEncontradoException("No se encontró un usuario con el número de documento:
" + usuarioFormDTO.getNumeroDocumento());
    }

    // Actualiza los campos permitidos
    usuario.setNombre(usuarioFormDTO.getNombre());
    usuario.setApellido(usuarioFormDTO.getApellido());
    usuario.setEmail(usuarioFormDTO.getEmail());
    usuario.setRol(usuarioFormDTO.getRol());
    // No se modifica la contraseña

    // Guarda los cambios y retorna el DTO actualizado
    Usuario actualizado = usuarioRepository.save(usuario);
    return UsuarioMapper.usuarioAUsuarioDTO(actualizado);
}
```

TAREA 12.3: CREAR FORMULARIO DE MODIFICACIÓN DE USUARIOS**Descripción**

Implementar formulario de modificación de usuarios copiando el diseño del formulario de alta de usuario. Crear campo de búsqueda de usuario en base al DNI, agregar función para consultar usuario por DNI mediante GET a /api/usuarios/id/{dni}.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	2	2	02/07	02/07

**Interfaz**
**HISTORIA 13: INICIAR SESIÓN**TAREA 13.1: IMPLEMENTAR USERDETAILSSERVICE**Descripción**

Crear una clase que implemente la interfaz UserDetailsService de Spring. Su método loadUserByUsername buscará un Usuario por DNI en UsuarioRepository.

Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	1	1	01/07	01/07

**Código**

```
@Service
public class UsuarioDetailsService implements UserDetailsService {

    private final UsuarioRepository usuarioRepository;
```

```

public UsuarioDetailsService(UsuarioRepository usuarioRepository) {
    this.usuarioRepository = usuarioRepository;
}

@Override
public UserDetails loadUserByUsername(String username){ //el username será el numeroDocumento
    Usuario usuario = usuarioRepository.findByNumeroDocumento(username);
    if (usuario == null) throw new UsuarioNoEncontradoException("Usuario no encontrado con documento
" + username);
    return usuario;
}
}

```

### TAREA 13.2: CONFIGURAR SPRING SECURITY

Descripción				
Crear una clase SecurityConfig.java. Configurar el SecurityFilterChain para proteger los endpoints específicos para un ADMIN. Definir el Bean de PasswordEncoder, usando BCryptPasswordEncoder.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Marcos Debona	3	6	01/07	02/07
Código				
<pre> @Configuration @EnableWebSecurity public class SecurityConfig {      private final UsuarioDetailsService usuarioDetailsService;     private final JwtAuthenticationFilter jwtAuthenticationFilter;      public SecurityConfig(UsuarioDetailsService usuarioDetailsService, JwtAuthenticationFilter jwtAuthenticationFilter) {         this.usuarioDetailsService = usuarioDetailsService;         this.jwtAuthenticationFilter = jwtAuthenticationFilter;     }      @Bean     public PasswordEncoder passwordEncoder() {         return new BCryptPasswordEncoder();     }      @Bean     public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {         return config.getAuthenticationManager();     }      @Bean     @Profile("dev")     public SecurityFilterChain devSecurityFilterChain(HttpSecurity http) throws Exception {         http             .csrf(csrf -&gt; csrf.disable()) </pre>				

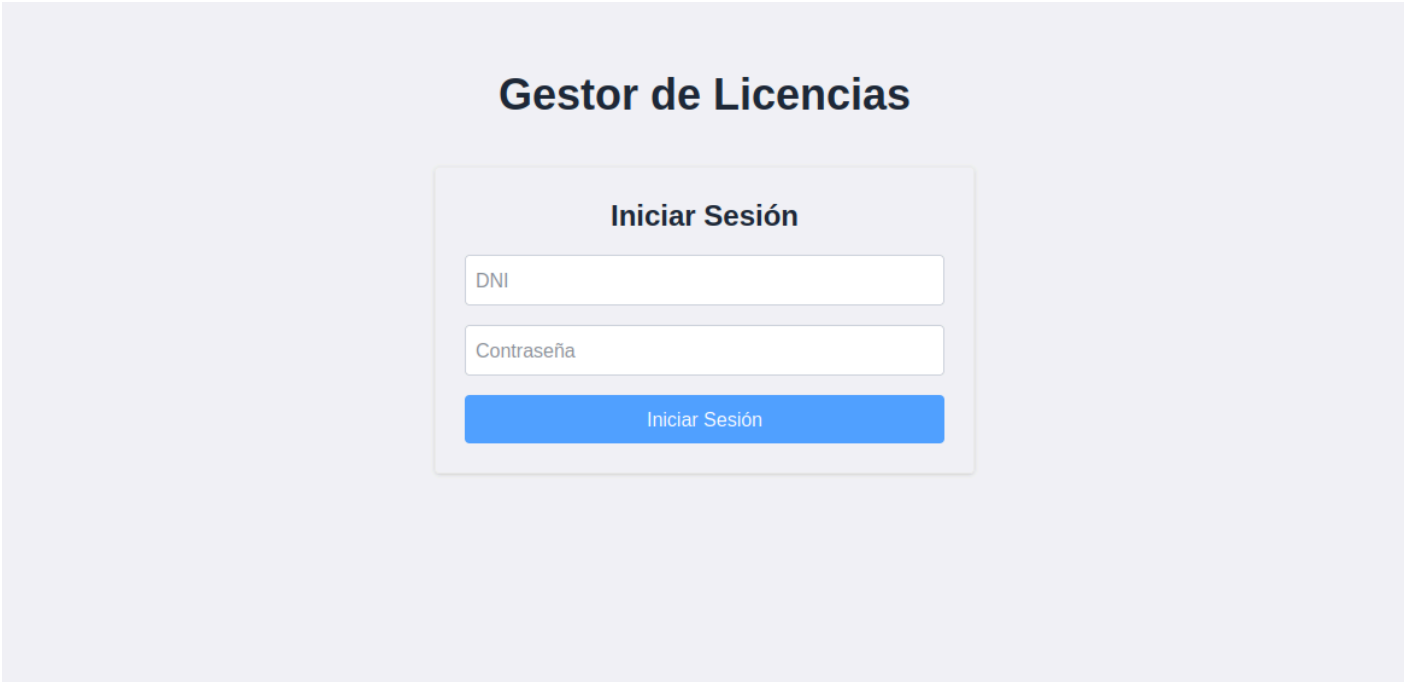
```

