

Universidad ORT Uruguay

Facultad de Ingeniería

Portal DGC: Digitalización de procesos de inscripción a llamados públicos del Estado

Entregado como requisito para la obtención del título

Licenciado en Ingeniería de Software

Gabriel Lutz

Tutor: Alejandro Adorjan

Diciembre 2025

Declaratoria de Autoría

Quien suscribe, Gabriel Lutz, declaro que el trabajo que se presenta en esa obra es de mi propia mano. Puedo asegurar que:

- La obra fue producida en su totalidad mientras realizaba el proyecto para la obtención del título de Licenciatura en Ingeniería de Software
- Cuando he consultado el trabajo publicado por otros, lo he atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente mía;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, he explicado claramente qué fue contribuido por otros, y qué fue contribuido por mí;
- En la obra, he acusado recibo de las ayudas recibidas;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.



Gabriel Lutz

10/12/2025

Dedicatoria

A Andrea, mi mujer, mi compañera, mi cable a tierra. Por bancarme en las noches interminables frente a la computadora, por entender cuando la carrera me demandaba todo y por recordarme quién soy cuando yo lo olvidaba. Este logro es nuestro, porque vos lo sostuviste tanto como yo. Gracias por creer, por esperar y por quedarte. Siempre.

A mi padre Adolfo, por ser mi primer maestro y mi guía constante. Por confiar en mí sin condiciones, por apoyarme en cada decisión y por enseñarme que los sueños se construyen con esfuerzo, perseverancia y dignidad. Este logro también es tuyo, papá, porque vos fuiste quien me enseñó a no rendirme y a ser un hombre de bien. Gracias por estar siempre.

A mis hermanos Adrián y Daniel. A vos, Daniel, aunque un océano nos separe, y a vos, Adrián, que estás más cerca, ambos son parte de quien soy y de este camino.

A Tomás, hermano que la carrera me regaló, por demostrarme que la familia se elige y se construye con lealtad y cariño.

A Christian, hermano de la vida, por aquella charla cuando más la necesitaba. Cuando estaba en una carrera contra el reloj, me mostraste que ese no era el camino, cambié la mirada, disfruté el recorrido. Cambiaste mi perspectiva y, con ella, todo lo que vino después.

A Rodrigo, Gastón, Nacho, Agustín, y Manuel, compañeros que se transformaron en amigos, hoy compartimos mucho más que lo académico.

A mi madre María, que me acompaña desde otro lugar. Tu presencia vive en cada paso que doy y los valores que sembraste en mí florecen cada día. Soy quien soy por vos, mamá. Este logro es tuyo también, porque todo lo que he construido nace de las raíces que plantaste.

Ojalá puedas verme desde donde estés, porque sé que estarías orgullosa.

A todos ustedes, que hicieron posible este sueño. Este logro también les pertenece.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que hicieron posible la realización de este proyecto, que marca el cierre de una etapa académica y el comienzo de una nueva como profesional en el área de la Ingeniería de Sistemas.

Deseo expresar un especial reconocimiento a mi tutor, el Ing. Alejandro Adorjan MSc., por su guía, compromiso y acompañamiento durante todo el proceso. Su orientación técnica y dedicación fueron fundamentales para superar obstáculos y alcanzar los objetivos propuestos.

También quiero agradecer a la coordinadora de la carrera, Ing. Cecilia Beletti Mg., por su consejo y apoyo permanente durante estos años de formación. Tuve el privilegio de tenerla como docente en Base de Datos II, y su compromiso con los estudiantes ha sido una inspiración a lo largo de mi trayectoria académica.

Extiendo mi agradecimiento a la División de Tecnología de la Información de la Dirección General de Casinos del Estado por brindarme la oportunidad de desarrollar este proyecto en un entorno real y desafiante. A todo el equipo, y especialmente a Fabricio Giordano como encargado del área, por facilitar el espacio y los recursos necesarios para llevar adelante este trabajo.

Finalmente, quiero agradecer a la Universidad ORT Uruguay por la formación integral y de excelencia que me brindó y por fomentar en mí la responsabilidad, la curiosidad y la pasión por la tecnología que hicieron posible la concreción de este proyecto.

A todos ustedes, gracias por acompañarme en este recorrido.

Abstract

El proceso de inscripción a los llamados públicos de la Dirección General de Casinos (DGC) se realiza actualmente mediante mecanismos presenciales o por envío de documentación por correo electrónico, lo que genera demoras considerables, duplicación de tareas administrativas, errores de digitación y dificultades en la trazabilidad de las postulaciones. Además, el tribunal evaluador enfrenta desafíos significativos al realizar manualmente el cálculo de puntajes y la aplicación de las cuotas de acción afirmativa establecidas por ley, un proceso complejo y propenso a errores.

El presente proyecto tiene como objetivo desarrollar una Prueba de concepto del portal institucional, una plataforma web que digitalice íntegramente el proceso de llamados públicos. La solución permite a los postulantes realizar inscripciones digitales mediante un workflow de seis pasos, garantizando la presentación ordenada, validada y accesible de la información. Incorpora herramientas específicas para el tribunal evaluador, facilitando la calificación de méritos y de pruebas, centralizando la documentación y automatizando el cálculo de puntajes. Implementa mecanismos de generación automática de ordenamientos que apliquen cuotas afirmativas, en consonancia con el marco normativo vigente.

La metodología aplicada combinó un conjunto de técnicas, como entrevistas semiestructuradas con usuarios interesados en el proyecto (postulantes, personal de recursos humanos (RR. HH.) y tribunal evaluador), análisis técnico de portales gubernamentales existentes (Banco de Previsión Social (BPS), Banco de Seguros del Estado (BSE), Uruguay Concursa), y un proceso de diseño iterativo validado tanto con los mismos usuarios participantes como con un usuario experto en Quality Assurance (QA). Además, se aplicó un enfoque de desarrollo ágil utilizando GitHub Projects como tablero Kanban e implementando una arquitectura basada en el patrón de tres capas (presentación, lógica de negocio y acceso a datos).

Los resultados parciales indican la viabilidad técnica de la solución, con los workflows principales de inscripción y evaluación implementados en un prototipo inicial que sigue las prácticas

de la Ingeniería de Software. La realización de este proyecto evidencia un enfoque metodológico y estructurado que permitiría modernizar de manera efectiva el proceso mediante una solución digital integral que garantice eficiencia, transparencia, trazabilidad y accesibilidad universal.

El desarrollo del prototipo presentó desafíos asociados a integrar análisis, diseño y validación en ciclos iterativos, pero permitió obtener retroalimentación temprana de los usuarios y ajustar la solución a sus necesidades reales. Aunque se trata de una versión inicial, su diseño habilita futuras extensiones dentro de la organización orientadas a consolidar una plataforma integral para la gestión eficiente, accesible y transparente de los llamados públicos.

Palabras Clave

Portal de postulaciones, Dirección General de Casinos del Estado, Cuotas de acción afirmativa, Accesibilidad digital.

Glosario

Accesibilidad digital

Conjunto de características y buenas prácticas que permiten que un sitio web o aplicación sea utilizable por todas las personas, incluyendo aquellas con discapacidades visuales, auditivas, motoras o cognitivas.

Afrodescendiente

Persona que se autodefine como descendiente de africanos, categoría protegida por la Ley 19.122 que establece cuotas de reserva en concursos públicos en Uruguay.

AGESIC (Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento)

Organismo público uruguayo responsable de impulsar el desarrollo del gobierno electrónico y la sociedad de la información. Establece estándares y normativas para sistemas del Estado, incluyendo el Decreto 406/022 sobre accesibilidad web.

Angular

Framework de desarrollo web basado en TypeScript, utilizado para crear interfaces de usuario dinámicas y responsivas. En este proyecto se utilizó Angular 19.

API (Application Programming Interface)

Conjunto de definiciones y protocolos que permiten que distintas aplicaciones se comuniquen entre sí. En el Portal DGC se utiliza una API REST para la comunicación entre frontend y backend.

ASP.NET Core

Framework de desarrollo web multiplataforma creado por Microsoft, utilizado para construir el backend del Portal DGC. Se empleó la versión 8.

Autenticación

Proceso mediante el cual el sistema comprueba la identidad de un usuario, generalmente a través de credenciales como usuario y contraseña.

Autorización

Mecanismo que, una vez autenticado el usuario, determina qué recursos, funciones o acciones dentro del sistema le están permitidos según su rol (postulante, tribunal, RRHH).

Backend

Parte de una aplicación que gestiona la lógica del negocio, las bases de datos y los servicios del servidor. Es invisible para el usuario final.

Backlog

Lista priorizada de tareas, requisitos o funcionalidades pendientes de realizar en un proyecto. Funciona como una agenda de trabajo que guía al equipo sobre qué se debe hacer y en qué orden.

C#

Lenguaje de programación desarrollado por Microsoft, orientado a objetos y utilizado para desarrollar el backend del Portal DGC sobre ASP.NET Core.

Constancia

Archivo digital que un postulante adjunta como evidencia para acreditar méritos o requisitos excluyentes declarados en su inscripción.

Criterios de aceptación

Requisitos específicos que una funcionalidad debe satisfacer para que se la considere completa y apta para su implementación en el sistema.

Cuota de acción afirmativa

Porcentaje de posiciones reservadas en un llamado para grupos específicos protegidos por ley (afrodescendientes, personas trans, personas con circunstancias personales especiales).

Decreto 406/022 (AGESIC)

Norma uruguaya que establece requisitos de accesibilidad para sitios web del Estado, basada en WCAG 2.1 nivel AA.

Departamento

División administrativa del Estado uruguayo. En el Portal DGC, cada llamado puede estar asociado a uno o más departamentos donde se ofrecen las posiciones laborales.

Dirección General de Casinos del Estado (DGC)

Organismo estatal uruguayo que administra los casinos públicos del país y es el cliente principal del Portal DGC.

DTO (Data Transfer Object)

Objeto utilizado para transferir datos entre capas de la aplicación o entre el backend y el frontend, sin exponer directamente las entidades del dominio.

Entity Framework

Framework de acceso a datos para .NET que permite trabajar con bases de datos relacionales usando objetos C# en lugar de SQL directo.

ESRE (Especificación de Requerimientos de Software)

Documento que describe de manera formal y detallada los requisitos funcionales y no funcionales del sistema, utilizado como base para el diseño e implementación del Portal DGC.

Evaluación de mérito

Proceso mediante el cual el tribunal asigna puntaje a los méritos declarados por un postulante según los ítems puntuables definidos en el llamado.

Evaluación de prueba

Proceso de asignación de puntaje por parte del tribunal a las pruebas (escritas, orales, prácticas) rendidas por los postulantes.

Frontend

Parte de una aplicación que interactúa directamente con el usuario, incluyendo todo lo que se ve en la pantalla y las interacciones.

GitHub

Plataforma de desarrollo colaborativo que utiliza Git para el control de versiones. Utilizado para

gestionar el código fuente del Portal DGC.

GitHub Projects

Herramienta integrada en GitHub para gestionar proyectos con tableros Kanban, utilizada para organizar el backlog y las tareas del Portal DGC.

Inscripción

Solicitud formal de un postulante para participar en un llamado específico, completada a través del workflow de 6 pasos del Portal DGC.

Ítem puntuable

Elemento definido en un llamado que puede recibir puntaje durante la evaluación de méritos (ej: años de experiencia, títulos académicos, cursos).

Kanban

Metodología ágil visual que utiliza tableros para gestionar el flujo de trabajo, mostrando tareas en columnas como “Por hacer”, “En progreso” y “Completado”.

Ley 18.651

Ley uruguaya de protección integral de personas con discapacidad, que establece cuotas laborales del 4 % en organismos estatales.

Ley 19.122

Ley uruguaya de acciones afirmativas para afrodescendientes, que establece cuotas del 8 % en ingresos a organismos públicos.

Ley 19.684

Ley integral para personas trans, que incluye cuotas laborales del 1 % para esta población en organismos estatales.

Llamado

Concurso público mediante el cual la DGC convoca a postulantes para cubrir posiciones laborales específicas, con requisitos, méritos y pruebas definidos.

Mérito

Atributo o característica del postulante que puede sumar puntaje en la evaluación (experiencia laboral, formación académica, capacitaciones).

Moq

Biblioteca de .NET utilizada para crear objetos simulados (mocks) en pruebas unitarias, permitiendo aislar el código bajo prueba de sus dependencias.

MVP (Minimum Viable Product)

Producto mínimo viable que contiene las funcionalidades esenciales para ser utilizado y validado por usuarios reales.

Ordenamiento

Lista final de postulantes ranqueados según puntaje total, generada automáticamente por el sistema considerando evaluaciones y aplicando lógica de cuotas y desempate.

Overleaf

Plataforma en línea para editar documentos LaTeX de forma colaborativa, utilizada para escribir la documentación académica del proyecto.

Persona trans

Persona cuya identidad de género no coincide con el sexo asignado al nacer, categoría protegida por la Ley 19.684.

Personas con circunstancias personales especiales

Categoría que incluye jefes/as de hogar monoparentales, víctimas de violencia doméstica, personas liberadas y ex tuteladas, protegidos por distintas leyes uruguayas con cuotas del 2% en total.

Postulante

Persona que se registra en el Portal DGC y completa una inscripción para participar en uno o más llamados.

Prueba

Instancia de evaluación formal (escrita, oral o práctica) que el postulante debe rendir como parte del proceso de selección.

Repository Pattern

Patrón de diseño que abstrae el acceso a datos, proporcionando una interfaz uniforme para operaciones CRUD independientemente del mecanismo de persistencia.

Requisito excluyente

Condición obligatoria que el postulante debe cumplir para participar en un llamado (ej: título habilitante, años mínimos de experiencia).

SQL Server

Sistema de gestión de bases de datos relacional desarrollado por Microsoft, utilizado para almacenar toda la información del Portal DGC.

Story points

Unidad de medida utilizada para estimar el esfuerzo relativo de una tarea o historia de usuario, considerando complejidad, esfuerzo e incertidumbre.

Tribunal

Grupo de evaluadores designados para calificar méritos, pruebas y generar el ordenamiento final de postulantes en un llamado específico.

TypeScript

Lenguaje de programación basado en JavaScript con tipado estático, utilizado como base para el desarrollo con Angular.

Unit of Work

Patrón de diseño que coordina transacciones y cambios en múltiples repositorios, asegurando consistencia de datos.

URCDP (Unidad Reguladora y de Control de Datos Personales)

Organismo uruguayo encargado de proteger los datos personales y controlar el cumplimiento

de la normativa de protección de datos. Relevante para el Portal DGC debido al manejo de información sensible de postulantes.

User Story (Historia de Usuario)

Descripción breve de una funcionalidad desde la perspectiva del usuario final, utilizada como unidad de trabajo en metodologías ágiles.

Visual Studio 2022

Entorno de desarrollo integrado (IDE) de Microsoft utilizado para escribir, depurar y ejecutar el código C# del backend.

WCAG 2.1 AA

Pautas de Accesibilidad para el Contenido Web, nivel AA de conformidad, que establecen criterios técnicos para hacer contenido digital accesible a personas con discapacidades.

Workflow de inscripción

Proceso guiado de 6 pasos implementado en el Portal DGC para completar una inscripción: datos personales, autodefinición de ley, requisitos excluyentes, méritos, apoyos necesarios y confirmación.

xUnit

Framework de pruebas unitarias para .NET utilizado en el Portal DGC para validar la lógica de negocio de servicios y controladores.

Índice general

1. Introducción	19
1.1. Motivación	19
1.2. Marco Teórico	20
1.3. Metodología	22
2. Objetivos	25
2.1. Objetivo General	25
2.2. Objetivos Específicos	25
2.3. Alineación de Objetivos con la Problemática Identificada	34
2.4. Criterios de cumplimiento	34
3. Definición del problema	37
3.1. Problemática General	37
3.2. Manifestaciones del Problema	37
3.3. Relevancia e Impacto	39
3.4. El Caso de la Dirección General de Casinos	39
3.5. Oportunidad de Solución	40
4. Antecedentes	41

4.1.	Portales de Concursos Públicos en Uruguay	41
4.2.	Apunte Diferencial del Portal DGC	44
5.	Marco teórico	47
5.1.	Transformación Digital del Estado	47
5.2.	Arquitecturas de Software para Aplicaciones Web	49
5.3.	Tecnologías y Plataformas de Desarrollo	51
5.4.	Accesibilidad Digital	55
5.5.	Marco Normativo y Legal	57
5.6.	Contexto Institucional: La Dirección General de Casinos	59
5.7.	Estado del Arte: Portales Gubernamentales en Uruguay	59
5.8.	Síntesis: Fundamentación Teórica del Portal DGC	60
6.	Metodología	62
6.1.	Enfoque Metodológico General	62
7.	Procedimiento Metodológico	63
7.1.	Fase 1: Elicitación y Relevamiento de Requisitos	63
7.2.	Fase 2: Documentación de Requisitos	70
7.3.	Fase 3: Diseño de Arquitectura y Modelo de Dominio	75
7.4.	Fase 4: Prototipado de Interfaces	89

7.5. Fase 5: Desarrollo Iterativo	100
7.6. Fase 6: Testing y Aseguramiento de Calidad	107
7.7. Fase 7: Documentación Técnica	109
7.8. Uso de herramientas de Inteligencia Artificial	111
8. Análisis de Resultados	114
8.1. Alcance de la Implementación	114
8.2. Validación de Workflows Principales	116
8.3. PoC Implementado	118
8.4. Verificación Técnica del Sistema	127
8.5. Validación con Usuarios Reales	135
8.6. Cumplimiento Normativo	140
8.7. Limitaciones y Trabajo a Futuro	141
8.8. Resultados	141
9. Lecciones Aprendidas	143
10. Conclusiones	144
Bibliografía	157
A. Apéndice A: Evidencia de la Plataforma Uruguay Concursa	158

B. Apéndice B: Evidencia del Portal de Concursos del BPS	160
C. Apéndice C: Evidencia del Portal del BSE	161
D. Apéndice D: Sistemas Comerciales de Reclutamiento para el Sector Público	162
E. Apéndice E: Marco Conceptual E-Recruitment	165
F. Apéndice F: Cuotas de Acción Afirmativa - Perspectiva Comparada	171
G. Apéndice G: Trabajos Integradores en Ingeniería de Software	174
H. Apéndice H: Detalle de Requisitos Implementados	176
I. Apéndice I: Especificaciones Técnicas de Arquitectura	186
J. Apéndice J: Cobertura Detallada de Pruebas Unitarias	209
K. Apéndice K: Plan de Validación de RNF Proyectado	272
L. Apéndice L: Trabajo Pendiente Priorizado	291
M. Apéndice M: Plan de pruebas	309
N. Apéndice N: ESRE	317
Ñ. Apéndice Ñ: Casos de Uso	345

Introducción

1.1 Motivación

Los procesos de selección de personal en organismos públicos constituyen un pilar fundamental para garantizar la transparencia, la equidad y la profesionalización de la Administración del Estado. Sin embargo, la gestión tradicional de llamados públicos presenta limitaciones significativas que afectan tanto a los aspirantes como a las instituciones responsables de conducir estos procesos.

La Dirección General de Casinos (DGC) [1] del Ministerio de Economía y Finanzas (MEF) [2] de Uruguay enfrenta actualmente esta problemática en su proceso de llamados a concursos públicos. El procedimiento vigente se desarrolla mediante dos modalidades: presentación presencial de formularios impresos en oficinas de Recursos Humanos (RR. HH.) de la DGC [1] o envío de documentación digitalizada por correo electrónico a casillas institucionales. Ambas alternativas generan demoras administrativas considerables, duplicación de tareas, errores de digitación, dificultades en la trazabilidad de postulaciones y riesgos de incumplimiento de normativas sobre protección de datos personales[3] y acciones afirmativas [4].

El último llamado público realizado por la DGC [1] para el cargo de Fiscal III [2], publicado en el portal gubernamental *gub.uy* [5] [6], ilustra estas limitaciones. Los postulantes debían descargar manualmente un formulario en formato Excel, completarlo, firmarlo, escanearlo en Portable Document Format (PDF) y enviarlo por correo electrónico a *concursofiscal@casinos.gub.uy*. Posteriormente, recibían una confirmación manual dentro de 72 horas hábiles con un código de identificación. Este proceso, si bien permite cierto nivel de digitalización, continúa siendo esencialmente manual y susceptible a errores humanos en cada etapa.

Para el tribunal evaluador, las dificultades son aún mayores. El cálculo manual de puntajes, la aplicación de cuotas de acción afirmativa establecidas por ley (8 % para personas afrodes-

cendientes según Ley 19.122 [7], 1% para personas trans según Ley 19.684 [8], y 4% para personas con discapacidad según Ley 18.651 [9]), y la generación de listas de prelación son tareas complejas que actualmente se realizan mediante planillas de cálculo sin automatización, aumentando significativamente el riesgo de errores e inconsistencias.

La motivación central de este proyecto radica en modernizar integralmente este proceso mediante una solución digital que elimine las ineficiencias operativas, garantice el cumplimiento normativo, fortalezca la transparencia institucional y mejore la experiencia tanto de postulantes como de evaluadores. La digitalización completa del workflow de llamados públicos no solo representa una mejora administrativa, sino también un paso fundamental hacia la transformación digital del Estado uruguayo, alineándose con los objetivos de gobierno electrónico promovidos por la Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento (AGESIC) [10] .

1.2 Marco Teórico

La Dirección General de Casinos

La Dirección General de Casinos es un organismo dependiente del Ministerio de Economía y Finanzas del Uruguay, responsable de la administración y fiscalización de los casinos y salas de esparcimiento del Estado. Como organismo público, está sujeto a los procedimientos establecidos por la normativa nacional para la selección y contratación de personal, incluyendo el régimen de concursos públicos de oposición y de méritos.

El organismo gestiona múltiples establecimientos distribuidos por todo el Uruguay, lo que requiere periódicamente la incorporación de personal para diversos roles administrativos y operativos. El cargo de Fiscal de Casinos, objeto del último llamado analizado [11], constituye una posición clave dentro de la estructura organizacional, con responsabilidades que abarcan tareas administrativas, de fiscalización en sala de juegos, de cajero, de apoyo y de atención al público en régimen de polifuncionalidad.

Proceso Actual de Llamados Públicos

El proceso tradicional de llamados públicos en la DGC [1] se estructura en las siguientes etapas:

1. **Publicación del llamado:** Las bases del concurso se publican en el portal institucional [6], especificando requisitos excluyentes, valorables, descripción del puesto, etapas de evaluación y documentación requerida.
2. **Período de postulación:** Los aspirantes descargan los formularios de inscripción, los completan manualmente, adjuntan documentación respaldatoria digitalizada y los remiten por correo electrónico. Alternativamente, pueden presentarse presencialmente en oficinas de RR. HH. de la DGC [1].
3. **Confirmación de recepción:** El personal administrativo procesa manualmente cada postulación, verifica la completitud formal y emite confirmaciones individuales dentro de 72 horas hábiles.
4. **Preselección:** Se realiza un ordenamiento aleatorio ante escribano público, separando a los postulantes en universos según las cuotas de acción afirmativa y el departamento de destino.
5. **Evaluación:** Se desarrollan pruebas de conocimiento (teóricas y prácticas), evaluación de méritos y antecedentes y entrevistas personales.
6. **Generación de listas de prelación:** El tribunal calcula puntajes totales, aplica cuotas legales y confecciona los ordenamientos finales, todo mediante procedimientos manuales en hojas de cálculo.