        .sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authorizeHttpRequests(auth -> auth
            .anyRequest().permitAll()
        );
    return http.build();
}

@Bean
@Profile("prod")
public SecurityFilterChain prodSecurityFilterChain(HttpSecurity http) throws Exception {
    http
        .csrf(csrf -> csrf.disable())
        .sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/api/auth/**").permitAll()
            .requestMatchers(HttpMethod.OPTIONS, "/**").permitAll()
            .requestMatchers(HttpMethod.POST, "/api/usuarios").hasRole("ADMIN")
            .requestMatchers(HttpMethod.PUT, "/api/usuarios/**").hasRole("ADMIN")
            .requestMatchers("/api/titulares/**").hasAnyRole("USER", "ADMIN")
            .requestMatchers("/api/licencias/**").hasAnyRole("USER", "ADMIN")
            .anyRequest().authenticated()
        )
        .userDetailsService(usuarioDetailsService)
        .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);
    return http.build();
}
}

```

TAREA 13.3: CREAR LA PÁGINA DE LOGIN

Descripción				
Diseñar página de login, solicitando DNI y contraseña. Redirigir al login en caso de no estar autenticado. Implementar schema para validar longitud de DNI y contraseña. Crear funciones para manipular localStorage y un componente ProtectedRoute para prohibir la navegación en caso de no estar autenticado. Actualizar métodos fetch para enviar el token JWT. Ocultar las funcionalidades de alta y modificación de usuarios para quienes no sean administradores.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	3	5	03/07	03/07
Interfaz				
				

TAREA 13.4: CREAR AUTHCONTROLLER PARA LOGIN/LOGOUT

Descripción				
Crear un AuthController.java con un endpoint POST /api/auth/login. Este endpoint recibirá las credenciales, las autenticará y, si son correctas, generará y devolverá un Token JWT.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Gonzalo Gaitán	3	3	03/07	03/07
Código				
<pre>@RestController @RequestMapping("/api/auth") public class AuthController {      private final AuthenticationManager authenticationManager;     private final UsuarioDetailsService usuarioDetailsService;</pre>				

```

private final JwtUtil jwtUtil;

public AuthController(AuthenticationManager authenticationManager,
    UsuarioDetailsService usuarioDetailsService,
    JwtUtil jwtUtil) {
    this.authenticationManager = authenticationManager;
    this.usuarioDetailsService = usuarioDetailsService;
    this.jwtUtil = jwtUtil;
}

@PostMapping("/login")
public ResponseEntity<JwtResponseDTO> login(@RequestBody LoginRequestDTO loginRequest) {
    try {
        // Autentica al usuario con el número de documento y contraseña
        authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(
                loginRequest.getNumeroDocumento(),
                loginRequest.getContrasena()
            )
        );

        // Carga los detalles del usuario autenticado
        UserDetails userDetails =
        usuarioDetailsService.loadUserByUsername(loginRequest.getNumeroDocumento());
        Usuario usuario = (Usuario) userDetails;

        // Genera el token JWT para el usuario autenticado
        String token = jwtUtil.generateToken(userDetails);