Este modelo actual presenta ineficiencias sistémicas: duplicación de datos entre formularios físicos y digitales, consolidación manual de información dispersa en múltiples correos electrónicos, riesgo de pérdida de documentación, imposibilidad de consultar el estado de postulación en tiempo real y ausencia de trazabilidad automatizada del proceso.

Análisis de Soluciones Existentes

Durante la fase de relevamiento se analizaron portales gubernamentales similares en Uruguay, específicamente los sistemas de concursos del Banco de Previsión Social (BPS) [12], Banco de Seguros del Estado (BSE)[13] y la plataforma Uruguay Concursa [14] de la Oficina Nacional del Servicio Civil (ONSC) [15]. Este análisis reveló patrones tecnológicos recurrentes: uso de frameworks JavaScript modernos (Angular [16], React [17]) para interfaces de usuario, Application Programming Interface (APIs) RESTful [18] para comunicación backend, bases de datos relacionales (PostgreSQL [19], SQL Server [20]) para persistencia, y formularios progresivos tipo workflow para guiar a los postulantes.

Sin embargo, ninguno de estos sistemas implementa la automatización completa del cálculo de cuotas de acción afirmativa ni la generación algorítmica de ordenamientos finales, manteniendo estos procesos en etapas manuales o semiautomatizadas. Esta carencia representa una oportunidad para que el Portal DGC incorpore una innovación significativa en un aspecto crítico del proceso.

1.3 Metodología

El desarrollo del Portal DGC se estructuró siguiendo principios de Ingeniería de Software y metodologías ágiles, organizándose en fases iterativas e incrementales que permitieron la validación continua con usuarios y stakeholders.

Fase de Elicitación y Relevamiento

La primera fase se dedicó a comprender profundamente las necesidades del proceso actual mediante técnicas cualitativas:

- **Entrevistas semiestructuradas:** Se realizaron entrevistas con postulantes que habían participado en llamados anteriores, personal de RR. HH. responsable de gestionar las inscripciones, miembros del tribunal de selección y usuarios con necesidades especiales de

accesibilidad. Las entrevistas revelaron necesidades convergentes: los postulantes demandaban procesos completamente digitales con confirmación inmediata, RR. HH. requería automatización de tareas repetitivas y verificación de requisitos; el tribunal necesitaba herramientas sistematizadas para la evaluación y el cálculo de puntajes.

- **Ingeniería inversa de portales existentes:** Se realizó un análisis técnico de sistemas similares mediante la inspección del código fuente, el análisis de las peticiones de Hypertext Transfer Protocol (HTTP [21]) a las APIs [18] y la navegación completa de los flujos de usuario. Se identificaron tecnologías prevalentes, patrones de diseño efectivos (validación en tiempo real, formularios progresivos, confirmación automática) y oportunidades de mejora (automatización de cuotas, interfaces más accesibles).

Fase de Especificación y Diseño

En esta fase se elaboró la Especificación de Requerimientos de Software (ESRE)[N](#) siguiendo estándar IEEE 830-1998 [22], documentando requerimientos funcionales (RF) y no funcionales (RNF). Se diseñaron casos de uso que cubren los flujos principales y alternativos, desde el registro de postulante hasta la generación de los ordenamientos finales.

Paralelamente, se desarrolló la arquitectura del sistema, adoptando el patrón de tres capas (presentación, lógica de negocio, acceso a datos), seleccionando el stack tecnológico ASP.NET Core [23] para el backend, Angular [16] para el frontend, Entity Framework Core [24] para el mapeo objeto-relacional (ORM) y SQL Server [20] como motor de base de datos. Se diseñaron diagramas de arquitectura, de despliegue, de dominio y de flujos de proceso, estableciendo una base inicial que puede ampliarse hacia un sistema de producción.

Fase de Desarrollo Iterativo

El desarrollo se condujo mediante metodología ágil con iteraciones semanales, utilizando GitHub Projects [25] como tablero Kanban para la gestión visual del progreso. Se implementaron los workflows principales: inscripción del postulante (formulario progresivo de 6 pasos con

validación en tiempo real) y el panel del tribunal (evaluación de méritos, pruebas y generación de ordenamientos aplicando las cuotas legales [4]).

Fase de Validación y Refinamiento

Se desarrolló un prototipo funcional que fue validado con usuarios representativos, incorporando retroalimentación iterativa para refinar las interfaces y los flujos de interacción. Este enfoque evitó rehacer el desarrollo por errores de diseño, confirmando tempranamente la adecuación de la solución a las necesidades reales.

Objetivos

Este capítulo establece el propósito fundamental del proyecto y desglosa las metas que guiaron el desarrollo del Portal DGC. Los objetivos se definen en función de la problemática identificada y se estructuran para proporcionar un marco claro de alcance y de resultados esperados.

2.1 Objetivo General

Desarrollar un portal web institucional bajo el dominio gub.uy [5] que digitalice integralmente el proceso de llamados públicos de la Dirección General de Casinos del Estado, modernizando y sistematizando los procedimientos de inscripción, evaluación y selección de personal, con el fin de garantizar eficiencia operativa, transparencia en todas las etapas del proceso, cumplimiento riguroso del marco normativo vigente, y accesibilidad universal para todos los ciudadanos que deseen participar en los concursos públicos.

El portal busca transformar un proceso actualmente manual, basado en procedimientos presenciales o envío de documentación por correo electrónico, en un sistema completamente digital que elimine las ineficiencias detectadas, reduzca tiempos administrativos, minimice el riesgo de errores, y fortalezca la confianza institucional en la gestión de recursos humanos del organismo.

2.2 Objetivos Específicos

Para alcanzar el objetivo general, se establecieron los siguientes objetivos específicos que articulan las distintas dimensiones del proyecto:

OE1: Digitalizar el Proceso de Inscripción

Implementar un workflow digital progresivo de seis pasos que permita a los postulantes completar su inscripción a llamados públicos de forma completamente remota, sin necesidad de

acerarse presencialmente a las oficinas de la DGC [1]. El sistema debe:

- Proporcionar una interfaz intuitiva y guiada que organice la recolección de información en etapas lógicas y secuenciales
- Validar automáticamente la información ingresada en cada paso, proporcionando retroalimentación inmediata sobre errores o inconsistencias
- Permitir la carga de documentación probatoria en formato digital (PDF), eliminando la necesidad de documentos físicos
- Generar un código único de confirmación al completar la inscripción, proporcionando al postulante evidencia inmediata de que su solicitud fue recibida correctamente
- Mantener un registro completo y trazable de todas las inscripciones desde su inicio hasta su finalización

Este objetivo responde directamente a una de las principales demandas identificadas durante las entrevistas con stakeholders: la necesidad de contar con un proceso accesible desde cualquier lugar y en cualquier momento, que elimine las barreras geográficas y temporales del procedimiento actual.

OE2: Facilitar la Evaluación por parte del Tribunal

Desarrollar un módulo especializado (backoffice) que proporcione al tribunal evaluador herramientas sistematizadas para desempeñar sus funciones de calificación de méritos, registro de resultados de pruebas y generación de ordenamientos finales. El sistema debe:

- Centralizar toda la documentación de cada postulante en una vista unificada que facilite la evaluación integral
- Proporcionar formularios estructurados para el registro de calificaciones de méritos y pruebas, con validaciones que aseguren coherencia de datos

- Calcular automáticamente los puntajes parciales y totales a medida que se ingresan las calificaciones individuales
- Ofrecer capacidades de filtrado, búsqueda y ordenamiento que permitan al tribunal gestionar eficientemente listas de cientos de inscripciones
- Generar reportes y estadísticas en tiempo real sobre el avance del proceso de evaluación

Este objetivo responde a la necesidad expresada por los miembros de tribunales de contar con herramientas que reduzcan el trabajo manual repetitivo, minimicen el riesgo de errores en los cálculos de puntajes y les permitan concentrarse en la evaluación de los postulantes.

OE3: Automatizar el Cálculo de Puntajes y Aplicación de Cuotas Legales

Implementar algoritmos que calculen automáticamente los puntajes totales de cada postulante y generen los ordenamientos finales aplicando las cuotas de acción afirmativa establecidas por la legislación uruguaya. El sistema debe:

- Calcular automáticamente el puntaje total de cada postulante sumando las calificaciones obtenidas en méritos, pruebas y entrevistas (cuando aplique)
- Generar ordenamientos automáticos que respeten el orden de mérito (mayor a menor puntaje) como criterio principal
- Aplicar automáticamente las cuotas legales obligatorias:
 - 8 % de posiciones para personas afrodescendientes (Ley 19.122 [7])
 - 1 % de posiciones para personas trans (Ley 19.684 [8], Decreto 104/019 [26])
 - 4 % de posiciones para personas con discapacidad (Ley 18.651 [9])
- Manejar casos especiales donde los postulantes se autodefinen en múltiples categorías, aplicando las reglas de precedencia establecidas en la normativa

- Generar tanto ordenamientos generales como ordenamientos específicos por cada cupo, facilitando la adjudicación de plazas

Este objetivo aborda uno de los aspectos más críticos y propensos a errores del proceso actual: la aplicación manual de cuotas. La automatización garantiza que el cumplimiento normativo sea exacto y auditável, eliminando el riesgo de cálculos erróneos que podrían derivar en conflictos legales.

OE4: Reducir Tiempos Administrativos y Eliminar Duplicación de Tareas

Optimizar el flujo de trabajo del personal de Recursos Humanos eliminando tareas manuales repetitivas que actualmente consumen tiempo significativo. El sistema debe:

- Eliminar la necesidad de digitación manual de datos, ya que los postulantes ingresan directamente su información en el sistema
- Automatizar la consolidación de información que actualmente se realiza mediante hojas de cálculo dispersas
- Reducir el tiempo necesario para organizar y clasificar documentación, mediante almacenamiento estructurado y categorizado automáticamente
- Minimizar el intercambio de emails para confirmaciones y consultas, proporcionando confirmación automática al completar la inscripción
- Facilitar la generación de reportes estadísticos que actualmente requieren recopilación y procesamiento manual de datos

La métrica asociada a este objetivo es la reducción estimada del tiempo total del proceso desde el lanzamiento del llamado hasta la publicación de resultados, que en el proceso actual puede tomar varias semanas, a un plazo significativamente menor con el sistema digitalizado.

OE5: Garantizar Transparencia y Trazabilidad Completa del Proceso

Implementar mecanismos que permitan auditar cada etapa del proceso de selección y proporcionen visibilidad a los postulantes sobre el estado de su participación. El sistema debe:

- Registrar automáticamente todas las acciones significativas realizadas por cada usuario del sistema (inscripciones, evaluaciones, generación de ordenamientos) con timestamp y responsable
- Proporcionar a los postulantes un panel donde puedan consultar en tiempo real el estado actual de su inscripción (recibida, en evaluación, evaluada, finalizada)
- Mantener un historial completo e inmutable de cambios en datos críticos, permitiendo reconstruir el proceso en caso de auditorías o controversias
- Generar códigos únicos de inscripción que sirvan como comprobante inequívoco de participación
- Publicar los ordenamientos finales de forma estructurada y accesible, respetando el anonimato según lo establecido en las bases del llamado

La transparencia y la trazabilidad son fundamentales para fortalecer la legitimidad del proceso de selección y generar confianza tanto en los postulantes como en la ciudadanía en general.

OE6: Garantizar Cumplimiento Normativo Integral

Asegurar que el sistema cumpla rigurosamente con el marco legal y normativo que regula tanto la protección de datos personales como la accesibilidad digital y las políticas de acción afirmativa. El sistema debe:

- **Protección de Datos (Ley 18.331 [3]):**

- Recolectar únicamente los datos estrictamente necesarios para el proceso de selección (principio de minimización)
 - Solicitar datos sensibles (autodefinición étnico-racial, identidad de género, discapacidad) exclusivamente para cumplir con las cuotas legales, con carácter opcional y consentimiento explícito
 - Implementar medidas de seguridad apropiadas para proteger la confidencialidad e integridad de los datos personales almacenados
 - Respetar los derechos de acceso, rectificación, actualización y supresión de datos establecidos en la ley
- **Accesibilidad Digital (Decreto 406/022) [27]:**
- Cumplir con las pautas de accesibilidad WCAG 2.1 nivel AA [28], tal como exige el decreto para sitios web de organismos públicos
 - Garantizar que las personas con discapacidad puedan acceder, navegar y completar el proceso de inscripción de forma autónoma
 - Proporcionar alternativas textuales para contenido no textual, estructura semántica adecuada, contraste de colores suficiente, y compatibilidad con tecnologías asistivas
- **Cuotas de Acción Afirmativa (Leyes 19.122 [7], 19.684 [8], 18.651 [9]):**
- Automatizar la aplicación de las cuotas legales en la generación de ordenamientos finales
 - Documentar los criterios de aplicación de cuotas en las bases del llamado y en la interfaz del sistema
 - Generar reportes que demuestren el cumplimiento de las obligaciones legales

El cumplimiento normativo no es opcional, ya que el incumplimiento de estas leyes podría traer consecuencias legales graves para la institución y afectar la validez jurídica de todo el proceso de selección.

OE7: Facilitar la Participación Equitativa de Todos los Ciudadanos

Diseñar e implementar un sistema que reduzca las barreras de acceso al proceso de selección, garantizando igualdad de oportunidades para todos los ciudadanos elegibles. El sistema debe:

- Permitir la inscripción completamente remota, beneficiando especialmente a personas que residen lejos de las oficinas de la DGC [1] o que tienen dificultades de movilidad
- Proporcionar una interfaz de usuario clara, intuitiva y bien documentada que no requiera conocimientos técnicos avanzados
- Ofrecer horario de disponibilidad 24/7 durante el período de inscripción, eliminando restricciones de horario laboral que afectan a personas con trabajos incompatibles
- Implementar funcionalidades de accesibilidad que permitan a personas con discapacidad visual, motora o cognitiva completar el proceso de forma autónoma
- Proporcionar mecanismos de solicitud de apoyos específicos para personas con discapacidad que los requieran durante las etapas presenciales del proceso (pruebas, entrevistas)

Este objetivo está alineado con los principios de equidad y no discriminación que deben regir los procesos de selección en la administración pública.

OE8: Mejorar la Experiencia de Usuario de Todos los Actores

Diseñar interfaces y flujos de trabajo que prioricen la usabilidad y la satisfacción de los distintos tipos de usuarios del sistema. El sistema debe:

- Para postulantes:
 - Proporcionar un proceso de inscripción guiado paso a paso que reduzca la incertidumbre

- Ofrecer feedback inmediato sobre el estado de la información ingresada (correcta, incorrecta, incompleta)
- Permitir guardar borradores y completar la inscripción en múltiples sesiones si fuera necesario
- Proporcionar mensajes que indiquen exactamente qué debe corregirse
- Para el tribunal evaluador:
 - Organizar la información de forma que facilite la evaluación de múltiples postulantes
 - Proporcionar visualizaciones de los datos clave (puntajes, estados, documentación)
 - Minimizar la cantidad de clics y navegación necesaria para completar tareas comunes
 - Ofrecer capacidades de búsqueda y filtrado simples de usar
- Para personal de RR. HH.:
 - Proporcionar dashboards con métricas relevantes sobre el avance del proceso
 - Facilitar la exportación de datos en formatos estándar para auditorías y análisis
 - Ofrecer herramientas de gestión de llamados que simplifiquen la configuración de nuevos concursos

Una buena experiencia de usuario no es solo deseable desde el punto de vista de satisfacción, ya que tiene impactos medibles en la tasa de completitud de inscripciones, la reducción de errores, y la adopción efectiva del sistema por parte de los usuarios internos.

OE9: Proveer Información Confiable y Auditabile para la Toma de Decisiones

Generar reportes, estadísticas y documentación que faciliten tanto la gestión operativa del proceso como la rendición de cuentas institucional. El sistema debe:

- Generar automáticamente los documentos oficiales requeridos (actas, ordenamientos, constancias)
- Proporcionar reportes estadísticos sobre cantidad de inscripciones, distribución por departamento, distribución por cupos, promedios de puntajes, etc.
- Facilitar la exportación de datos en formatos estructurados (Excel[29], PDF, Comma Separated Values (CSV)) para análisis posteriores.
- Mantener logs de auditoría completos que permitan reconstruir la secuencia de eventos en caso de controversias.
- Generar evidencia documental que respalte las decisiones tomadas en cada etapa del proceso.

La disponibilidad de información confiable y auditável es fundamental tanto para la gestión efectiva del proceso como para la rendición de cuentas y la transparencia institucional.

OE10: Validar la Viabilidad Técnica mediante una prueba de concepto (POC)

Dado el carácter académico y exploratorio de este trabajo integrador, un objetivo específico es desarrollar un Proof of Concept (Producto Mínimo Viable (MVP)) que permita validar la viabilidad técnica de la solución propuesta. Este MVP debe:

- Implementar los dos workflows críticos del sistema (inscripción de postulantes y evaluación por tribunal) de forma funcional.
- Demostrar que la arquitectura tecnológica propuesta (Angular [16] + ASP.NET Core [23] + SQL Server [20]) es apropiada para el dominio del problema.
- Validar que las funcionalidades core (workflow de inscripción, cálculo automático de puntajes, generación de ordenamientos con cuotas) son técnicamente viables.

- Proporcionar una base sólida de código, documentación y decisiones de diseño que facilite la evolución futura hacia un sistema completo listo para producción.
- Identificar mediante pruebas con usuarios reales si el diseño de las interfaces y los flujos de trabajo responde efectivamente a las necesidades identificadas.

Este objetivo reconoce que, en el contexto de un proyecto académico con restricciones temporales, no es realista pretender desarrollar un sistema completamente listo para la producción. En su lugar, el enfoque está en validar los conceptos fundamentales y demostrar que la dirección técnica y de diseño es correcta, proporcionando una base para desarrollos futuros.

2.3 Alineación de Objetivos con la Problemática Identificada

Los objetivos específicos planteados responden directamente a los problemas identificados en el análisis de la situación actual:

La alineación entre problemas y objetivos, como se ilustra en la **Tabla 2.1**, asegura que el desarrollo del sistema esté fundamentado en necesidades reales y que cada componente implementado contribuya a resolver aspectos concretos de la problemática actual.

2.4 Criterios de cumplimiento

Para cada objetivo específico se definieron criterios de éxito que permiten evaluar si el objetivo fue alcanzado¹:

- **OE1 (Digitalizar inscripción):** Tasa de completitud exitosa >90 %, tiempo promedio <15 minutos
- **OE2 (Facilitar evaluación):** Tiempo de evaluación por postulante reducido >50 % vs. proceso manual

¹Se presentan las métricas esperadas en una futura implementación para el proyecto integral de desarrollo final del portal (no del PoC)

- **OE3 (Automatizar cálculos):** 0 errores de cálculo detectados en pruebas de validación
- **OE4 (Reducir tiempos):** Tiempo total del proceso reducido >40 % vs. proceso actual
- **OE5 (Transparencia):** 100 % de acciones críticas registradas en logs de auditoría
- **OE6 (Cumplimiento normativo):** 0 hallazgos críticos en auditoría legal y de accesibilidad
- **OE7 (Participación equitativa):** Incremento >30 % en inscripciones de personas con discapacidad
- **OE8 (Experiencia de usuario):** Satisfacción usuario >4/5, tasa de abandono <10 %
- **OE9 (Información auditabile):** 100 % de reportes generados automáticamente sin intervención manual
- **OE10 (PoC viable):** Workflows críticos 100 % funcionales, cobertura de tests >80 %

Problema Actual	Objetivo que lo Aborda
Procedimiento presencial que excluye a personas con movilidad reducida o residentes alejados	OE1: Digitalizar inscripción / OE7: Facilitar participación equitativa
Demoras de 72 horas en confirmación de recepción	OE1: Confirmación inmediata con código único
Duplicación de tareas (postulante completa formulario, RR. HH. redigita)	OE4: Reducir tiempos administrativos
Errores de digitación manual de datos	OE1: Ingreso directo por postulante / OE4: Eliminar digitación manual
Dificultades de trazabilidad y transparencia	OE5: Garantizar transparencia y trazabilidad
Cálculo manual de puntajes propenso a errores	OE3: Automatizar cálculo de puntajes
Aplicación manual de cuotas legales compleja y susceptible a errores	OE3: Automatizar aplicación de cuotas
Riesgo de incumplimiento de Ley 18.331 [3] (datos personales)	OE6: Garantizar cumplimiento normativo
Incumplimiento de Decreto 406/022 [27] (accesibilidad)	OE6: Accesibilidad WCAG 2.1 AA [28] / OE7: Facilitar participación equitativa
Dispersión de documentación dificulta evaluación	OE2: Centralizar información del postulante
Falta de herramientas sistematizadas para tribunal	OE2: Desarrollar módulo especializado de evaluación
Proceso opaco para postulantes	OE5: Panel de seguimiento / OE8: Mejorar experiencia de usuario

Tabla 2.1: Alineación de objetivos con problemática

Definición del problema

3.1 Problemática General

La gestión de procesos de selección de personal en organismos públicos enfrenta desafíos estructurales que trascienden las particularidades institucionales. A nivel global, la permanencia de procedimientos administrativos basados en mecanismos presenciales y flujos de trabajo manuales genera ineficiencias sistémicas que afectan tanto la capacidad operativa de las instituciones como la experiencia de los ciudadanos que interactúan con ellas.

Los procesos de concursos públicos, en particular, requieren la gestión de grandes volúmenes de información personal sensible, la coordinación de múltiples actores (postulantes, personal administrativo, tribunales evaluadores), el cumplimiento riguroso de marcos normativos complejos, y la garantía de transparencia en cada etapa del procedimiento. Cuando estos procesos se desarrollan mediante metodologías tradicionales como formularios en papel, envío de documentación por correo postal o electrónico sin estructura, consolidación manual de datos, cálculos en hojas de cálculo, aparecen problemas recurrentes que comprometen la eficiencia, equidad y confiabilidad del sistema.

3.2 Manifestaciones del Problema

Las manifestaciones concretas de esta problemática pueden agruparse en cuatro dimensiones principales:

Ineficiencia operativa

La duplicación de tareas constituye quizás el síntoma más evidente. Los postulantes deben completar formularios que posteriormente son redigitados por personal administrativo, generando esfuerzo redundante y aumentando la probabilidad de errores de transcripción. La orga-

nización y consolidación manual de documentación dispersa (archivos en diversos formatos, múltiples correos electrónicos, carpetas físicas) consume tiempo valioso que podría destinarse a actividades de mayor valor. Las verificaciones de requisitos excluyentes, cuando se realizan manualmente sobre decenas o cientos de postulaciones, resultan propensas a inconsistencias.

Falta de transparencia y trazabilidad

Los procesos manuales dificultan el seguimiento en tiempo real del estado de las postulaciones. Los aspirantes carecen de visibilidad sobre si su documentación fue recibida correctamente, si cumple con los requisitos formales, o en qué etapa del proceso se encuentra su candidatura. Esto genera incertidumbre, consultas administrativas adicionales y, en última instancia, baja la confianza ciudadana en las instituciones públicas.

Riesgo de incumplimiento normativo

Los marcos legales contemporáneos imponen obligaciones en materia de protección de datos personales (Ley 18.331 [3]), accesibilidad digital (Decreto 406/022 [27]), y acciones afirmativas [4] para grupos históricamente discriminados (Leyes 19.122 [7], 19.684 [8], 18.651 [9]). La gestión manual dificulta garantizar el cumplimiento sistemático de estas normativas. Por ejemplo, la aplicación de cuotas de acción afirmativa en la generación de listas de prelación, cuando se realiza manualmente mediante hojas de cálculo, introduce riesgos significativos de error que pueden derivar en incumplimientos legales involuntarios.

Barreras de accesibilidad

Los procedimientos presenciales o que requieren manipulación de documentos físicos establecen barreras para personas con discapacidad, residentes en zonas geográficamente alejadas de oficinas administrativas, o ciudadanos con restricciones de movilidad. Si bien el envío de documentación por correo electrónico mitiga parcialmente esta limitación, la ausencia de interfaces diseñadas considerando principios de accesibilidad universal excluye a sectores significativos de la población.

3.3 Relevancia e Impacto

La relevancia de esta problemática radica en sus consecuencias múltiples y de largo alcance. Desde la perspectiva ciudadana, los procesos ineficientes generan frustración, desconfianza en las instituciones públicas y pueden disuadir a candidatos calificados de postularse, empobreciendo la calidad del pool de aspirantes. Desde la perspectiva institucional, los procesos manuales demandan recursos humanos relevantes para tareas administrativas de bajo valor, limitando la capacidad de los organismos para escalar sus operaciones o invertir en actividades estratégicas.

A nivel sistémico, la permanencia de procesos no digitalizados en pleno siglo XXI contrasta con los objetivos de transformación digital del Estado proclamados por gobiernos contemporáneos. La agenda de gobierno electrónico, impulsada en Uruguay por AGESIC [10] y materializada en diversas iniciativas (portal *gub.uy* [5], sistema de firma electrónica - Ley N° 18600 [30], identificación digital [31]), requiere la modernización progresiva de todos los procedimientos administrativos para cumplir sus promesas de eficiencia, transparencia y accesibilidad.

3.4 El Caso de la Dirección General de Casinos

La Dirección General de Casinos constituye un caso de esta problemática general. Como organismo público responsable de la administración de casinos y salas de esparcimiento del Estado. La DGC [1] realiza periódicamente llamados a concursos para incorporar personal en diversos roles operativos y administrativos. El procedimiento vigente reproduce fielmente las limitaciones descritas, formularios descargables en Excel [29] que deben completarse manualmente, enviarse por correo electrónico, ser procesados individualmente por personal de Recursos Humanos, y consolidarse en planillas de cálculo para su evaluación por el tribunal.

El análisis del último llamado público realizado por la DGC (para el cargo de Fiscal III, con 20 plazas distribuidas entre Montevideo y Canelones) [11] revela la magnitud del desafío. Con etapas que incluyen preselección de requisitos excluyentes, ordenamiento aleatorio ante escribano público, pruebas de conocimiento teóricas y prácticas, evaluación de méritos y an-

tecedentes, entrevistas personales, y generación de listas de prelación aplicando tres cuotas de acción afirmativa distintas, se nota una complejidad administrativa considerable. La gestión manual de este proceso no solo es ineficiente, sino que también introduce riesgos significativos de error humano en puntos críticos del workflow.

3.5 Oportunidad de Solución

La unión de diversos factores crea una ventana de oportunidad para abordar esta problemática mediante soluciones tecnológicas. En primer lugar, la madurez alcanzada por tecnologías web modernas (frameworks JavaScript para interfaces de usuario, APIs RESTful [18] para comunicación backend, bases de datos relacionales robustas) permite construir sistemas escalables, seguros y accesibles con inversiones razonables. En segundo lugar, la disponibilidad de patrones de diseño probados en portales gubernamentales existentes (BPS [12], BSE [13], Uruguay Concursa [14]) reduce riesgos de implementación. En tercer lugar, el marco normativo vigente (particularmente el Decreto 406/022 [27] sobre accesibilidad digital) establece obligaciones que funcionan como catalizadores para la modernización tecnológica.

El presente proyecto se inscribe en este contexto, proponiendo el desarrollo de un portal institucional que digitalice integralmente el proceso de llamados públicos de la DGC [1]. Si bien el alcance inmediato se circscribe a un organismo específico, las lecciones derivadas, los patrones de diseño implementados y los componentes desarrollados tienen potencial de replicación en otros organismos públicos que enfrentan desafíos similares. En ese sentido, este trabajo constituye tanto una solución particular como un caso de estudio para la modernización más amplia de procesos administrativos en el Estado uruguayo.

La digitalización completa del workflow de llamados públicos (desde la postulación inicial hasta la generación automatizada de listas finales) no representa meramente una mejora incremental, sino una transformación cualitativa que redefine la relación entre ciudadanos e instituciones públicas, establece nuevos estándares de eficiencia administrativa y fortalece la equidad en el acceso a oportunidades laborales en el Estado.

Antecedentes

Este capítulo revisa los antecedentes relevantes que contextualizan el desarrollo del Portal DGC, abarcando tanto sistemas similares implementados en Uruguay y a nivel internacional, como investigación académica sobre digitalización de procesos de selección de personal, implementación de cuotas de acción afirmativa, y marcos normativos aplicables. El análisis de estos antecedentes proporcionó fundamentos teóricos y prácticos que orientaron las decisiones de diseño del proyecto.

4.1 Portales de Concursos Públicos en Uruguay

Uruguay cuenta con varios sistemas digitales de gestión de concursos públicos desarrollados por distintos organismos del Estado. Durante la fase de investigación de este proyecto se realizó un análisis comparativo de estos portales para identificar buenas prácticas, patrones de diseño exitosos y oportunidades de mejora.

Uruguay Concursa - Oficina Nacional del Servicio Civil (ONSC)

Uruguay Concursa es la plataforma oficial de la Oficina Nacional del Servicio Civil para la gestión de llamados a concursos en toda la administración pública central del Estado uruguayo [14]. Lanzado en 2017, constituye el referente más completo en materia de digitalización de concursos públicos en el país.

Características principales:

- Portal público donde los ciudadanos pueden consultar todos los llamados abiertos de la administración central
- Sistema de postulación en línea mediante formulario web estructurado.
- Validación de requisitos excluyentes mediante declaración jurada del postulante.

- Carga de documentación probatoria en formato digital (PDF).
- Generación automática de código de confirmación de inscripción.
- Seguimiento del estado del llamado por parte del ciudadano.

Fortalezas identificadas:

- Proceso de postulación bien estructurado que guía al postulante paso a paso.
- Confirmación automática de recepción mediante código único vía correo electrónico.
- Interfaz relativamente clara y accesible.
- Centralización de todos los llamados de la administración central en un único portal.

Limitaciones detectadas:

- La aplicación de cuotas de acción afirmativa (afrodescendientes, trans, discapacidad) no está automatizada, requiriendo procesamiento manual posterior.
- El cálculo de puntajes totales y generación de ordenamientos se realiza fuera del sistema, generalmente en hojas de cálculo.
- Errores de carga durante el proceso de postulación.
- La gestión del llamado (evaluación, puntajes y actas) se realiza fuera de la plataforma por cada Unidad Ejecutora. Uruguay Concursa [14] actúa como un portal centralizador de la postulación y no como un sistema integral de gestión de concursos.
- Problemas de accesibilidad detectados: navegación por teclado incompleta en algunos componentes, contraste insuficiente en ciertos elementos, estructura de encabezados inconsistente.

Para una visualización detallada de alguno de los problemas detectados en la plataforma, véase el Apéndice A.

Portal de Concursos del Banco de Previsión Social (BPS)

El BPS [12] implementó un portal web para gestionar sus llamados internos a concursos de promoción y de ingreso [32]. El sistema permite a funcionarios y aspirantes externos postularse digitalmente.

Características destacadas:

- Formulario digital de inscripción con validaciones básicas
- Integración con bases de datos internas del BPS [12] para validar información de funcionarios
- Sistema de carga de documentación en formato PDF
- Notificaciones por email en etapas clave del proceso

Análisis: Si bien el portal del BPS [12] representa un avance respecto a procedimientos completamente manuales, el análisis reveló que varios aspectos del proceso siguen requiriendo intervención manual significativa. La verificación documental, la consolidación de información dispersa y el cálculo de puntajes no están completamente automatizados.

La Figura B.1 en el Apéndice B muestra la evidencia visual del formulario digital utilizado por el BPS [12] para la postulación a concursos.

Portal de Llamados del Banco de Seguros del Estado (BSE)

Similar al BPS [12], el BSE [13] desarrolló un sistema web para sus procesos de selección, orientado tanto a público externo como a promociones internas [33].

Observaciones: El portal del BSE comparte muchas características con el del BPS [12], incluyendo limitaciones similares en automatización de procesos posteriores a la inscripción. Un aspecto positivo es la integración con sistemas de autenticación institucional para funcionarios,

facilitando el acceso. Sin embargo, al igual que otros portales analizados, presenta problemas de accesibilidad que vulneran el cumplimiento del Decreto 406/022 [27], incluyendo errores en las etiquetas de los formularios y bajo contraste en elementos clave (Ver Figura C.1 en el Apéndice C).

Síntesis del Análisis de Portales Uruguayos

El análisis comparativo de estos portales permitió identificar un patrón común, si bien todos han logrado digitalizar exitosamente la etapa de inscripción (eliminando formularios en papel y procedimientos presenciales), las etapas posteriores de evaluación, cálculo de puntajes y generación de ordenamientos siguen dependiendo en gran medida de procesamiento manual fuera del sistema.

Esta observación fue fundamental para definir el alcance del Portal DGC, que se propuso desde el inicio no solo digitalizar la inscripción sino también automatizar completamente el cálculo de puntajes y la aplicación de cuotas legales, aspectos que ninguno de los sistemas analizados resuelve satisfactoriamente. Un análisis comparado de cómo otros países implementan sistemas de cuotas de acción afirmativa se presenta en el Apéndice F. El análisis de soluciones comerciales internacionales (ver Apéndice D) complementa esta perspectiva con casos de implementaciones exitosas en otros contextos gubernamentales.

4.2 Aporte Diferencial del Portal DGC

El análisis de antecedentes revela que, si bien existen precedentes tanto a nivel nacional como internacional de sistemas de reclutamiento digital, el Portal DGC presenta características diferenciales significativas. El marco teórico sobre e-recruitment y digitalización de RR. HH. que fundamenta estas decisiones se desarrolla en profundidad en el Apéndice E.

A nivel nacional:

- Primer sistema uruguayo que automatiza completamente el cálculo de puntajes y aplica-

ción de cuotas legales en ordenamientos finales

- Integración completa desde inscripción hasta generación de actas finales en un único sistema
- Diseñado específicamente para cumplir con particularidades del marco normativo uruguayo de acción afirmativa

A nivel internacional:

- Implementación de algoritmo de ordenamiento que aplica cuotas obligatorias múltiples y superpuestas (afro, trans, discapacidad) de manera completamente automática y auditável.
- Enfoque en accesibilidad digital (WCAG 2.1 AA [28]).
- Sistema de código abierto desarrollado específicamente para sector público, a diferencia de soluciones comerciales propietarias.

A nivel académico:

- Aborda problemática real de organismo público con impacto potencial directo en mejora de gestión de RR. HH..
- Combina múltiples áreas de Ingeniería de Software (desarrollo web full-stack, arquitectura de sistemas, ingeniería de requisitos, testing automatizado, cumplimiento normativo).
- Proporciona evidencia empírica de viabilidad técnica de automatización de cuotas de acción afirmativa en contexto uruguayo.

El contexto académico y metodológico específico de trabajos integradores en Ingeniería de Software se detalla en el Apéndice G.

Los antecedentes analizados proporcionaron fundamentos teóricos y prácticos esenciales para el desarrollo del Portal DGC, permitiendo evitar errores comunes identificados en sistemas

existentes, adoptar mejores prácticas probadas de la industria, y diseñar soluciones innovadoras para aspectos únicos del contexto uruguayo que no han sido resueltos satisfactoriamente por sistemas existentes.

Marco teórico

5.1 Transformación Digital del Estado

Gobierno Electrónico: Conceptos Fundamentales

El gobierno electrónico (e-government) representa un paradigma de administración pública que aprovecha tecnologías de la información y la comunicación (TIC) para transformar las relaciones entre el Estado, los ciudadanos, las empresas y otras entidades gubernamentales. Según las Naciones Unidas, el gobierno electrónico se define como “el empleo de Internet y el World Wide Web para entregar información y servicios gubernamentales a los ciudadanos”[34].

La Organización para la Cooperación y el Desarrollo Económicos (OCDE [35]) distingue cuatro dimensiones principales del gobierno electrónico:

- *e-administration*, que se enfoca en mejorar procesos internos gubernamentales mediante digitalización.
- *e-services*, orientada a proveer servicios públicos electrónicos a ciudadanos y empresas.
- *e-democracy*, que busca fortalecer la participación ciudadana en los procesos democráticos mediante canales digitales.
- *e-governance*, que implica la transformación integral de la gobernanza mediante las TIC.

El proyecto Portal DGC se inscribe primordialmente en las dimensiones de e-administration y e-services, al digitalizar un proceso administrativo interno (gestión de llamados públicos) y, simultáneamente, proveer un servicio electrónico a los ciudadanos (plataforma de postulación en línea).

Modelos de Madurez en Gobierno Electrónico

Los investigadores han desarrollado diversos modelos para caracterizar los niveles de madurez de las iniciativas de gobierno electrónico. El modelo de cuatro etapas propuesto por Layne y Lee [36] constituye uno de los marcos más referenciados:

1. **Catalogación:** Presencia web básica con información estática sobre servicios gubernamentales.
2. **Transacción:** Posibilidad de completar transacciones en línea sin necesidad de interacción presencial.
3. **Integración vertical:** Integración de sistemas entre niveles gubernamentales similares.
4. **Integración horizontal:** Integración de sistemas entre diferentes niveles y funciones gubernamentales.

El Portal DGC aspira a alcanzar el segundo nivel (transacción), permitiendo que el proceso completo de postulación, evaluación y generación de listas de prelación se realice íntegramente en línea, sin requerir interacciones presenciales en ninguna etapa.

Gobierno Electrónico en Uruguay

Uruguay ha demostrado liderazgo regional en el gobierno electrónico. El Índice de Desarrollo de Gobierno Electrónico (EGDI) de la Organización de las Naciones Unidas (ONU) [37] ubica de manera consistente a Uruguay entre los países más avanzados de América Latina, alcanzando en 2020 la posición 26 a nivel global [34]. Esta posición se sustenta en iniciativas como:

- La creación de AGESIC [10] en 2005, como órgano rector de las políticas de gobierno digital.

- El desarrollo del portal *gub.uy* [5] como punto único de acceso a trámites y servicios estatales.
- La implementación de la identidad digital mediante la cédula electrónica.
- El sistema de firma electrónica avanzada para trámites en línea.
- El marco normativo en accesibilidad digital - Decreto 406/022 [27].

El Portal DGC se alinea con estos esfuerzos sistémicos, adoptando dominios institucionales bajo *gub.uy* [5] y siguiendo lineamientos técnicos establecidos por AGESIC [10].

5.2 Arquitecturas de Software para Aplicaciones Web

Arquitectura en Capas

El patrón arquitectónico de capas es uno de los enfoques más consolidados para estructurar aplicaciones empresariales. Fowler[38] describe este patrón como la organización del sistema en grupos de subtareas donde cada grupo se ubica en un nivel particular de abstracción, con cada capa proveyendo servicios a la capa inmediatamente superior y consumiendo servicios de la capa inmediatamente inferior.

La arquitectura de tres capas, adoptada en el Portal DGC, separa las responsabilidades en:

Capa de Presentación (Frontend).

Responsable de la interacción con los usuarios finales, el renderizado de interfaces, la validación de la entrada del lado del cliente y la orquestación de llamadas a servicios de backend. En el Portal DGC, esta capa se implementa mediante Angular 19 [16], un framework JavaScript mantenido por Google que provee características esenciales para aplicaciones empresariales: tipado fuerte mediante TypeScript, sistema de componentes reutilizables, inyección de dependencias

(DI [39]), lazy loading de módulos para optimizar la carga inicial y herramientas integradas para testing.

Capa de Lógica de Negocio (Backend).

Encapsula las reglas de negocio, la orquestación de operaciones, las validaciones del lado del servidor y la coordinación del acceso a datos. Esta capa se implementa mediante ASP.NET Core 8 [23], un framework multiplataforma de Microsoft para la construcción de aplicaciones web modernas. Los servicios implementados (PostulanteService, InscripcionService, LlamadoService, TribunalService, ValidacionService, ArchivoService, ConstanciaService) encapsulan la lógica específica del dominio, garantizando que las reglas del proceso de llamados públicos se apliquen de forma consistente.

Capa de Acceso a Datos (Persistencia).

Responsable de la interacción con el motor de base de datos, abstrayendo operaciones como Create, Read, Update, Delete (CRUD) y gestionando transacciones. Esta capa utiliza Entity Framework Core 8 [24] como Object-Relational Mapper (ORM)[24] , implementando los patrones Repository y Unit of Work [38] para garantizar desacoplamiento entre lógica de negocio y detalles de persistencia.

Esta separación de responsabilidades proporciona beneficios significativos: cada capa puede evolucionar independientemente, siempre que se respeten los contratos de interfaz; los componentes resultan más fáciles de testear en aislamiento; el sistema resulta más mantenable al localizar responsabilidades específicas; y la arquitectura facilita la escalabilidad horizontal mediante el despliegue distribuido de capas.

Arquitecturas RESTful

Representational State Transfer (REST [18]) constituye un estilo arquitectónico para sistemas distribuidos, formalizado por Fielding[40]. Los servicios RESTful [18] se caracterizan por seis restricciones arquitectónicas: arquitectura cliente-servidor, comunicación sin estado, capacidad de caché, interfaz uniforme, sistema en capas y, opcionalmente, código bajo demanda.

El Portal DGC implementa una API RESTful [18] que expone recursos (llamados, inscripciones, postulantes, departamentos) mediante URIs estructuradas, utiliza métodos HTTP estándar [21](GET para consultas, POST para creaciones, PUT para actualizaciones completas, PATCH para actualizaciones parciales, DELETE para eliminaciones), representa recursos mediante JavaScript Object Notation (JSON [41]), y aplica códigos de estado HTTP [21] semánticamente apropiados (200 para éxito, 201 para creación, 400 para errores del cliente, 404 para recursos no encontrados, 500 para errores del servidor).

Esta arquitectura facilita la integración con clientes diversos (aplicaciones web, aplicaciones móviles futuras, integraciones con otros sistemas gubernamentales), aprovecha la infraestructura web estándar (proxies, cachés) y resulta intuitiva para desarrolladores familiarizados con las convenciones HTTP [21].

5.3 Tecnologías y Plataformas de Desarrollo

ASP.NET Core: Framework Backend

ASP.NET Core [23] representa la evolución multiplataforma y de código abierto del framework ASP.NET de Microsoft. Lanzado en 2016, ASP.NET Core [23] fue rediseñado desde cero para ejecutarse en Windows, Linux y macOS, priorizando el rendimiento, la modularidad y la capacidad de ejecutarse en contenedores.

Las características que justifican su selección para el Portal DGC incluyen:

- **Rendimiento:** Benchmarks independientes como TechEmpower[42] ubican consistentemente a ASP.NET Core [23] entre los frameworks web más rápidos, superando alternativas como Node.js, Django, Ruby on Rails o Spring Boot en throughput de solicitudes por segundo.
- **Madurez del ecosistema:** Soporte empresarial de Microsoft[43], extensa documentación oficial, abundancia de bibliotecas de terceros mediante el repositorio de paquetes de .NET (NuGet [44]) y comunidad activa de desarrolladores.
- **Inyección de dependencias nativa:** El framework incorpora un contenedor de Inversion of Control (IoC) que facilita el testing mediante la inyección de dependencias [39].
- **Middleware pipeline:** Arquitectura basada en middleware que permite la composición flexible de comportamientos transversales (autenticación, logging, manejo de errores, Cross-Origin Resource Sharing (CORS)).
- **Integración con Entity Framework Core [24]:** ORM oficial de Microsoft que simplifica las operaciones de persistencia.

Angular: Framework Frontend

Angular [16] es un framework JavaScript de código abierto, mantenido por Google, para la construcción de aplicaciones web de página única (Single Page Application, SPA). La versión 19, adoptada en el Portal DGC, incorpora mejoras significativas en el rendimiento, la experiencia de desarrollo y las capacidades de optimización.

Las características distintivas de Angular [16] incluyen:

- **TypeScript:** Superset de JavaScript que añade tipado estático opcional, lo que permite la detección de errores en tiempo de compilación y una mejor experiencia de desarrollo mediante autocompletado inteligente en IDEs.

- **Sistema de componentes**: Organización de interfaces mediante componentes reutilizables, con encapsulación de la lógica, plantillas y estilos.
- **Inyección de dependencias [39]**: Patrón similar al backend que facilita el testing y la reutilización de servicios.
- **Lazy loading**: Carga diferida de módulos que reduce el tamaño del bundle inicial y mejora el tiempo de carga percibido.
- **Reactive Forms**: API declarativa para la construcción de formularios complejos con validaciones síncronas y asíncronas.
- **RxJS**: Biblioteca de programación reactiva que facilita el manejo de flujos de datos asíncronos mediante observables.

La elección de Angular [16] sobre alternativas como React [17] se fundamenta (además de haber sido utilizada en el curso de Diseño de aplicaciones 2) en su naturaleza, ya que proporciona convenciones claras en lugar de requerir múltiples decisiones arquitectónicas, en su ecosistema integrado (routing, testing, internacionalización incluidos) y en su adopción en contextos empresariales y gubernamentales similares.

Entity Framework Core y Patrones de Acceso a Datos

Entity Framework Core [24] es un ORM moderno que abstrae las operaciones de base de datos mediante objetos .NET. A diferencia de Structured Query Language (SQL) escrito manualmente, Entity Framework Core [24] permite expresar consultas mediante Language Integrated Query (LINQ [45]), proporcionando verificación de tipos en tiempo de compilación y abstracción del dialecto SQL específico del motor de base de datos.

El Portal DGC implementa dos patrones de diseño fundamentales sobre Entity Framework Core [24]:

Patrón Repository.