        // Prepara la respuesta con el token, número de documento y rol
        JwtResponseDTO response = new JwtResponseDTO();
        response.setNumeroDocumento(usuario.getNumeroDocumento());
        response.setRol(usuario.getRol());
        response.setToken(token);

        // Devuelve la respuesta exitosa con el JWT
        return ResponseEntity.ok(response);

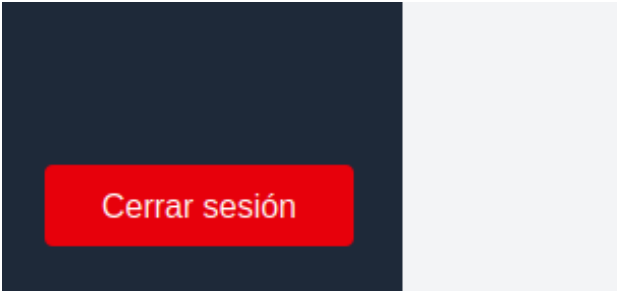
    } catch (BadCredentialsException e) {
        // Si las credenciales son incorrectas, retorna 401
        return ResponseEntity.status(401).build();
    } catch (Exception e) {
        // Para otros errores, retorna 500
        return ResponseEntity.status(500).build();
    }
}
}

```



# HISTORIA 14: CERRAR SESIÓN

## TAREA 14.1: AÑADIR BOTÓN DE LOGOUT

Descripción				
Añadir botón de logout en la parte inferior de la barra lateral de navegación. Asegurarse de borrar los tokens y redirigir al login.				
Persona asignada	TP estimados	TP reales	Inicio	Fin
Santino Paggi	1	3	03/07	03/07
Interfaz				
				

## GRÁFICO BURN DOWN

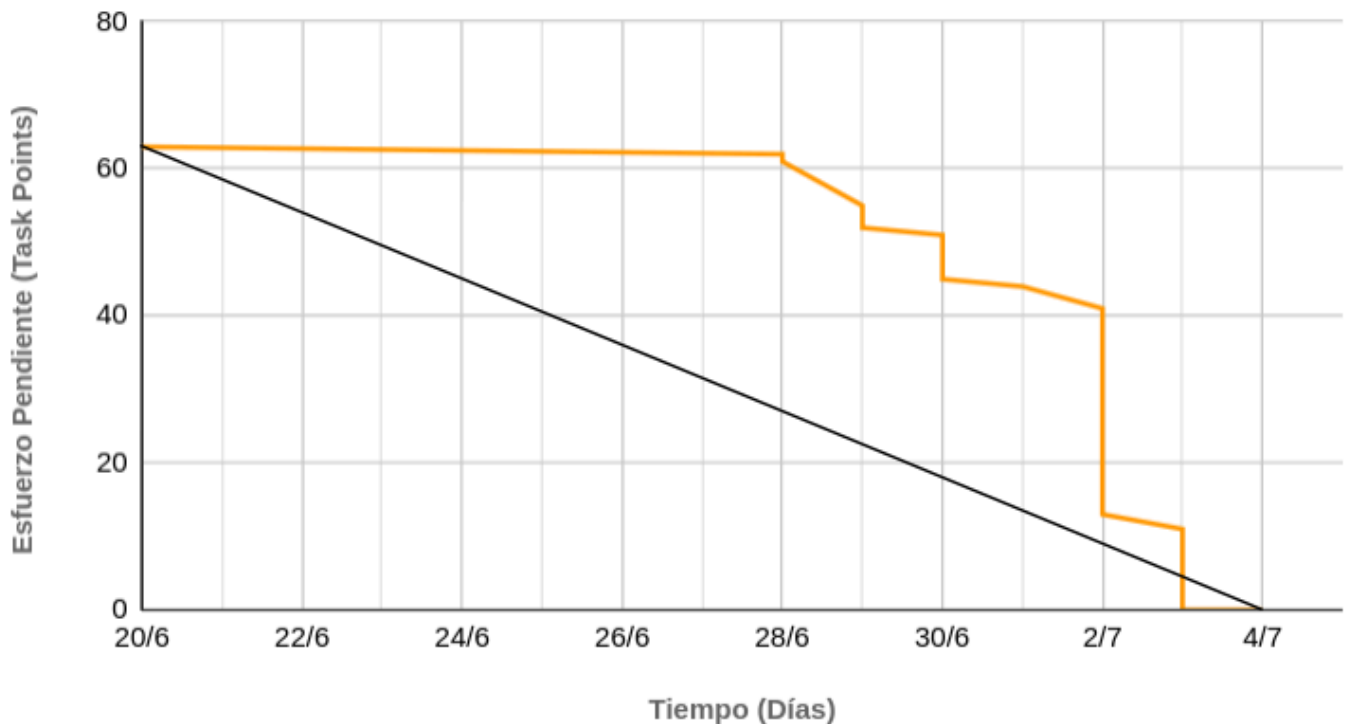
FECHA	TASK POINTS REAL RESTANTE	TASK POINTS IDEAL RESTANTE
20/06	63	59
21/06	63	55
22/06	63	50
23/06	63	46
24/06	63	42
25/06	63	38
26/06	63	34
27/06	63	29
28/06	55	25
29/06	52	21
30/06	45	17
01/07	44	13
02/07	19	8
03/07	0	4

04/07

0

0

## Burn-Down Sprint 2



A pesar de lo concluído el sprint anterior, los compromisos académicos implicaron que el desarrollo del segundo sprint se concentrara en la última semana de este, lo que muestra una progresión aún más alejada de la ideal que la vista en el primer sprint.

### DESCRIPCIÓN DE CADA DAILY MEETING

#### DAILY MEETING 28/06:

Este daily meeting se llevó a cabo con el objetivo de ver la situación y disponibilidad de tiempo que tenía cada integrante, para así ver cómo podíamos organizarnos durante esta semana tan repleta de responsabilidades, para así completar el sprint satisfactoriamente.

#### DAILY MEETING 02/07:

En esta daily meeting se revisó el progreso individual. Se confirmó que gran parte del backend estaba desarrollado, con los endpoints principales y la lógica de negocio implementados. Sin embargo, se detectó un retraso significativo en las tareas de frontend, las cuales aún restaban en su mayoría.

#### DAILY MEETING 03/07:

Este jueves se discutió el módulo de login. Las cuestiones clave fueron el manejo de contraseñas, incluyendo el uso de algoritmos de hashing robustos. Se convalidó el uso de BCrypt.

## **REVISIÓN DEL SPRINT**

- Se mostró el sistema desarrollado al PO.
- Testeó ciertas funcionalidades básicas propias del sprint, funcionaban correctamente exceptuando la página de modificar titular (solucionado poco después). El product owner hizo recomendaciones sobre cómo realizar mejores presentaciones para no encontrarnos en situaciones comprometedoras como esta.
- Se comentó sobre cuestiones de organización y manejo de tiempos.