Abstacta la lógica de acceso a datos detrás de las interfaces, permitiendo que la capa de negocio opere sobre colecciones de entidades sin conocer los detalles de persistencia[46]. El sistema implementa repositorios específicos (PostulanteRepository, InscripcionRepository, LlamandoRepository, etc.) que encapsulan operaciones CRUD y consultas especializadas.

Patrón Unit of Work.

Coordina cambios en múltiples repositorios como una transacción atómica. El UnitOfWork (UoW [38]) del Portal DGC garantiza que operaciones relacionadas (por ejemplo, crear una inscripción, asociar requisitos del postulante, vincular méritos presentados) se ejecuten transaccionalmente: todas se persisten con éxito o todas se revierten en caso de error.

SQL Server: Motor de Base de Datos

SQL Server [20] es un sistema de gestión de bases de datos relacionales desarrollado por Microsoft. La elección de SQL Server [20] sobre alternativas como PostgreSQL [19], MySQL [47] u Oracle [48] se fundamenta en:

- Integración óptima con el ecosistema .NET y Entity Framework Core [24].
- Soporte empresarial consolidado en organismos gubernamentales uruguayos.
- Capacidades avanzadas de respaldo, de recuperación ante desastres y de alta disponibilidad.
- Herramientas de administración robustas (SQL Server Management Studio[49]).
- Cumplimiento con estándares SQL:2016[50] (o ISO/IEC 9075:2016) y características empresariales (transacciones de Atomicidad, Consistencia, Aislamiento y Durabilidad (ACID), aislamiento de transacciones e integridad referencial).

5.4 Accesibilidad Digital

Fundamentos de Accesibilidad Web

La accesibilidad web se refiere a la práctica de diseñar y desarrollar sitios web accesibles para personas con discapacidades. El World Wide Web Consortium (W3C), a través de su iniciativa de Accesibilidad Web (WAI - Web Accessibility Initiative), define la accesibilidad como garantizar que “personas con discapacidades puedan percibir, entender, navegar e interactuar con la Web, y que puedan contribuir a la Web”[\[51\]](#).

Las Web Content Accessibility Guidelines (WCAG) 2.1, publicadas en 2018, constituyen el estándar internacional de referencia [\[28\]](#). Las WCAG se organizan en cuatro principios fundamentales (POUR):

- **Perceptible:** La información y componentes de interfaz deben presentarse de modo que los usuarios puedan percibirlos (alternativas textuales para contenido no textual, contenido adaptable que puede presentarse en diferentes formas sin perder información, contenido distingible que facilita ver y oír).
- **Operable:** Los componentes de interfaz y navegación deben ser operables (funcionalidad disponible mediante teclado, tiempo suficiente para leer y usar el contenido, diseño que no cause convulsiones fotosensitivas, ayudas para la navegación y la localización del contenido).
- **Comprensible:** La información y la operación de la interfaz deben ser comprensibles (texto legible y comprensible, páginas que aparecen y operan de manera predecible, y ayuda para evitar y corregir errores).

Las WCAG [\[28\]](#) definen tres niveles de conformidad: A (mínimo), AA (recomendado para la mayoría de los sitios) y AAA (máximo). El Portal DGC debe alcanzar el nivel AA conforme a lo establecido por el Decreto 406/022 [\[27\]](#).

Marco Normativo de Accesibilidad en Uruguay

El Decreto 406/022, emitido por el Poder Ejecutivo uruguayo el 22 de diciembre de 2022[27], establece obligaciones específicas de accesibilidad digital para los organismos públicos. El decreto dispone que:

“El objetivo de este documento es presentar los requisitos que deben cumplir los servicios y productos digitales para ser considerados accesibles, [...]. Alcance: En servicios y productos digitales quedan comprendidos los sitios web, intranets, servicios web y móviles, aplicaciones web y móviles, trámites, aplicaciones de uso interno y documentos en cualquier formato”

Este marco legal transforma la accesibilidad de la aspiración voluntaria en obligación jurídica, estableciendo plazos graduales de implementación y mecanismos de fiscalización a cargo de AGESIC [10].

Consideraciones de Accesibilidad en el Portal DGC

Durante la fase de investigación, se realizaron entrevistas con un funcionario de RR. HH. con discapacidad visual identificaron requisitos específicos de accesibilidad que se incorporaron como requisitos no funcionales prioritarios:

- Navegación completa mediante teclado sin requerir mouse.
- Compatibilidad con lectores de pantalla (NVDA, JAWS, VoiceOver).
- Contraste de colores conforme a un ratio mínimo de 4.5:1 para texto normal y de 3:1 para texto grande.
- Etiquetas de Accessible Rich Internet Applications (ARIA) en componentes interactivos.
- Indicadores visuales de foco para la navegación por teclado.

- Mensajes de error descriptivos asociados programáticamente a los campos de formulario.

La implementación de estas características no solo cumple obligaciones legales sino que beneficia a todos los usuarios: interfaces navegables por teclado facilitan uso por power users, contraste adecuado mejora legibilidad en condiciones de iluminación adversa, y mensajes de error claros reducen frustración general.

5.5 Marco Normativo y Legal

Protección de Datos Personales

La Ley 18.331 de Protección de Datos Personales y Acción de Habeas Data [3], promulgada en agosto de 2008, regula el tratamiento de datos personales en Uruguay. La ley establece principios fundamentales que deben observarse en sistemas que procesan información personal:

- **Principio de licitud:** Los datos personales solo pueden recolectarse mediante medios lícitos y con el consentimiento del titular.
- **Principio de finalidad:** Los datos deben recolectarse para propósitos determinados, explícitos y legítimos.
- **Principio de proporcionalidad:** Los datos deben ser adecuados, pertinentes y no excesivos respecto al propósito.
- **Principio de calidad:** Los datos deben ser exactos y actualizarse cuando sea necesario.
- **Principio de seguridad:** Deben adoptarse medidas técnicas y organizativas apropiadas para proteger datos contra destrucción, pérdida, alteración, comunicación o acceso no autorizado.

El Portal DGC procesa información personal sensible (datos de identificación, documentación respaldatoria, autodefinición étnico-racial, declaración de discapacidad). El cumplimiento

de la Ley 18.331 [3] requiere implementar medidas técnicas (encriptación de datos en tránsito y reposo, control de acceso basado en roles, auditoría de operaciones sobre datos sensibles) y organizativas (políticas de privacidad transparentes, procedimientos de respuesta ante brechas, capacitación de personal con acceso a datos).

Acciones Afirmativas

El marco normativo uruguayo, bajo la Ley 19.584, establece cuotas de acción afirmativa en la Administración Pública [4] orientadas a compensar discriminaciones históricas:

Ley 19.122 - Cuota afrodescendiente.

Promulgada en 2013, establece que “el ocho por ciento (8 %) de las vacantes generadas en cada organismo público deberán ser cubiertas por personas afrodescendientes”[7].

Ley 19.684 - Cuota trans.

Promulgada en 2018, establece que “al menos el uno por ciento (1 %) de los puestos de trabajo en el Estado [...] será cubierto por personas trans”[8]. El Decreto 104/019 [26] reglamenta los procedimientos de autodefinición conforme a esta normativa.

Ley 18.651 - Cuota de discapacidad.

Promulgada en 2010, establece que “el cuatro por ciento (4 %) del total de vacantes de cada organismo estatal [...] deberá ser cubierto por personas con discapacidad”[9].

La aplicación manual de estas cuotas en procesos de selección constituye una tarea compleja y propensa a errores. El Portal DGC automatiza este proceso mediante algoritmos que generan ordenamientos que respetan rigurosamente los porcentajes legales, reduciendo el riesgo de incumplimiento normativo involuntario.

5.6 Contexto Institucional: La Dirección General de Casinos

La Dirección General de Casinos (DGC)^[1] es una Unidad Ejecutora dependiente del Ministerio de Economía y Finanzas de Uruguay, cuya base legal para la explotación de juegos de azar fue establecida y consolidada mediante las Leyes N° 11.041, N° 11.183 y N° 11.238 de 1948-1949, siendo formalizada por la Ley N° 13.921 de 1970^[52].

La DGC ^[1] tiene como cometido principal la administración, explotación y fiscalización de los juegos de azar en casinos y salas de esparcimiento estatales.

La estructura organizacional comprende establecimientos en 18 de 19 departamentos de Uruguay (salvo Flores). En algunos de ellos, más de un establecimiento, con un total de 34 salas distribuidas a lo largo del territorio nacional, y empleando aproximadamente 800 funcionarios en roles operativos, administrativos y gerenciales.

Los llamados a concursos públicos constituyen el mecanismo legal para incorporar personal. El cargo de Fiscal de Casinos, objeto del último llamado analizado, representa una posición de ingreso con trayectoria en la carrera administrativa. Las tareas del cargo incluyen la fiscalización en sala de juegos, las operaciones de caja, la atención al público y las tareas administrativas en régimen de polifuncionalidad.

5.7 Estado del Arte: Portales Gubernamentales en Uruguay

Uruguay Concursa

Uruguay Concursa ^[14] es la plataforma oficial de la Oficina Nacional del Servicio Civil (ONSC) ^[15] para los llamados a concursos en organismos de la Administración Central. Lanzado en 2018, Uruguay Concursa constituye la referencia más relevante de portal gubernamental de selección de personal en Uruguay.

El análisis de Uruguay Concursa ^[14] revela fortalezas (workflow de postulación estructu-

rado en pasos progresivos, validación temprana de requisitos excluyentes, panel de seguimiento para postulantes) y limitaciones (ausencia de automatización en la aplicación de cuotas de acción afirmativa y de la generación manual de listas de prelación).

Portales de Concursos BPS y BSE

El Banco de Previsión Social (BPS) [12] y Banco de Seguros del Estado (BSE)[13] operan sistemas propios de gestión de llamados para sus respectivas dotaciones. Ambos portales implementan formularios digitales de postulación, pero mantienen significativas porciones del proceso en modalidades semimanuales (verificación documental, consolidación de información, cálculo de puntajes).

El análisis comparativo de estos portales permitió identificar buenas prácticas, como la confirmación automática por correo electrónico, los códigos únicos de identificación para preservar el anonimato y la visualización de requisitos excluyentes, así como oportunidades de innovación, entre ellas la automatización de cuotas, el cálculo de puntajes y las interfaces para la accesibilidad.

5.8 Síntesis: Fundamentación Teórica del Portal DGC

La revisión de literatura y análisis del estado del arte establece fundamentos sólidos para el Portal DGC:

1. El marco conceptual de gobierno electrónico posiciona al proyecto como una iniciativa de e-services que materializa los objetivos de transformación digital del Estado uruguayo.
2. La arquitectura en tres capas y patrones de diseño consolidados (Repository [38], Unit of Work [38], REST [18]) proporcionan una estructura técnica robusta y mantenible .
3. El stack tecnológico seleccionado (ASP.NET Core [23], Angular [16], SQL Server [20], Entity Framework Core [24]) ofrece un balance óptimo entre madurez, rendimiento, soporte empresarial y disponibilidad de desarrolladores capacitados.

4. El marco normativo uruguayo establece obligaciones específicas en materia de accesibilidad digital, protección de datos personales y acciones afirmativas [4] que rigen los requisitos no funcionales del sistema.
5. El análisis de portales gubernamentales existentes proporciona lecciones sobre patrones efectivos y oportunidades de mejora que orientan las decisiones de diseño.

En su conjunto, estos fundamentos teóricos establecen que la digitalización integral de procesos de selección de personal en organismos públicos no solo es técnicamente viable mediante tecnologías contemporáneas, sino que constituye un imperativo para cumplir objetivos de eficiencia administrativa, transparencia institucional, equidad en acceso a oportunidades laborales y cumplimiento de marcos normativos progresivos en materia de inclusión y accesibilidad universal.

Metodología

6.1 Enfoque Metodológico General

El desarrollo del Portal DGC adoptó un enfoque metodológico híbrido que combina elementos de metodologías ágiles con prácticas consolidadas de Ingeniería de Software. Esta estrategia responde a las características específicas del proyecto: alcance bien definido pero sujeto a refinamiento iterativo, equipo de desarrollo reducido, necesidad de validación continua con stakeholders, y restricciones temporales académicas.

El proceso se estructuró en siete fases principales que se ejecutaron secuencialmente con elementos de retroalimentación iterativa:

1. Elicitación y relevamiento de requisitos.
2. Análisis y consolidación de información.
3. Diseño de la arquitectura y del modelo de dominio.
4. Prototipado de interfaces.
5. Desarrollo iterativo del MVP.
6. Testing y validación.
7. Documentación técnica.

Esta estructura permitió avanzar sistemáticamente desde la comprensión del problema hasta la implementación de una solución funcional, manteniendo en todo momento trazabilidad entre requisitos, diseño y código.

Procedimiento Metodológico

7.1 Fase 1: Elicitación y Relevamiento de Requisitos

Elicitación mediante Entrevistas Semiestructuradas

La fase de investigación se extendió durante cuatro semanas y constituyó el fundamento de todas las decisiones subsiguientes del proyecto. Se adoptó la técnica de entrevistas semiestructuradas como método principal de elicitation de requisitos, complementada con el análisis técnico de los sistemas existentes.

Las entrevistas semiestructuradas representan un término medio entre las entrevistas completamente abiertas y los cuestionarios cerrados. Zowghi y Coulin describen esta técnica como especialmente apropiada para proyectos en los que existe conocimiento previo del dominio, pero se requiere profundidad cualitativa para comprender matices y contextos [53].

Diseño del instrumento de entrevista.

Se diseñó una guía de entrevista con tres secciones principales:

- Contexto del entrevistado y su relación con el proceso de llamados públicos.
- Experiencia actual con el proceso vigente (puntos de dolor, frustraciones, ineficiencias).
- Expectativas respecto a un sistema digitalizado (funcionalidades deseadas, preocupaciones, restricciones).

La guía establecía temas obligatorios a cubrir, pero permitía flexibilidad para explorar aspectos emergentes que los entrevistados identificaran como relevantes.

Selección de participantes.

Se implementó muestreo intencional para garantizar representación de todos los actores involucrados en el proceso:

- **Postulantes**_(n=4): Personas que habían participado en llamados anteriores de la DGC [1], seleccionadas para incluir diversidad de edades, niveles educativos y resultados en procesos previos (tanto seleccionados como no seleccionados).
- **Personal de Recursos Humanos**_(n=3): Funcionarios responsables de recibir postulaciones, verificar la documentación, organizar la información y coordinar con el tribunal evaluador.
- **Tribunal evaluador**_(n=2): Miembros con experiencia en la evaluación de méritos, la administración de pruebas de conocimiento y la elaboración de listas de prelación.
- **Usuarios con discapacidad**_(n=1): Funcionario de RR. HH. con limitación visual, cuya perspectiva resultó crítica para identificar los requisitos de accesibilidad.
- **Referentes institucionales**_(n=2): Personal con visión estratégica del proceso y conocimiento de restricciones organizacionales y normativas.

Mapeo de stakeholders.

Previo al diseño del instrumento de entrevista y la selección de participantes, se realizó un análisis sistemático para identificar y clasificar todos los stakeholders del proyecto. Este mapeo utilizó el modelo de círculos concéntricos propuesto por Sharp et al., organizando stakeholders según su nivel de proximidad e influencia sobre el sistema [54]:

- **Nivel Operacional**: Actores que interactúan directamente con el sistema en operación diaria (equipo de desarrollo, operadores de RR. HH., infraestructura de TI).

- **Nivel de Negocio:** Actores que definen necesidades funcionales y establecen criterios de éxito (cliente sponsor DGC, beneficiarios funcionales de RR. HH., consultores internos del área de TI).
- **Nivel de Contexto:** Actores que establecen restricciones regulatorias, proveen recursos o representan intereses estratégicos sin involucramiento directo (reguladores DNSC y jurídica, tutor académico, financiadores MEC-DGC, beneficiarios políticos, clientes potenciales de otras direcciones estatales).

Esta clasificación permitió priorizar los esfuerzos de elicitation, concentrándolos en stakeholders operacionales y de negocio durante las entrevistas, mientras se mantenía informados a los stakeholders sobre el contexto mediante reportes periódicos. La tabla 7.1 detalla la clasificación completa, junto con el rol específico de cada actor identificado.

Tipo	Stakeholder	Rol principal
Contexto	Reguladores: DNSC, Jurídica	Normas y cumplimiento
	Consultor externo: Tutor académico	Asesor técnico-metodológico
	Beneficiario financiero: Presupuesto DGC / MEC	Financiación
	Beneficiario político: DGC / MEC	Apoyo institucional y estratégico
	Clientes potenciales: Otras direcciones del Estado	Replicabilidad del sistema
Negocio	Cliente sponsor: DGC	Impulsor y propietario
	Consultor interno: Área TI	Asesoría técnica interna
	Beneficiario funcional: RR. HH.	Uso operativo del sistema
Operacional	Soporte y mantenimiento: Infraestructura TI	Asegurar disponibilidad
	Operador usuario: Funcionarios RR. HH.	Gestión de llamados
	Equipo de proyecto: Desarrolladores TI	Implementación y mantenimiento

Tabla 7.1: Clasificación de stakeholders del Portal DGC

La Figura 7.1 presenta visualmente este modelo mediante círculos concéntricos, en los que el núcleo operacional (Portal DGC) está rodeado por capas de influencia creciente. Esta representación facilitó la comunicación del alcance del proyecto a diferentes audiencias y sirvió como guía para la estrategia de gestión de stakeholders a lo largo de todo el ciclo de desarrollo.

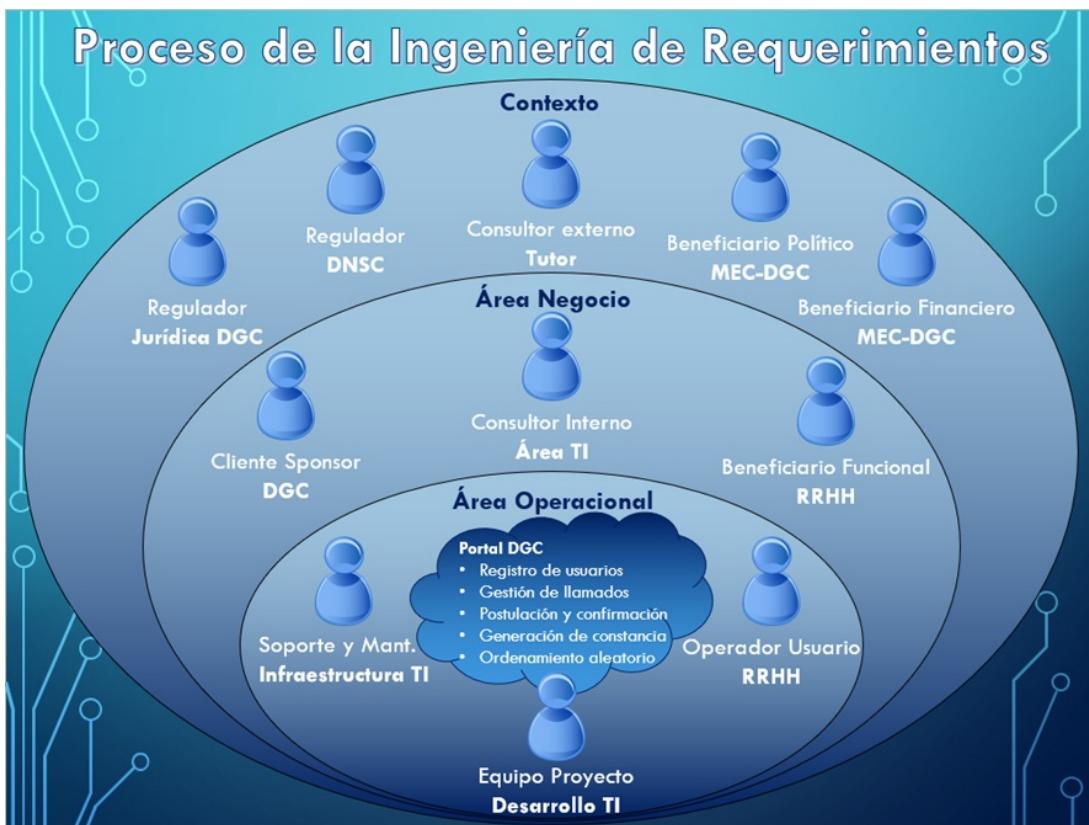


Figura 7.1: Modelo de stakeholders del Portal DGC organizado por nivel de participación

Ejecución de entrevistas.

Las entrevistas se realizaron presencialmente en instalaciones de la DGC [1]. Durante las entrevistas se practicó la escucha activa, reformulando comprensiones para verificar la interpretación correcta y explorando aspectos ambiguos mediante preguntas de seguimiento.

Análisis de datos cualitativos.

El resultado de las entrevistas se analizaron mediante codificación temática, identificando patrones recurrentes en las necesidades expresadas por entrevistados[55]. Este análisis reveló cinco categorías principales de requisitos:

1. **Digitalización completa del flujo:** Todos los grupos de stakeholders enfatizaron la necesidad de eliminar las interacciones presenciales y la manipulación de documentos físicos.
2. **Confirmación y trazabilidad:** Los postulantes demandaban confirmación inmediata de la recepción de la postulación y visibilidad del estado en cada etapa.
3. **Automatización de verificaciones:** RR. HH. requería validación automática de requisitos excluyentes y consolidación de documentación sin digitación manual.
4. **Herramientas de evaluación sistematizadas:** El tribunal necesitaba interfaces que facilitaran el ingreso de calificaciones y calcularan automáticamente los puntajes.
5. **Accesibilidad universal:** Los usuarios con discapacidad identificaron requisitos específicos (navegación por teclado, compatibilidad con lectores de pantalla, contraste adecuado) como condiciones habilitantes para participar.

Ingeniería Inversa de Sistemas Existentes

Paralelamente a las entrevistas, se condujo un análisis técnico de tres portales gubernamentales de concursos públicos en Uruguay: el sistema de concursos del BPS [32], el portal de llamados del BSE [33] y la plataforma Uruguay Concursa [14] de la ONSC [15].

Metodología de análisis.

El análisis se estructuró en tres niveles:

- **Análisis funcional:** Navegación completa de todos los flujos de usuario disponibles públicamente, documentando secuencias de pasos, campos de formulario, validaciones implementadas y mensajes de retroalimentación.
- **Análisis técnico:** Inspección de código fuente mediante herramientas de desarrollador del navegador (Chrome DevTools), identificación de frameworks frontend utilizados, análisis de estructura de peticiones HTTP [21] a APIs [18] backend.

- **Análisis de experiencia de usuario:** Evaluación de usabilidad aplicando algunas de las heurísticas de Nielsen^[56] (Visibilidad del estado del sistema, Correspondencia entre el sistema y el mundo real, Prevención de errores, etc), identificación de puntos de fricción, y evaluación básica de conformidad con WCAG 2.1 [28].

Aspectos relevantes

El análisis comparativo reveló patrones tecnológicos recurrentes y oportunidades de mejora:

Fortalezas identificadas:

- Uso de formularios progresivos, tipo workflow, que dividen los procesos.
- Validación del lado del cliente para proporcionar retroalimentación inmediata.
- Generación de códigos únicos de identificación para preservar anonimato.
- Confirmación automática por correo electrónico.

Debilidades identificadas:

- Ausencia de automatización en aplicación de cuotas de acción afirmativa.
- Cálculo manual de puntajes totales mediante hojas de cálculo externas al sistema.
- Problemas de accesibilidad: navegación incompleta por teclado, contraste insuficiente, falta de etiquetas ARIA [57] en componentes interactivos.
- Validación insuficiente del lado del servidor, confiando excesivamente en validaciones del cliente que usuarios pueden eludir.

7.2 Fase 2: Documentación de Requisitos

Especificación de Requisitos de Software (ESRE)

Con base en los hallazgos de la fase de investigación, se elaboró el ESRE [N] siguiendo el estándar IEEE 830-1998 [22]. El documento ESRE constituye el contrato implícito entre stakeholders y equipo de desarrollo, estableciendo qué debe hacer el sistema (alcance) y qué no debe hacer.

Estructura del ESRE.

El documento se organizó en tres secciones principales:

1. **Introducción:** Contexto del problema, propósito del documento, alcance del sistema.
2. **Descripción general:** Perspectiva del producto, funciones principales, características de usuarios, restricciones, supuestos y dependencias.
3. **Requisitos específicos:** Detalle de 21 RF y 16 RNF.

Requerimientos Funcionales.

Los RF describen comportamientos observables del sistema. Se documentaron usando plantilla que incluye: identificador único, descripción detallada, entradas, procesamiento, salidas, precondiciones, postcondiciones, y prioridad (crítica, alta, media, baja).

Ejemplos de RF:

- **RF-01:** Autenticación de usuarios mediante credenciales (cédula/email + contraseña).
- **RF-04:** Workflow de inscripción en seis pasos con validación progresiva.

- **RF-13:** Calificación de méritos por parte del tribunal con cálculo automático de puntaje total.
- **RF-15:** Generación automática de ordenamientos aplicando cuotas de acción afirmativa (8 % afro, 1 % trans, 4 % discapacidad).

Requerimientos No Funcionales.

Los RNF especifican atributos de calidad del sistema. Se categorizaron en nueve dimensiones siguiendo el modelo de calidad ISO/IEC 25010[58] : rendimiento, usabilidad, portabilidad, escalabilidad, seguridad, fiabilidad, mantenibilidad, compatibilidad y legalidad.

Cada RNF incluye: atributo de calidad, descripción precisa, método de validación, y métrica cuantitativa esperada. Por ejemplo:

- **RNF-UB-02:** El sistema debe cumplir WCAG 2.1 nivel AA [28]. *Validación:* Auditoría con herramienta WAVE. *Métrica:* 100 % de páginas sin errores críticos de accesibilidad.

Casos de Uso

Complementando la especificación de requisitos, se documentaron 20 casos de uso que describen interacciones específicas entre actores (Postulante, Tribunal, RR. HH.) y el sistema. Los casos de uso se documentaron usando plantilla de Cockburn[59] que incluye: actor primario, stakeholders e intereses, precondiciones, garantías mínimas, garantías de éxito, flujo básico, y extensiones (flujos alternativos).

Ejemplos de casos de uso incluyen:

- **CU-04:** Inscribirse a llamado público.
- **CU-13:** Evaluar méritos de postulante.
- **CU-16:** Generar ordenamiento con aplicación de cuotas.

Diagramas de flujo de casos de uso

Los casos de uso se documentaron mediante diagramas de actividad BPMN que muestran la interacción entre actores y sistema. Las figuras 7.2, 7.3, 7.4 7.5 presentan los flujos principales del módulo de autenticación y postulación.

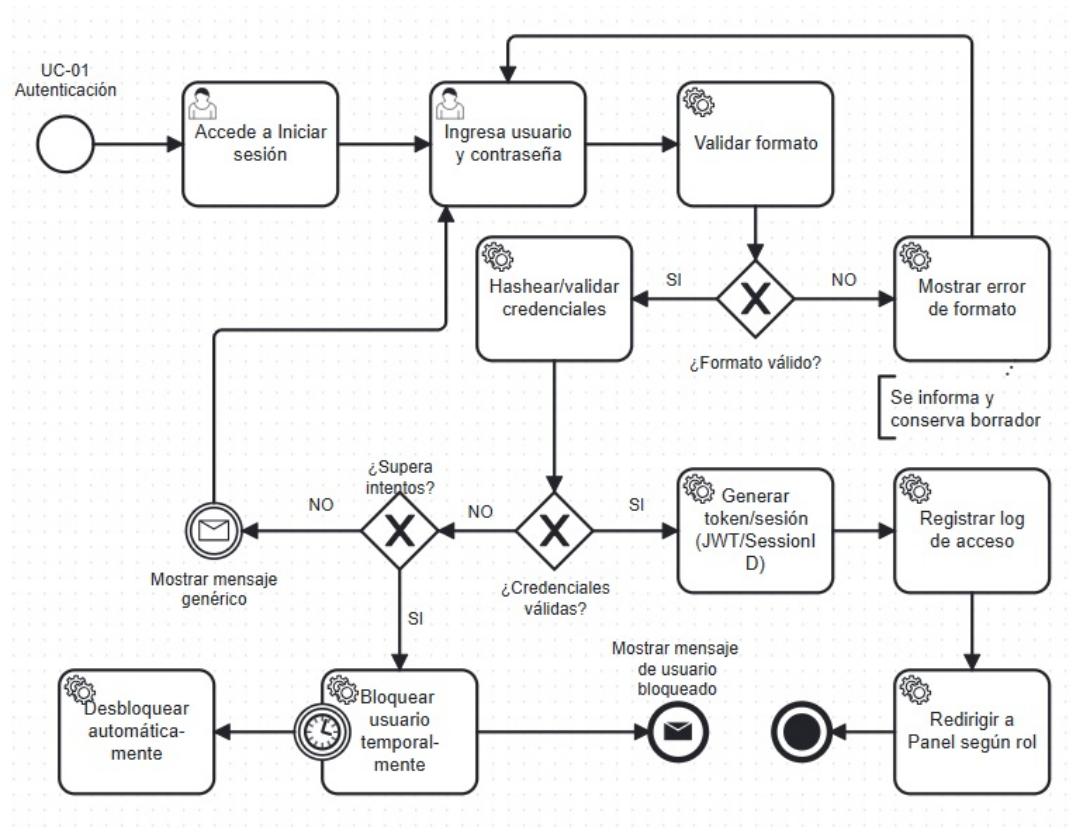


Figura 7.2: UC-01: Flujo de autenticación de usuarios

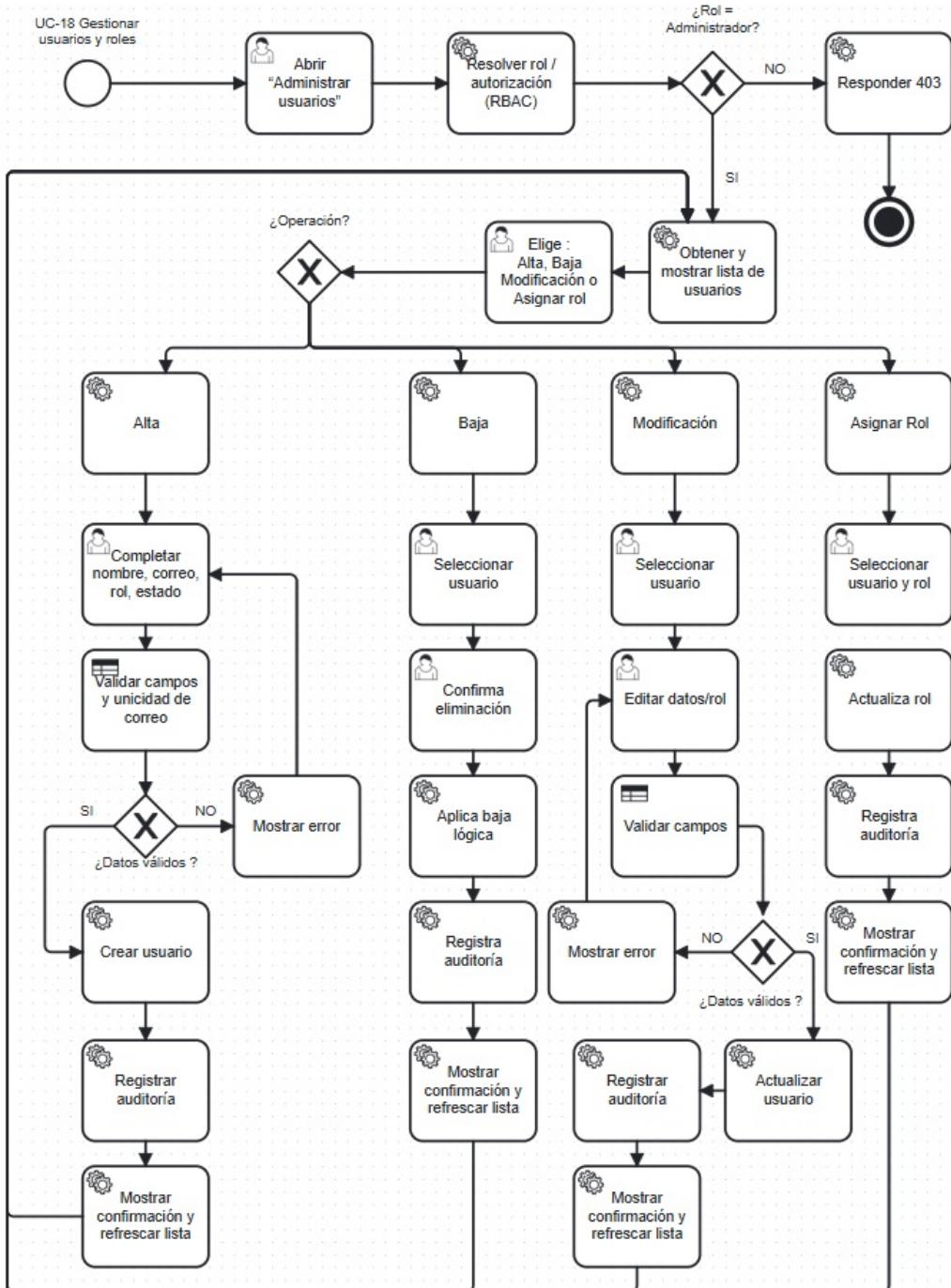


Figura 7.3: UC-18: Flujo de gestión de usuarios y roles

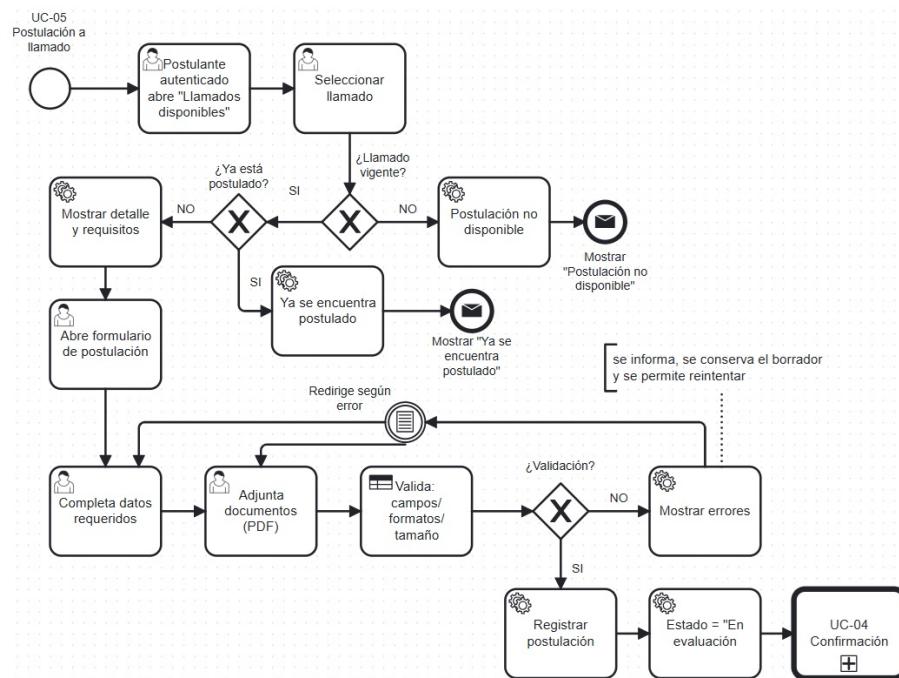


Figura 7.4: UC-05: Flujo de postulación a llamado público

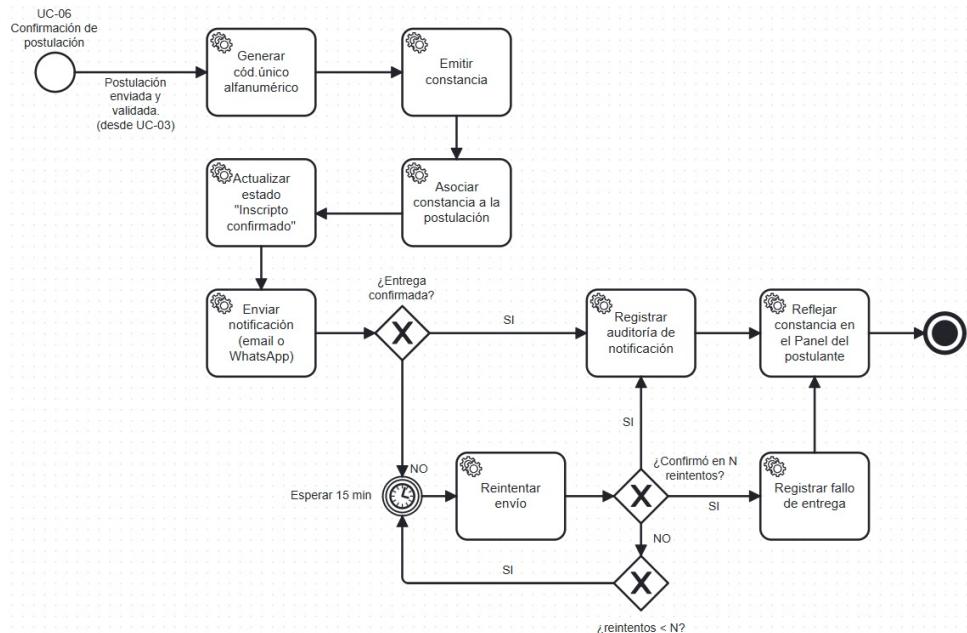


Figura 7.5: UC-06: Flujo de confirmación de postulación

7.3 Fase 3: Diseño de Arquitectura y Modelo de Dominio

Decisiones Arquitectónicas

Con requisitos documentados, se procedió al diseño de la arquitectura del sistema. Este proceso involucró decisiones en tres niveles: arquitectura lógica (organización en capas), arquitectura física (topología de despliegue), y selección de stack tecnológico.

Arquitectura lógica: Patrón de tres capas.

Se adoptó arquitectura en tres capas por sus beneficios en separación de responsabilidades, testabilidad y mantenibilidad. Las capas definidas fueron:

- **Capa de Presentación (Frontend):** Implementada en Angular 19 [16], responsable de renderizado de interfaces, validación del cliente, y llamadas a APIs backend.
- **Capa de Lógica de Negocio (Backend):** Implementada en ASP.NET Core 8 [23], encapsula reglas del dominio (validaciones, cálculo de puntajes, aplicación de cuotas), orquesta operaciones sobre datos, y expone APIs RESTful [18].
- **Capa de Acceso a Datos (Persistencia):** Implementada mediante Entity Framework Core 8 [24] con patrones Repository y Unit of Work [38], abstrae operaciones sobre SQL Server [20].

La Figura 7.6 ilustra la organización en tres capas con las tecnologías utilizadas en cada nivel.

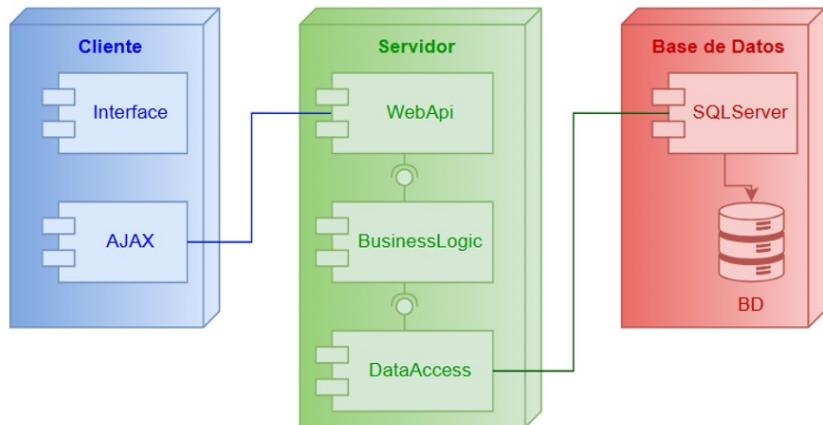


Figura 7.6: Diagrama de implementación - Arquitectura en tres capas con tecnologías asociadas

La comunicación entre capas se establece mediante interfaces bien definidas: el frontend consume APIs REST [18] expuestas por el backend; el backend interactúa con la capa de datos mediante interfaces de repositorios; la capa de datos ejecuta operaciones SQL contra el motor de base de datos.

La Figura 7.7 detalla la organización jerárquica de las capas mostrando las dependencias entre los diferentes paquetes del sistema, destacando la separación clara entre WebApi, BusinessLogic, DataAccess y Domain.

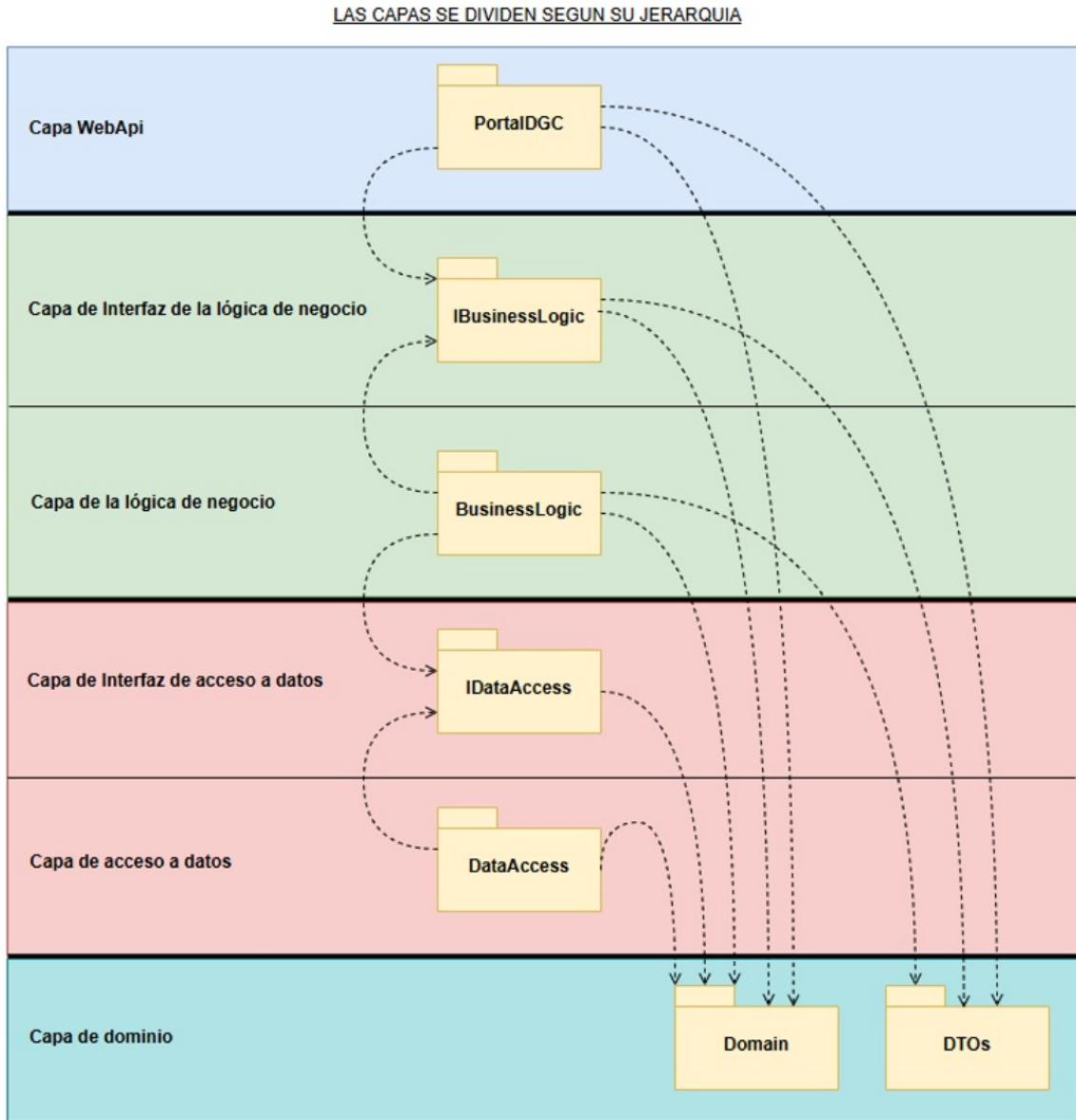


Figura 7.7: Jerarquía de capas y dependencias entre paquetes

Selección del stack tecnológico.

La elección de tecnologías se fundamentó en criterios de madurez, rendimiento, y alineación con estándares gubernamentales:

- **Backend:** ASP.NET Core 8 [23] por su rendimiento superior en benchmarks independientemente de la complejidad del sistema.

dientes, soporte empresarial de Microsoft, y ecosistema maduro.

- **Frontend:** Angular 19 [16] por su naturaleza opinionada (reduce decisiones arquitectónicas), tipado fuerte mediante TypeScript, y adopción en contextos gubernamentales similares.
- **Base de datos:** SQL Server [20] por integración óptima con .NET, soporte consolidado en organismos públicos uruguayos, y capacidades empresariales de respaldo y alta disponibilidad.
- **ORM:** Entity Framework Core 8 [24] por abstracción de SQL, soporte de migraciones de esquema, y lazy loading de relaciones.

Arquitectura física: Topología de despliegue.

Se diseñó arquitectura de despliegue con segmentación de red mediante VLANs (Virtual Local Area Networks) para cumplir requisitos de seguridad¹. La topología incluye cuatro VLANs:

1. **VLAN DMZ:** Expone frontend Angular [16] al público internet mediante servidor web.
2. **VLAN Aplicación:** Aloja API backend ASP.NET Core [23], accesible solo desde DMZ.
3. **VLAN Datos:** Contiene SQL Server [20], accesible solo desde VLAN Aplicación.
4. **VLAN Administración:** Permite acceso de administradores para mantenimiento, aislada de las demás.

Firewalls configurados entre VLANs limitan comunicación al mínimo necesario, implementando defensa en profundidad. La Figura 7.8 presenta el diagrama de despliegue completo mostrando la segmentación de red, los componentes desplegados en cada VLAN, y las reglas de firewall que controlan el tráfico entre zonas.