- Se dió la aprobación respecto a la lógica de negocio y funcionamiento de lo involucrado a las historias desarrolladas en el sprint.

## **RETROSPECTIVA DEL SPRINT**

- En este sprint realizamos una planificación más detallada que nos ayudó de manera considerable a la hora de realizar las nuevas tareas. Las tareas fueron escritas de una manera más refinada definiendo el manejo de datos, métodos, DTOs, controllers, entidades, relación entre entidades, etc.
- Los procesos se realizaron de manera más rápida por la experiencia a comparación del anterior sprint, esto nos llevó a concluir sobre que sería mejor sobrecargar menos las primeras iteraciones, con el objetivo de que estas sirvan como toma de contacto con las tecnologías y el know how del negocio, para luego en las siguientes ir progresivamente aumentando la cantidad de story points disponibles en cada una.

## CONCLUSIONES

Realmente creemos que nuestra experiencia con la metodología Scrum en este proyecto nos brindó aprendizajes valiosos, especialmente como primera inmersión en su aplicación práctica.

Entendimos que la dedicación de tiempo a la buena escritura de historias y tareas es fundamental. Inicialmente, la ausencia de modelos de datos y la falta de granularidad en las historias de usuario generaron confusión. Comprendimos la importancia de definir y estimar exhaustivamente las historias y tareas, incluso si parecen breves, ya que su calidad impacta directamente en la eficiencia del desarrollo y ayuda a evitar ambigüedades. Esto se evidenció en la notable mejora entre el primer y segundo sprint, donde la mayor claridad de las tareas agilizó el trabajo de los desarrolladores.

Nos encontramos con que la colaboración con el Product Owner fue esencial para entender las necesidades del cliente y despejar dudas sobre el dominio del problema. Asimismo, la falta de estándares claros para llevar a cabo las tareas fue un desafío que debimos compensar con daily meetings o spikes (tareas de investigación), para así alinear al equipo, comentar nuestra situación y resolver dudas e incompatibilidades.

En cuanto a las herramientas, utilizamos Jira para la gestión del proyecto. Aunque su implementación de Scrum no se ajusta fielmente a la metodología, su tablero Kanban integrado resultó útil para visualizar el estado de las historias. Los gráficos burn-down fueron una gran herramienta a la hora de monitorear el estado actual de cada sprint, ofreciendo una visión realista del progreso de estos y, por consiguiente, facilitando la toma de decisiones basada en esta información.

En retrospectiva, esta asignatura y la implementación práctica que nos requirió nos resultaron sumamente enriquecedoras. Más allá de la teoría, pudimos aplicar y afianzar conceptos y formas de trabajo propias de entornos profesionales, lo cual representa un aprendizaje invaluable para futuros proyectos.