¹Dicho diseño no se impementa en el POC

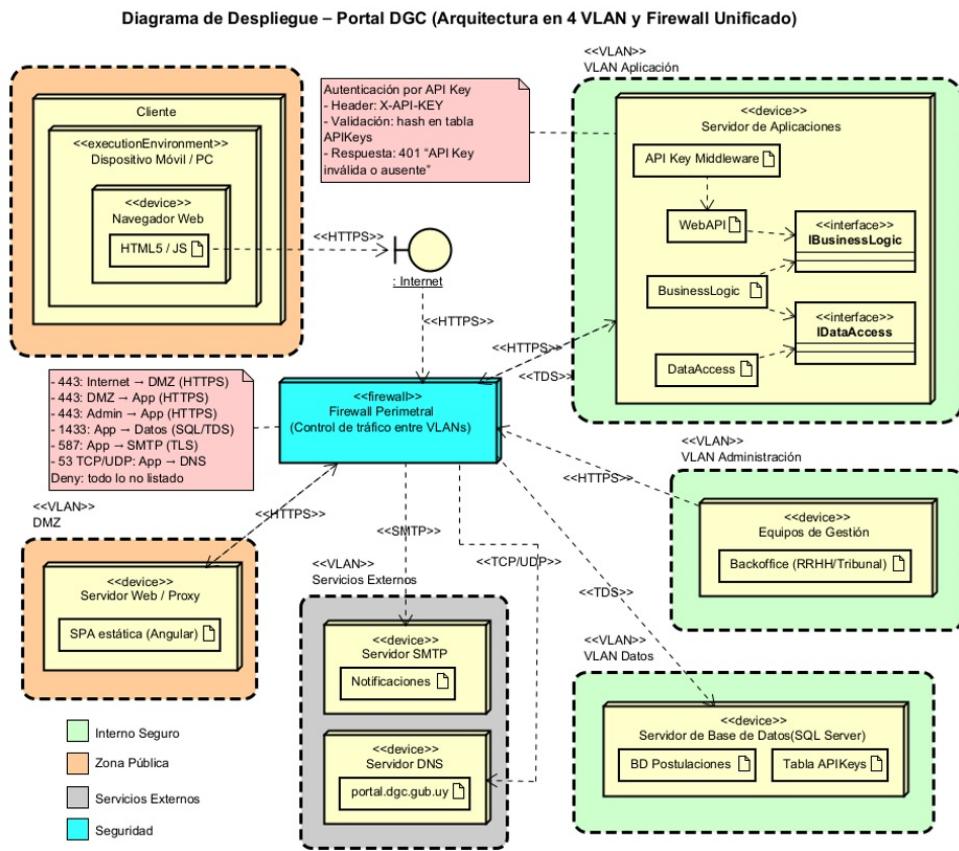


Figura 7.8: Diagrama de despliegue con arquitectura de 4 VLANs y firewall unificado

Esta arquitectura proporciona múltiples capas de seguridad: incluso si un atacante compromete el servidor web en la DMZ, no puede acceder directamente a la base de datos debido a la segmentación de red y las políticas restrictivas del firewall.

Mecanismos de autenticación y autorización.

El sistema implementará autenticación basada en JWT [41] y autorización mediante roles ². El flujo de autenticación se diseñó siguiendo el estándar RFC 7519 para garantizar interoperabilidad y seguridad[60].

La Figura 7.9 presenta el diagrama de secuencia completo del proceso de autenticación

²Dicho mecanismo no se impementa en el POC y se da como supuesto

y autorización, mostrando las interacciones entre el cliente, el controlador de autenticación (`AuthenticationController`), el filtro de autorización (`AuthorizationFilter`), y el servicio de sesión (`ISessionService`).

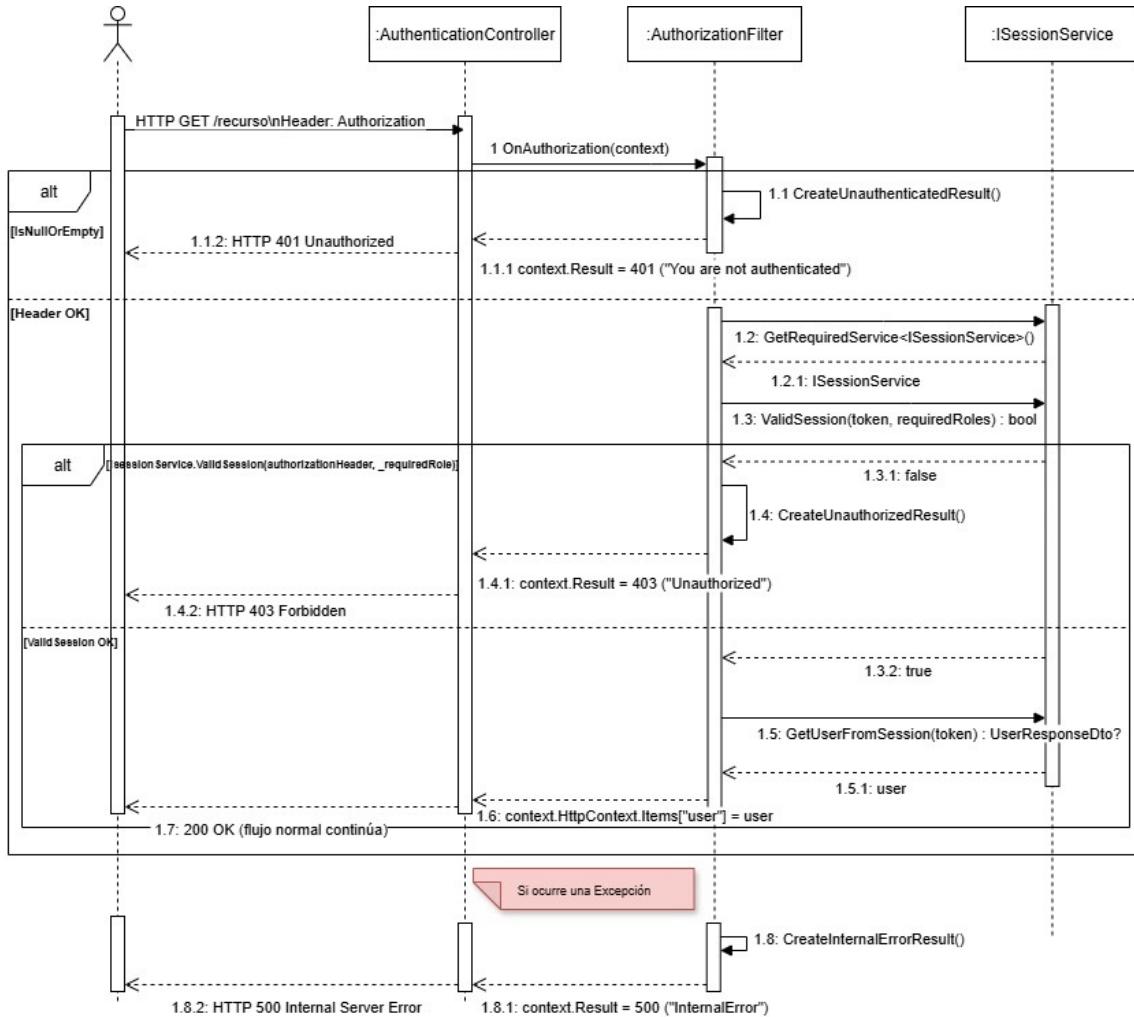


Figura 7.9: Diagrama de secuencia del flujo de autenticación y autorización

El flujo muestra:

1. El usuario envía credenciales mediante petición HTTP [21] con encabezado `Authorization`.
2. Si el token no existe o es inválido, el sistema retorna HTTP 401 (Unauthorized) o 403 (Forbidden) según corresponda.

3. Si la sesión es válida, el filtro extrae el usuario del token y permite la continuación de la petición.
4. Cualquier excepción interna se captura y retorna HTTP 500 (Internal Server Error).

Esta arquitectura de seguridad centralizada permite auditar todos los accesos al sistema y facilita la implementación de políticas de autorización basadas en roles.

Modelado del Dominio

El modelo de dominio representa conceptos centrales del proceso de llamados públicos y sus relaciones. Se diseñó modelo con 21 entidades agrupadas en “Aggregates” cohesivos[61]:

Aggregate Llamado

Incluye entidades: Llamado, LlamadoDepartamento, RequisitoExcluyente, ItemPuntuable, ApoyoNecesario. El Llamado actúa como raíz del agregado; todas las operaciones sobre entidades relacionadas se coordinan a través de él.

Aggregate Inscripción.

Incluye: Inscripcion, RequisitoPostulante, MeritoPostulante, ApoyoSolicitado, Constancia. Representa una postulación completa con toda la información proporcionada por el aspirante.

Aggregate Evaluación.

Incluye: EvaluacionMerito, EvaluacionPrueba, Prueba. Captura calificaciones asignadas por el tribunal.

Aggregate Ordenamiento.

Incluye: `Ordenamiento`, `PosicionOrdenamiento`. Representa lista de prelación final aplicando cuotas de acción afirmativa.

Las relaciones entre agregados se modelaron para respetar límites transaccionales: una transacción modifica una única raíz de agregado; las referencias entre Aggregates se realizan mediante identificadores, no mediante navegación de objetos.

La Figura 7.10 presenta el diagrama completo del modelo de dominio con las entidades identificadas, sus atributos principales, y las relaciones entre ellas. Este modelo fue generado mediante ingeniería inversa desde el código fuente, garantizando sincronización entre diseño y implementación.

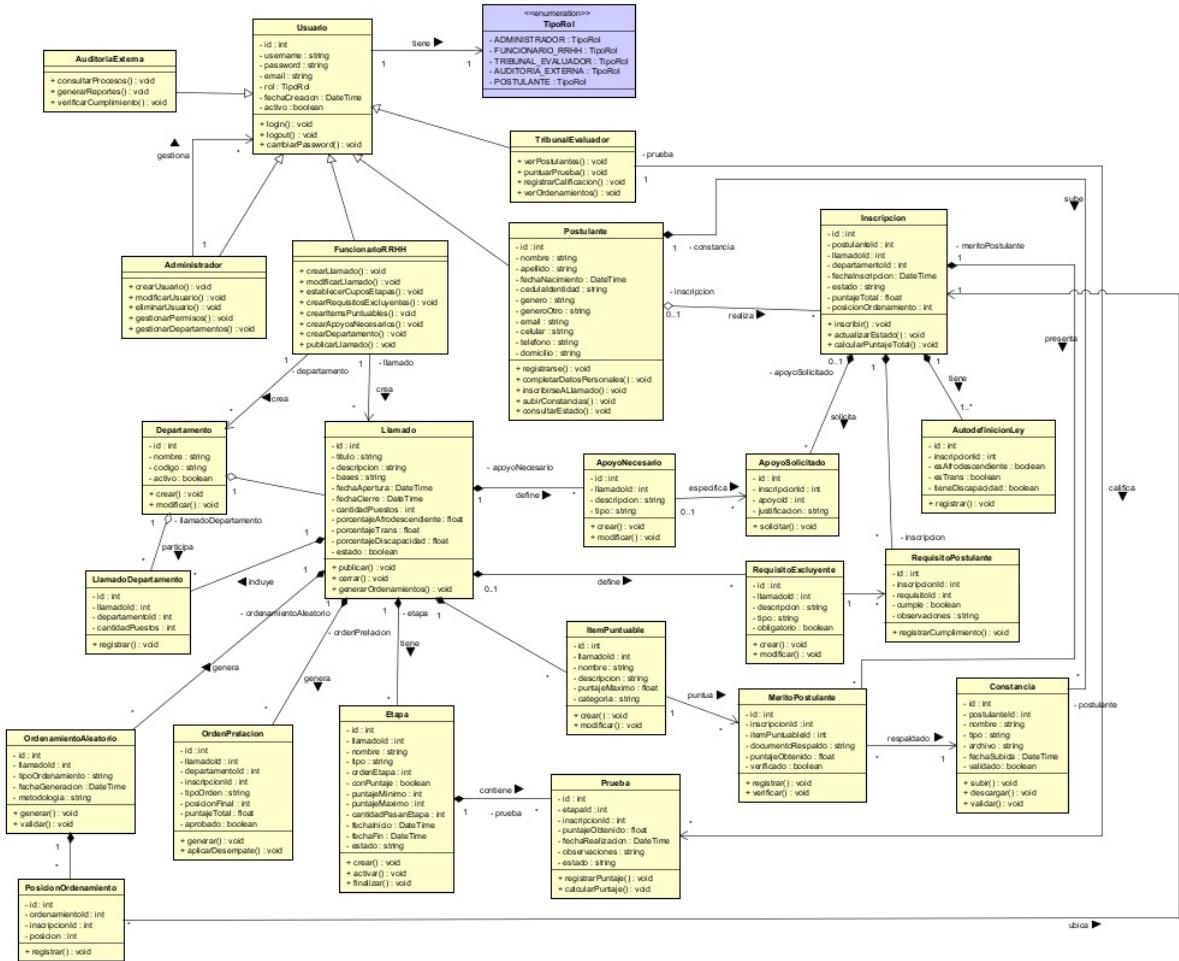


Figura 7.10: Modelo de dominio completo del Portal DGC con entidades y relaciones

El modelo captura tanto las entidades del núcleo del negocio (Llamado, Inscripcion, Postulante) como las entidades de soporte para funcionalidades transversales (Constancia, Tribunal, AutodefinicionLey).

La cardinalidad de las relaciones refleja las reglas del dominio: un llamado puede tener múltiples inscripciones, una inscripción pertenece a exactamente un postulante, y un postulante puede tener múltiples inscripciones a diferentes llamados.

Flujos de Proceso por Rol

El sistema implementa tres flujos principales diferenciados según el rol del usuario: postulante, RR. HH. y tribunal evaluador.

Flujo del Postulante

La Figura 7.11 presenta el flujo completo desde la consulta de llamados hasta el seguimiento del estado de postulación.

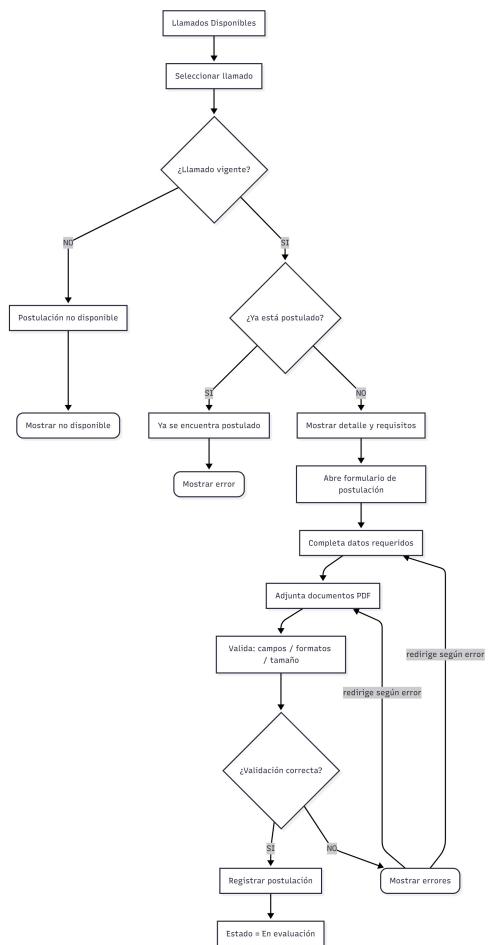


Figura 7.11: Flujo completo de postulación a llamado

Flujo de RR. HH.

El módulo de Recursos Humanos gestiona la creación de llamados, configuración de requisitos y coordinación con el tribunal³. La Figura 7.12 ilustra este proceso.

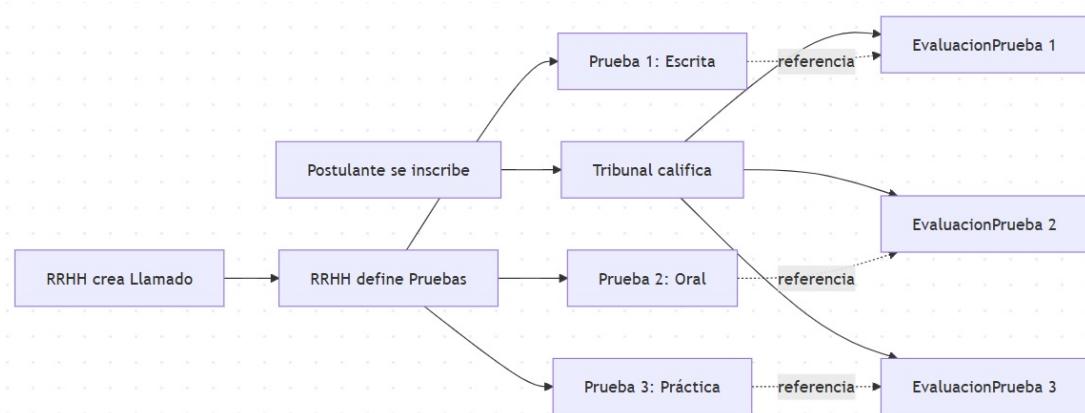


Figura 7.12: Flujo del módulo RR. HH.

Flujo del Tribunal Evaluador

El tribunal accede a funcionalidades especializadas de evaluación. La Figura 7.13 muestra el dashboard y opciones disponibles.

La Figura 7.14 detalla el proceso completo de evaluación desde la revisión de postulaciones hasta la generación de ordenamientos.

³Esta funcionalidad no está implementada en el POC

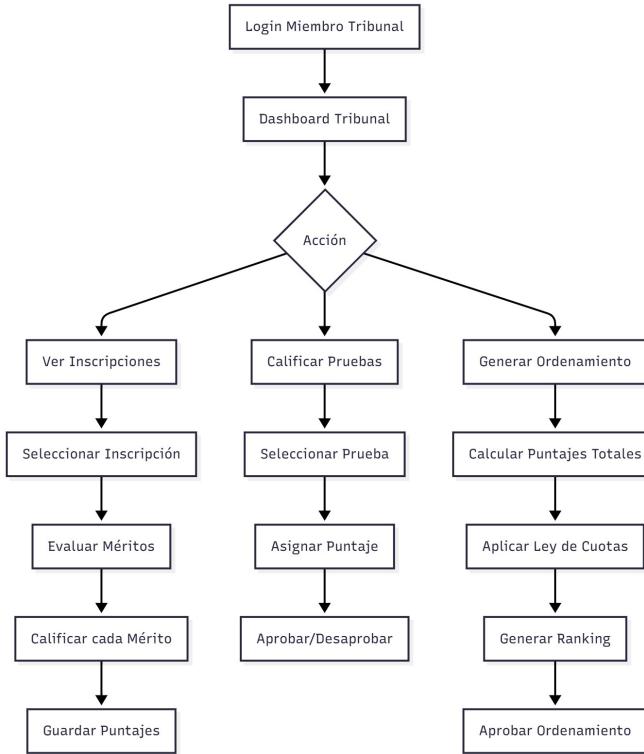


Figura 7.13: Dashboard del tribunal evaluador

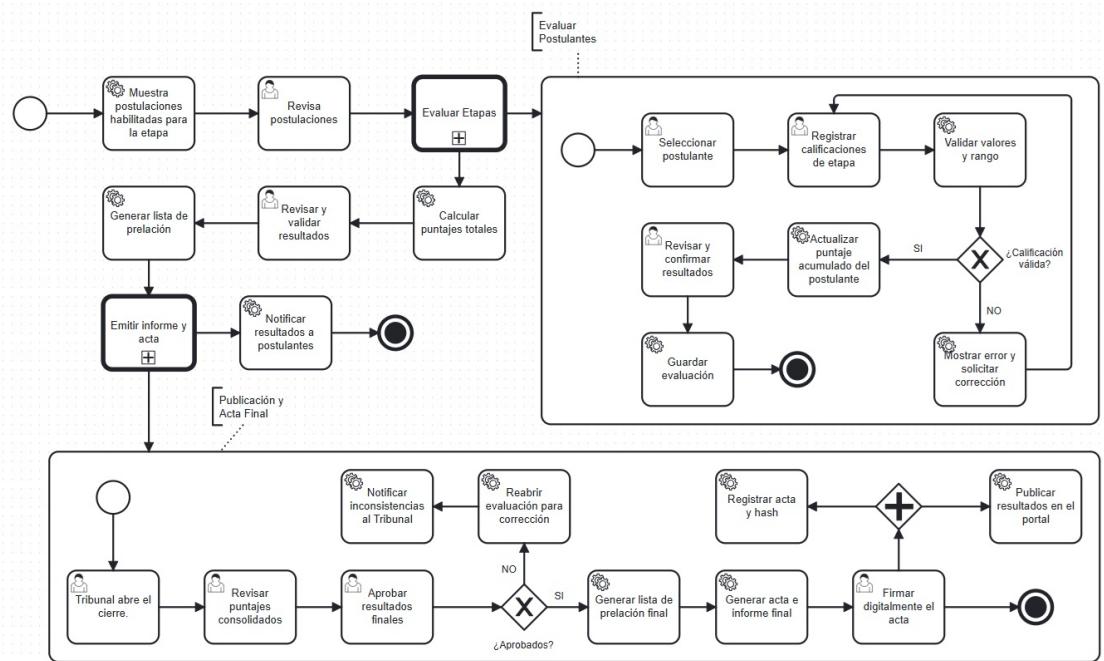


Figura 7.14: Flujo completo de evaluación del tribunal

A continuación, en la Figura 7.15, se muestra el proceso de calificación específico.

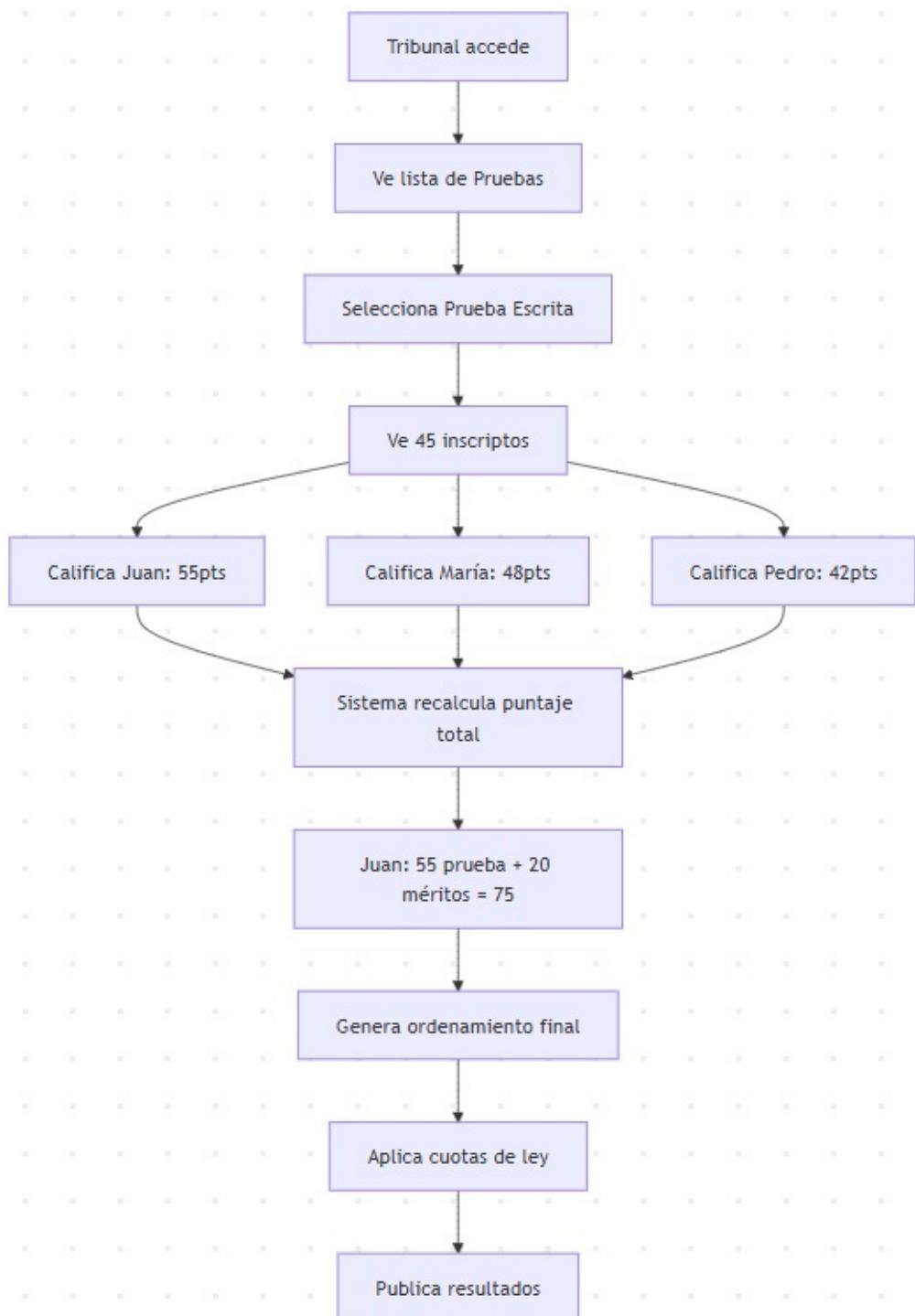


Figura 7.15: Flujo de calificación y asignación de puntajes

Flujo de Evaluación Integrado

La Figura 7.16 presenta la intersección de los tres roles principales durante el proceso de evaluación y generación de listas de prelación.

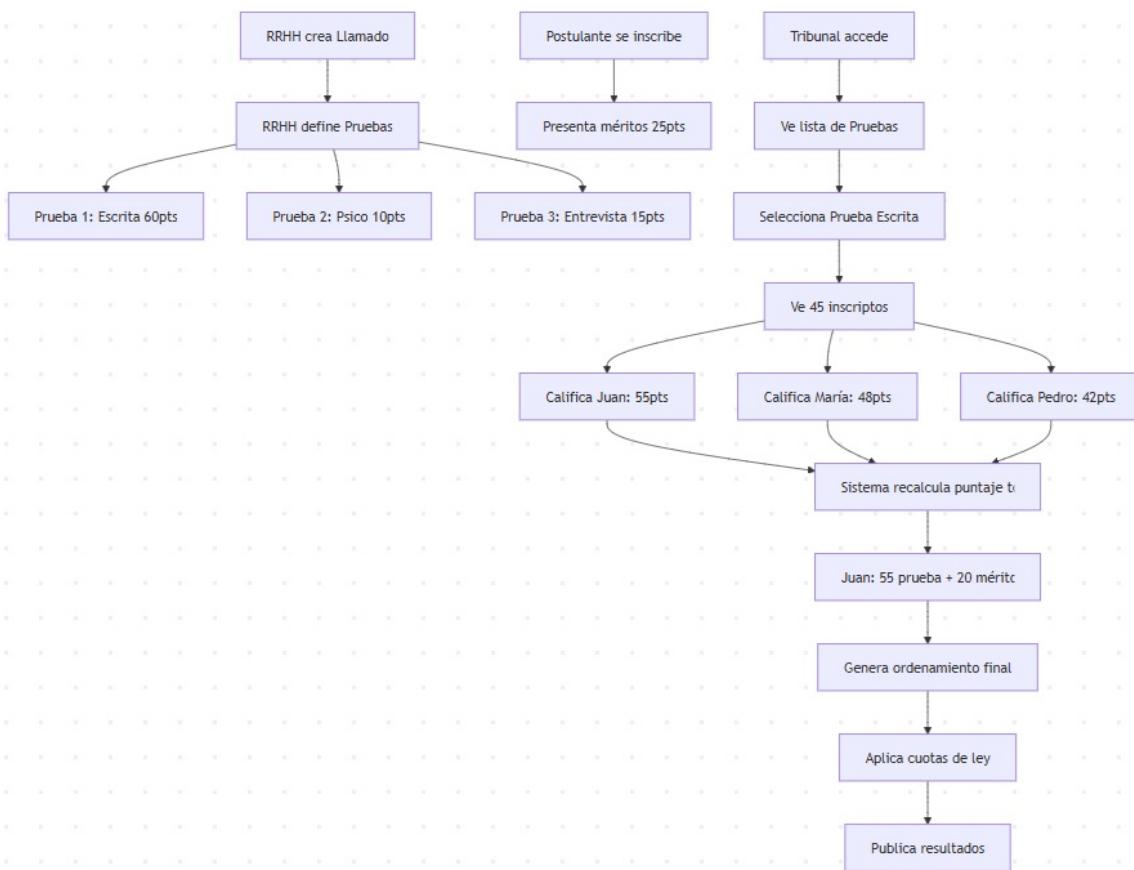


Figura 7.16: Flujo integrado de evaluación con participación de RR. HH., Tribunal y Postulante

Algoritmo de Ordenamiento con Cuotas

El algoritmo de generación de ordenamiento aplicando cuotas de acción afirmativa constituye una funcionalidad crítica del sistema. La Figura 7.17 ilustra el proceso algorítmico.

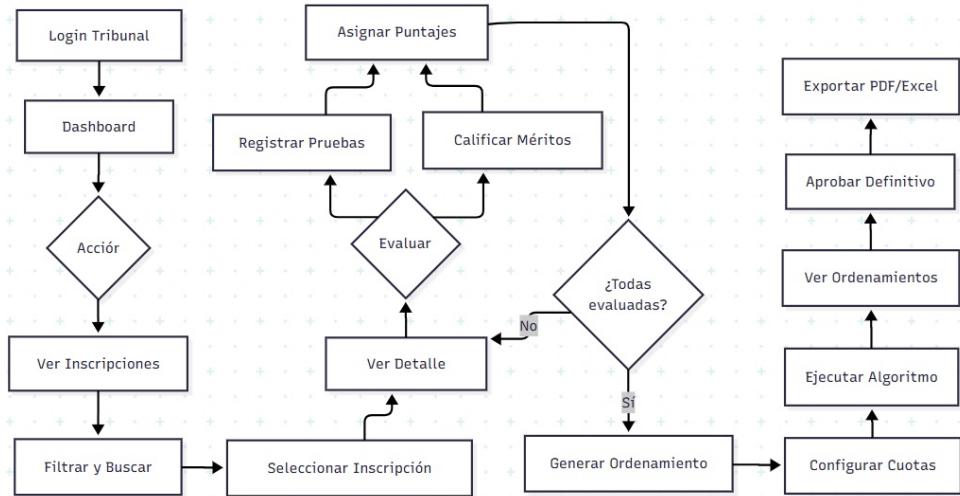


Figura 7.17: Flujo del algoritmo de ordenamiento con aplicación de cuotas legales

7.4 Fase 4: Prototipado de Interfaces

Previo al desarrollo de código productivo, se creó prototipo interactivo de alta fidelidad utilizando Framer [62], herramienta especializada en diseño de interfaces y prototipado. El propósito fue validar flujos de interacción y diseño visual con usuarios representativos antes de invertir esfuerzo en implementación. El proceso siguió metodología de diseño iterativo con ciclos de prototipado-validación-refinamiento.

“User interface design must be iterative: it is practically impossible to get it right the first time.” — Nielsen, *Usability Engineering*, 1993[63].

Alcance del prototipo

Se prototiparon los dos flujos principales:

1. Inscripción del postulante, desde consulta de llamados hasta confirmación de postulación (sin recepción por correo electrónico), utilizando la herramienta Framer [62].

- Evaluación del tribunal, desde dashboard inicial hasta generación de ordenamiento. En este caso se utilizaron Mockups para muestra y validación.

Proceso de Diseño Iterativo: Dashboard del Tribunal

El dashboard del tribunal atravesó tres iteraciones de diseño basadas en feedback de usuarios y refinamiento progresivo de elementos visuales y funcionales.

Iteración 1: Wireframe de baja fidelidad.

La primera iteración consistió en wireframe de baja fidelidad enfocado en estructura de información y disposición de componentes. La Figura 7.18 muestra esta versión inicial con elementos representados mediante cajas grises sin estilización.

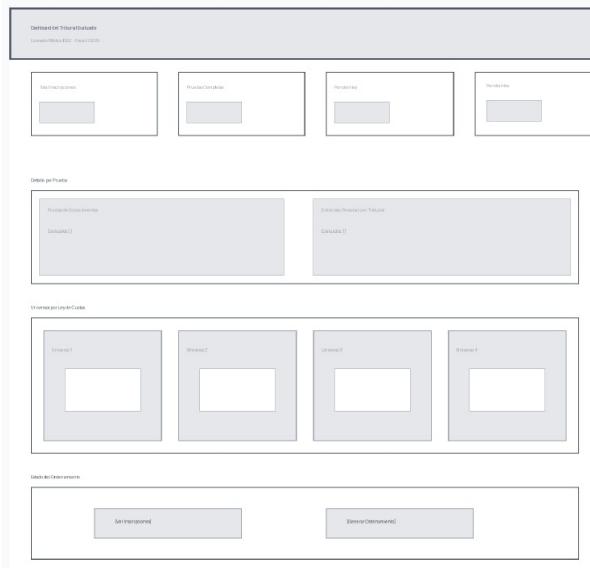


Figura 7.18: Iteración 1: Wireframe de baja fidelidad del dashboard del tribunal

Esta versión confirma mostrar la información mediante cuatro componentes principales:

- Tarjetas de resumen estadístico (tasas de inscripciones, pruebas completadas, aprobados,

promedio general)

- Panel de flujo de pruebas (estadísticas de evaluación del tribunal)
- Sección de universos por ley de cuotas (distribución por universos afro, trans, discapacidad, general)
- Panel de estado del departamento con acciones disponibles

Iteración 2: Refinamiento visual y paleta de colores.

La segunda iteración incorporó paleta de colores institucionales, tipografía definitiva, y refinamiento de espaciado entre componentes. La Figura 7.19 muestra la evolución hacia mockup de media fidelidad.

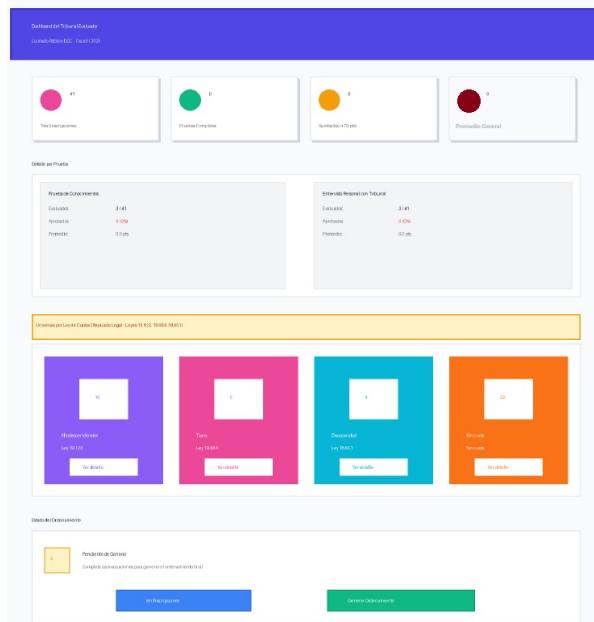


Figura 7.19: Iteración 2: Mockup de media fidelidad con paleta de colores institucional

Cambios principales respecto a iteración 1:

- Aplicación de color institucional en encabezado.

- Diferenciación cromática de tarjetas estadísticas (rosa, verde, naranja, morado).
- Uso de iconografía para reforzar significado de métricas.
- Tarjetas con sistema de colores para universos de ley de cuotas (turquesa, salmón, celeste, naranja).

Iteración 3: Prototipo de alta fidelidad final.

La tercera iteración alcanzó nivel de alta fidelidad con elementos interactivos funcionales, microinteracciones, y datos realistas. La Figura 7.20 presenta la versión final validada con usuarios.

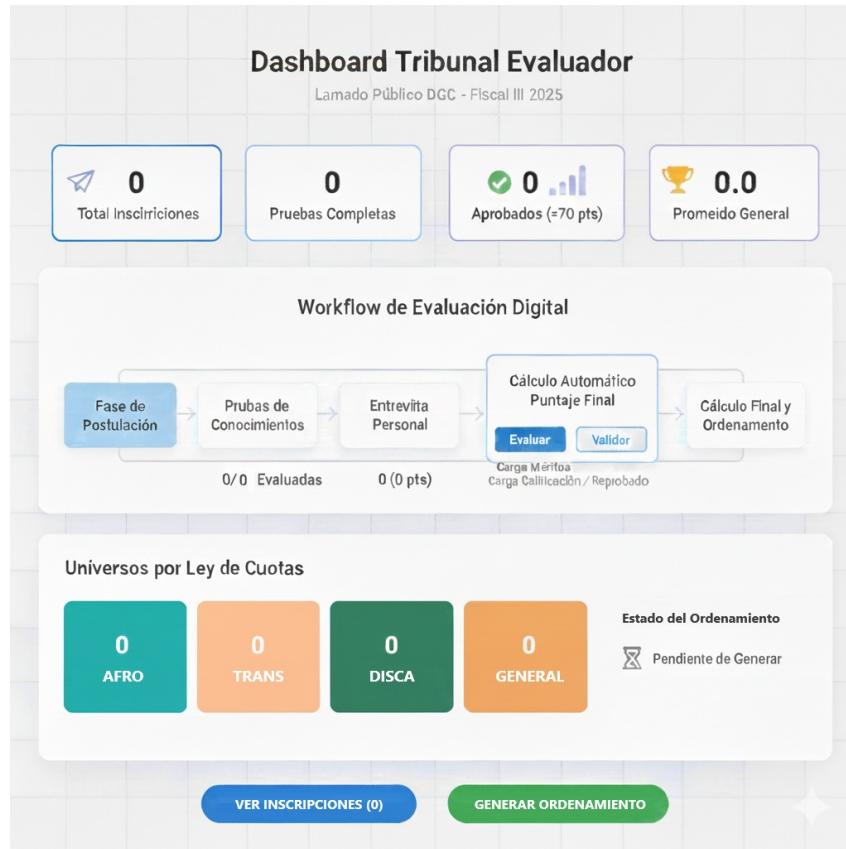


Figura 7.20: Iteración 3: Prototipo de alta fidelidad final del dashboard del tribunal

Refinamientos finales:

- Visualización de workflow de evaluación digital con fases claramente demarcadas
- Indicador de progreso (X/X Evaluadas, X pts) con estado de carga de méritos/calificación.
- Botones de acción con estados (Evaluar activo, Validar deshabilitado)
- Contador de inscripciones por universo (X AFRO, X TRANS, X DISCA, X GENERAL)
- Estado del ordenamiento con icono visual ("Pendiente de Generar")
- Botones de acción principales ("VER INSCRIPCIONES", "GENERAR ORDENAMIENTO")
- Contexto del llamado ("Llamado Público DGC - Fiscal III 2025")

Prototipo del Flujo de Postulante

El flujo de postulante se diseñó como proceso guiado (workflow) de múltiples pasos que acompaña al usuario desde el ingreso al sistema hasta la confirmación de postulación. A diferencia del dashboard del tribunal, este flujo se validó en una única iteración de alta fidelidad. Un video explicativo del prototipo está en:

<https://www.loom.com/share/8ad68c8c6b2943998a13bd780472d5a7>

Pantalla principal y acceso.

La Figura 7.21 muestra la página principal del Portal DGC que presenta información institucional y provee acceso a las funcionalidades del sistema.

Portal de la Dirección General de Casinos

Institucional Dirección General de Casinos

The screenshot shows the main institutional page of the DGC. At the top left is the MEF logo and at the top right is the Casinos del Estado logo. The main title "Portal de la Dirección General de Casinos" is centered above the "Institucional" section. The "Institucional" section contains two columns: "Información Institucional" on the left and "Trasparencia Activa" and "Normativa aplicada" on the right. The "Información Institucional" column includes links to the Mision, vision, objetivos y organigrama, and the Ministerio de Economía y Finanzas logo. The "Trasparencia Activa" column includes links to Responsables de Transparencia Activa y Pasiva, Formulario de Solicitud de Acceso a la Información Pública, Nómina, Respuetas a las Solicituds de Acceso a la Información Pública, Seguridad de la Información, Adquisiciones, Llamados públicos, Presupuestos, Balances y Estados Contables, Protección de Datos Personales, and Remuneraciones. The "Normativa aplicada" column includes a link to Normativa relacionada a la Dirección General de Casinos. At the bottom right of the page is a link to report errors: "¿Hay algún error en el sitio? Enviar reporte". The footer contains links to Unidades Ejecutoras (with a list of 10 units), a contact form for newsletter subscription, and social media icons. The footer also includes links to the official website of Uruguay, map, accessibility statement, terms of use, and privacy policy.

Figura 7.21: Página principal del Portal DGC con información institucional

Esta pantalla incluye:

- Branding institucional (logos del MEF [2] y Casinos del Estado [1])
- Navegación principal (Llamados, Mis Postulaciones, Notificaciones, Mi Perfil, Salir)
- Sección de información institucional con enlaces a transparencia activa y normativa
- Footer con enlaces a unidades ejecutoras y contactos

La Figura 7.22 presenta la pantalla de autenticación que constituye el punto de entrada al sistema para usuarios registrados.



Figura 7.22: Pantalla de autenticación con formulario de login

Workflow de inscripción multipaso.

El proceso de inscripción se implementó mediante Workflow de 6 pasos que guía progresivamente al postulante. Las figuras 7.23 y 7.24 muestran el primer paso: captura de datos personales.

This is a screenshot of a web-based application for job applications. At the top, there is a navigation bar with links: 'Llamados', 'Mis Postulaciones', 'Notificaciones', 'Mi Perfil', and 'Salir'. Below the navigation, the title 'Postulararme a un llamado' is displayed. A sub-instruction states: 'Este servicio permite a las personas postularse a los llamados o concursos abiertos disponibles' and 'Llamado a concurso público y abierto de oposición y méritos para desempeñar tareas de fiscal de Casinos y Salas de esparcimiento.' The main form is titled 'Datos Personales'. It contains three text input fields: 'Nombres' (Jane), 'Apellidos' (Smith), and 'Cédula de identidad' (1234567). Below these, there is a section titled 'ADJUNTAR CÉDULA:' with a button 'Seleccionar archivo' and a note '(Un único archivo, tamaño máximo 5MB)'. At the bottom, there is a 'Género' section with radio buttons for 'Masculino' (selected), 'Femenino', and 'Otro', followed by a text input field '¿Cuál?'.

Figura 7.23: Paso 1A: Datos personales (nombres, apellidos, cédula, adjuntar cédula, género)

Email
janesmith@gmail.com

Teléfono
099123456

Celular
099123456

Domicilio
18 de julio 1234

REQUISITOS EXCLUYENTES PARA EL CARGO AL QUE POSTULA
Marque los requisitos que cumple

Ciudadano uruguayo (Art. 76 de la Constitución)
 Edad, 18 años de edad o más
 Bachillerato completo

Siguiente

Figura 7.24: Paso 1B: Datos de contacto y requisitos excluyentes

El paso 1 captura información básica del postulante y verifica cumplimiento de requisitos excluyentes mediante checkboxes explícitos que obligan al usuario a confirmar cada condición (ciudadanía uruguaya, edad mínima, bachillerato completo). La adjunción de cédula escaneada se implementa mediante botón de carga de archivo con validación de formato y tamaño.

La Figura 7.25 presenta pasos intermedios del workflow donde el postulante declara auto-definición según leyes de acción afirmativa y solicita apoyos necesarios para accesibilidad.

Postularme a un llamado

Este servicio permite a las personas postularse a los llamados a concursos abiertos disponibles
Llamado a concurso público y abierto de oposición y méritos para desempeñar tareas de fiscal de Casinos y Salas de esparcimiento.

Amparo de leyes especiales

Por favor, marque las opciones que correspondan.

- No me apoyo a ninguna ley
- Ley 19864 - Personas Trans
- Ley 19122 - Personas Afrodescendientes
- Ley 18651 - Personas con discapacidad
(En caso de seleccionar esta opción, completar el formulario de accesibilidad)

Apoyos necesarios para las pruebas

A efectos de tomar las providencias que hagan accesible el desarrollo de las pruebas, se requiere información relacionada a los elementos de apoyo necesarios para que el concursante pueda desempeñarse con autonomía e independencia.

Indique los apoyos que necesita:

- | | |
|--|--|
| <input type="radio"/> Micrófono para recibir las instrucciones | <input type="radio"/> Intérprete de lengua de señas |
| <input type="radio"/> Magnificador de pantalla | <input type="radio"/> Lector de pantalla (programa informático) |
| <input type="radio"/> Mouse lado izquierdo o derecho | <input type="radio"/> Adecuación de altura de mesa para sillas de ruedas |
| <input type="radio"/> Adaptación de entrevistas escritas u orales | <input type="radio"/> Acompañante terapéutico |
| <input type="radio"/> Textos con letra ampliada a: <input type="text" value="Tamaño ..."/> | |
| <input type="radio"/> Flexibilidad para la comprensión lectora, tiempo para procesar consignas verbales escritas, léxico accesible, etc. | |

Figura 7.25: Pasos 4-5: Autodefinición según leyes especiales y solicitud de apoyos necesarios

Esta sección implementa requisitos legales de las leyes 19864 [8] (personas trans), 19.122 [7] (afrodescendientes), y 18651 [9] (personas con discapacidad). El diseño utiliza checkboxes con etiquetas descriptivas y mensajes contextuales explicando implicaciones de cada selección. La sección de apoyos presenta opciones mediante radio buttons e inputs de texto libre permitiendo especificar necesidades particulares (micrófono, intérprete de lengua de señas, magnificador de pantalla, lector de pantalla, mouse adaptado, acompañante terapéutico, textos ampliados, etc.).

La Figura 7.26 muestra el paso de declaración jurada y adjunción de constancias documentales.

Declaración Jurada

Por favor, marque las opciones que correspondan.
Declaro bajo juramento que **NO** me encuentro comprendido en las siguientes situaciones:

- No mantengo vínculos vigentes con la Administración Pública, salvo en los casos en que la acumulación de cargos está expresamente permitida por ley.
- No percibo pasividad, retiro ni subsidio proveniente de actividad pública generada por mí, salvo en los casos en que su percepción esté legalmente habilitada o se encuentre suspendida.
- No he sido destituido de ningún cargo público por falta administrativa grave ni por incumplimiento de obligaciones, bajo ninguna forma de vinculación con el Estado.
- No estoy inhabilitado para el ejercicio de funciones públicas por sentencia penal ejecutoriada.
- No me he acogido a los regímenes de retiro incentivado establecidos por las Leyes N°17.556 (art. 10), N°17.672 (art. 1), N°17.930 (art. 29) ni N°18.172 (art. 9).

Adjuntar documentación

La documentación cargada debe ser en formato PDF

ADJUNTAR CONSTANCIA:

Seleccionar archivo

Ningún archivo se...

(Solo debe adjuntar constancia en caso que se ampare a la Ley 19864 - Persona Trans. EN este caso es obligatoria la carga de la constancia)

ADJUNTAR CONSTANCIA:

Seleccionar archivo

Ningún archivo se...

(Solo debe adjuntar constancia en caso que se ampare a la Ley 18651 - Persona con Discapacidad En este caso es obligatoria la carga de la constancia)

Departamento al cual postularse

Select...

Enviar postulación

Figura 7.26: Paso 6: Declaración jurada y adjunción de constancias en formato PDF

Este paso requiere que el postulante acepte declaraciones juradas mediante checkboxes (no inhabilitación para funciones públicas, no acogimiento a regímenes de retiro incentivado). La carga de constancias se implementa mediante botones de selección de archivo con indicación clara del formato requerido (PDF) y restricción explícita de casos de uso.

Finalmente, la Figura 7.27 presenta la pantalla de confirmación de postulación exitosa.

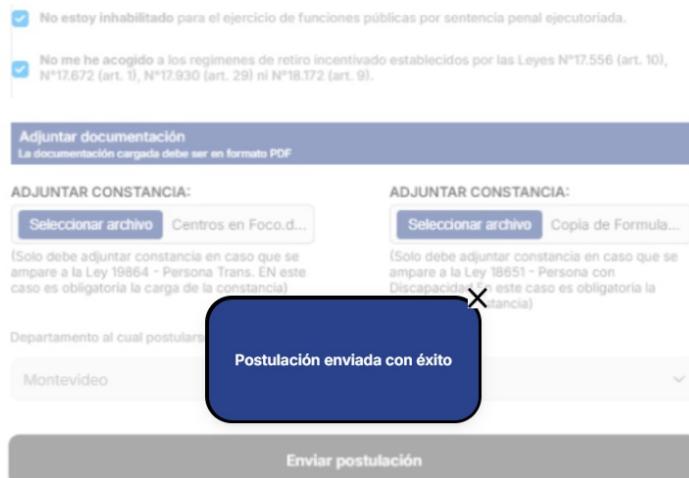


Figura 7.27: Confirmación de postulación enviada con éxito

El mensaje de confirmación utiliza modal overlay con fondo translúcido que enfoca la atención del usuario. El diseño del mensaje de confirmación indica ("Postulación enviada con éxito") mostrando un cierre correcto.

Pruebas con usuarios

El prototipo se validó con usuarios representativos (ex postulantes, funcionarios de RR. HH. y tribunal) mediante sesiones de testing de usabilidad. Se aplicó protocolo de pensamiento en voz alta [63], donde usuarios dicen sus pensamientos mientras interactúan con el prototipo.

Refinamientos derivados

Las sesiones de testing identificaron ajustes necesarios:

- Incrementar contraste de colores en botones primarios para cumplir ratio WCAG 4.5:1 [28].
- Añadir indicador visual de progreso en Workflow de inscripción.
- Reorganizar disposición de información en vista de detalle de postulante para priorizar

datos más relevantes.

- Simplificar formulario de calificación de méritos, consolidando campos relacionados.

Estos ajustes se incorporaron al desarrollo. El proceso iterativo validó decisiones de diseño y ayudó a que la implementación partiera de fundamentos sólidos validadolos con usuarios reales.

7.5 Fase 5: Desarrollo Iterativo

Metodología Ágil con Sprints Semanales

El desarrollo adoptó metodología ágil con sprints de una semana de duración. Esta decisión respondió a restricciones temporales académicas (necesidad de avances semanales para seguimiento docente) y al tamaño reducido del equipo (desarrollo individual, sin overhead de coordinación de equipos grandes).

Cada sprint seguía esta estructura:

1. **Planificación** (miércoles): Selección de tareas del backlog para el sprint, basada en prioridad y dependencias (tutoreado).
2. **Desarrollo** (jueves-domingo): Implementación de funcionalidades, documentación, análisis y búsqueda.
3. **Revisión** (lunes): Demostración de funcionalidades con recopilación de feedback.
4. **Reflexión** (martes): Reflexión sobre proceso, identificación de mejoras.

Esto es una adaptación del proceso ágil de desarrollo, aplicado a un trabajo individual. Se aplicaron estas buenas prácticas para asegurar un proceso ordenado, trazable y adaptable, manteniendo la disciplina propia de un entorno ágil y garantizando incrementos funcionales verificables en cada sprint.

Gestión mediante GitHub Projects

GitHub Projects⁴ se utilizó como herramienta de gestión, implementando tablero Kanban digital integrado directamente con el repositorio de código. El tablero se organizó en cuatro columnas que representan estados del flujo de trabajo:

- **Backlog:** Ideas y tareas futuras no priorizadas.
- **To Do:** Tareas seleccionadas para el sprint actual, priorizadas.
- **In Progress:** Trabajo activo, limitado a 2-3 tareas simultáneas para evitar trabajo en proceso excesivo.
- **Done:** Tareas completadas, probadas y validadas.

La Figura 7.28 muestra el tablero Kanban en GitHub Projects durante una semana típica de desarrollo, con tareas distribuidas en las diferentes columnas según su estado de avance.

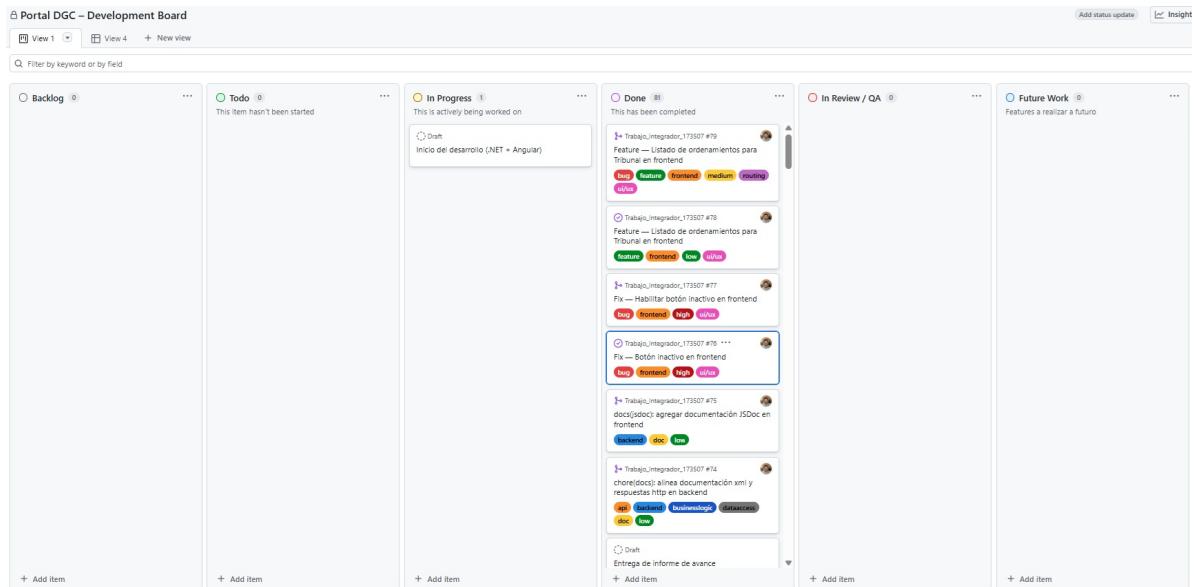


Figura 7.28: Tablero Kanban en GitHub Projects mostrando flujo de trabajo

⁴El proyecto está disponible en https://github.com/GabrielLutz/Trabajo_Integrador_173507.git

Cada tarea se documentó con:

- Título descriptivo siguiendo convención: [Tipo] Descripción breve.
- Descripción detallada del trabajo a realizar.
- Etiquetas de categorización (Investigación, Análisis, Diseño, Desarrollo, Testing, Documentación).
- Fechas de inicio y fin para seguimiento de tiempo invertido.
- Asignación a sprint específico.

Adicionalmente, GitHub Projects permitió visualizar el progreso mediante vista de tabla (Figura 7.29) que facilitó el seguimiento temporal de las tareas y la identificación de dependencias críticas.

Title	Assignees	Status	Linked pull requests	Start Date	End Date	Category
1 Feature — Listado de ordenamientos para Tribunal en frontend #79	GabrielLutz	Done				
2 Feature — Listado de ordenamientos para Tribunal en frontend #78	GabrielLutz	Done	#79			
3 Fix — Habilitar botón inactivo en frontend #77	GabrielLutz	Done				
4 Fix — Botón inactivo en frontend #76	GabrielLutz	Done	#77			
5 docs(doc): agregar documentación JSDoc en frontend #75	GabrielLutz	Done				
6 chore(doc): alinea documentación xml y respuestas http en backend #74	GabrielLutz	Done				
7 Entrega de informe de avance	GabrielLutz	Done		Nov 18, 2025	Nov 18, 2025	Documentación
8 feature/dataaccess-repositories — Implementar repositorios concretos #6		Done	#87			
9 feat(data): agrega repositorios concretos por módulo #7		Done				
10 feature/businesslogic — Implementar servicios de negocio #8		Done	#9			
11 feature/businesslogic — Implementar servicios de negocio #9		Done				
12 feature/webapi — Configurar host e implementar controladores principales #10		Done	#11			
13 feat(api): configuración del host y controllers principales #11		Done				
14 Kick-off del proyecto		Done		Sep 17, 2025	Sep 20, 2025	Relevamiento
15 Entrevistas y análisis de necesidades (ESRE)		Done		Sep 17, 2025	Sep 24, 2025	Relevamiento
16 Análisis intermedio y revisión de hallazgos		Done		Sep 24, 2025	Oct 7, 2025	Analisis / Consolidación
17 Ingeniería de reversa - Portales públicos		Done		Oct 8, 2025	Oct 15, 2025	Investigación
18 Diseño de workflows y modelo de dominio		Done		Oct 15, 2025	Oct 22, 2025	Diseño
19 Seguridad y autenticación (supuestos/protocolos)		Done		Oct 22, 2025	Oct 31, 2025	Arquitectura
20 Inicio del desarrollo (.NET + Angular)		In Progress		Oct 31, 2025		Desarrollo
21 Test — Pruebas de integración WebApi #70		Done	#71			
22 tests/business+webapi — Agregar proyecto de pruebas y tests de servicios/controlador #68		Done	#69			
23 fix/backend — Inscripciones: incluir AutodefinitionLey en GetByLlamadoId #66		Done	#67			

Figura 7.29: Vista de tabla del Development Board mostrando timeline y categorías

Esta organización proporcionó visibilidad completa del progreso en cualquier momento, facilitó identificación de bloqueos, y generó registro histórico auditable del proceso de desarrollo.

Prácticas de Control de Versiones

El código fuente se gestionó mediante Git [64], implementando estrategia de branching Git Flow [65] simplificada:

- **Rama main:** Código estable, potencialmente desplegable. Protegida contra pushes directos.
- **Rama develop:** Integración de características completadas. Base para nuevas ramas de feature.
- **Ramas feature/*:** Desarrollo de funcionalidades específicas en aislamiento.

La Figura 7.30 presenta el gráfico de red del repositorio Git mostrando el historial de commits y la topología de ramas durante dos meses de desarrollo activo. Se observan las ramas main, develop, y múltiples ramas feature/* que se fusionan periódicamente.

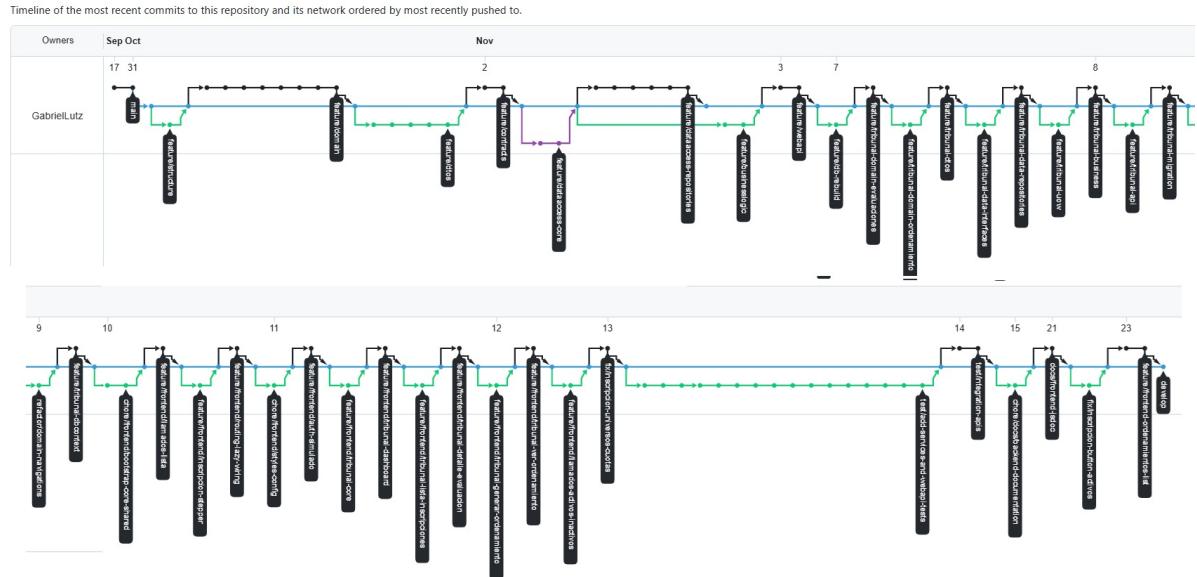


Figura 7.30: Gráfico de red de Git mostrando estrategia de branching y flujo de trabajo

Conventional Commits

Los mensajes de commit siguieron convención Conventional Commits^[66], estructurando cada mensaje con prefijo que indica tipo de cambio: `feat` (para nuevas funcionalidades), `fix` (para correcciones), `docs` (para documentación), `refactor` (para mejoras de código sin cambio funcional), `test` (para pruebas).

La Figura 7.31 muestra el historial de commits del proyecto evidenciando la aplicación consistente de esta convención en todas las contribuciones.

chore(solution): crea solución PortalDGC en la raíz del repositorio	fix(dataaccess/inscripcion): incluir AutodefinicionLey en GetByLlamad...
 GabrielLutz committed on Oct 31	 GabrielLutz committed 3 weeks ago
chore(structure): crea estructura base de proyectos por capas (.NET 8)	Merge pull request #67 from GabrielLutz/fix/inscripcion-universos-cuotas
 GabrielLutz committed on Oct 31	 GabrielLutz authored 3 weeks ago
Merge pull request #1 from GabrielLutz/feature/structure	chore(tests): inicializa proyecto de pruebas xUnit con Mog
 GabrielLutz authored on Oct 31	 GabrielLutz committed 3 weeks ago
feat(domain): agrega entidades de postulante (Postulante, Autodefinic...	test(services/validacion): agrega ValidacionServiceTests (xUnit + cas...
 GabrielLutz committed on Oct 31	 GabrielLutz committed 3 weeks ago
feat(domain): agrega requisitos y méritos (RequisitoPostulante, Requi...	test(services/postulante): agrega tests de obtención/actualización y ...
 GabrielLutz committed on Oct 31	 GabrielLutz committed 3 weeks ago
build(db): agrega migración inicial InitialCreate	docs(jsdoc): agregar documentación JSDoc en frontend
 GabrielLutz committed last month	 GabrielLutz committed 2 weeks ago
Merge pull request #13 from GabrielLutz/feature/db-rebuild	Merge pull request #75 from GabrielLutz/docs/frontend-jsdoc
 GabrielLutz authored last month	 GabrielLutz authored 2 weeks ago
feat(domain/tribunal): agrega entidades de evaluación (Prueba/Evaluac...	fix(ui): habilitar botón en fix/inscripcion-button-activos que estaba...
 GabrielLutz committed last month	 GabrielLutz committed 2 weeks ago
Merge pull request #15 from GabrielLutz/feature/tribunal-domain-evalu...	Merge pull request #77 from GabrielLutz/fix/inscripcion-button-activos
 GabrielLutz authored last month	 GabrielLutz authored 2 weeks ago

Figura 7.31: Historial de commits siguiendo convención Conventional Commits

Esta convención facilita generación automática de changelogs, permite filtrado de commits por tipo, y mejora la comprensión rápida del historial de cambios mediante semántica clara y consistente.

Pull Requests como Mecanismo de Revisión

Aunque el equipo de desarrollo es individual, se utilizaron Pull Requests (PRs) como punto de control de calidad. Antes de fusionar una rama `feature/*` a `develop`, se creaba PR que:

1. Documenta cambios realizados con descripción detallada.
2. Ejecuta automáticamente suite de tests unitarios mediante GitHub Actions [67].
3. Verifica que el código compila sin errores ni warnings.
4. Permite revisión del código escrito antes de integración.

La Figura 7.32 muestra el listado de Pull Requests cerrados durante el desarrollo, evidenciando el uso sistemático de etiquetas semánticas (bug, feature, frontend, backend, test, doc) para categorizar y priorizar cambios.

	Author	Label	Projects	Milestones	Reviews	Assignee	Sort
<input type="checkbox"/>  Open  43 Closed							
<input type="checkbox"/>  Feature — Listado de ordenamientos para Tribunal en frontend      							
<small>#79 by GabrielLutz was merged last week</small>							
<input type="checkbox"/>  Fix — Habilitar botón inactivo en frontend    							
<small>#77 by GabrielLutz was merged 2 weeks ago</small>							
<input type="checkbox"/>  docs(jSDoc) : agregar documentación JSDoc en frontend   							
<small>#75 by GabrielLutz was merged 2 weeks ago</small>							
<input type="checkbox"/>  chore(docs) : alinea documentación xml y respuestas http en backend      							
<small>#74 by GabrielLutz was merged 2 weeks ago</small> 							
<input type="checkbox"/>  test(integration) : agrega batería completa de pruebas end-to-end del sistema   							
<small>#71 by GabrielLutz was merged 3 weeks ago</small>							
<input type="checkbox"/>  test(*) : agrega batería completa de tests unitarios y de controlador    							
<small>#69 by GabrielLutz was merged 3 weeks ago</small>							
<input type="checkbox"/>  fix(dataaccess/inscripcion) : incluir AutodefinicionLey en GetByLlamadold    							
<small>#67 by GabrielLutz was merged 3 weeks ago</small>							
<input type="checkbox"/>  Feature — Llamados: Vista de Activos e Inactivos    							
<small>#65 by GabrielLutz was merged 3 weeks ago</small>							
<input type="checkbox"/>  feat(tribunal) : ver ordenamiento (detalle, filtros, publicar, export ...    							
<small>#63 by GabrielLutz was merged 3 weeks ago</small>							
<input type="checkbox"/>  feature/frontend — Tribunal: Generar Ordenamiento (wizard + preview + resultado)  		  					
<small>#61 by GabrielLutz was merged 3 weeks ago</small>							
<input type="checkbox"/>  feature/frontend — Tribunal: Detalle de Evaluación (pruebas + méritos + requisitos)     							
<small>#59 by GabrielLutz was merged 3 weeks ago</small>							
<input type="checkbox"/>  feature/frontend — Tribunal: Lista de Inscripciones (filtros + progreso + navegación)     							
<small>#57 by GabrielLutz was merged 3 weeks ago</small>							
<input type="checkbox"/>  feature/frontend — Tribunal: Dashboard con estadísticas y accesos rápidos    							
<small>#55 by GabrielLutz was merged 3 weeks ago</small>							

Figura 7.32: Listado de Pull Requests con etiquetas semánticas y estado de integración

Cada PR incluye:

- Título descriptivo siguiendo formato: [Tipo] --- Descripción breve
- Número de PR para trazabilidad (#61, #63, #65, etc.)
- Etiquetas múltiples indicando módulo afectado, tipo de cambio, y prioridad
- Estado de integración (merged, open, closed)
- Vinculación automática con issues relacionados

Esta práctica, aunque parezca redundante en desarrollo individual, establece disciplina que

facilita futura colaboración con otros desarrolladores y proporciona documentación clara de decisiones de diseño. Para la misma se utilizó la herramienta de Claude.ia

7.6 Fase 6: Testing y Aseguramiento de Calidad

Estrategia de Testing

Se implementó estrategia de testing en múltiples niveles siguiendo pirámide de testing[68]:

Tests unitarios (base de la pirámide).

Se escribieron tests unitarios para servicios de lógica de negocio utilizando xUnit [69] como framework de testing y Moq para creación de mocks. El objetivo fue alcanzar cobertura superior al 80 % en servicios críticos: IncripcionService, TribunalService, ValidacionService, OrdenamientoService.

Ejemplo de test unitario del servicio ValidacionService:

```
1 [Theory]
2 [InlineData("1.234.567-8", true)]
3 [InlineData("12345678", true)]
4 [InlineData("abc.def.ghi-j", false)]
5 public void ValidarCedulaIdentidad_VariosFormatos_RetornaResultadoEsperado(
6     string cedula, bool esperado){
7     // Act
8     var resultado = _sut.ValidarCedulaIdentidad(cedula);
9
10    // Assert
11    Assert.Equal(esperado, resultado.Success);
12    Assert.Equal(esperado, resultado.Data);
13 }
```

Tests de integración (medio de la pirámide).

Se implementaron tests de integración que verifican interacción correcta entre servicios y capa de acceso a datos. Estos tests utilizan base de datos en memoria (SQLite) para aislar tests de la base de datos productiva.

Tests end-to-end (cima de la pirámide).

Se realizaron tests manuales de flujos completos navegando la aplicación como usuario final. Si bien se consideró automatización mediante Selenium, restricciones de tiempo hicieron priorizar los tests unitarios y de integración, ya que proporcionan mayor retorno inmediato.

Integración Continua

Se configuró pipeline básico de integración continua mediante GitHub Actions que ejecuta automáticamente al hacer push al repositorio:

1. Restaura dependencias de NuGet [44].
2. Compila solución completa en modo Release.
3. Ejecuta suite completa de tests unitarios.
4. Reporta resultados en interfaz de GitHub.

Si cualquier paso falla, el build se marca como fallido, alertando inmediatamente sobre problemas introducidos. Esta validación temprana reduce tiempo entre introducción de defecto y su detección.

7.7 Fase 7: Documentación Técnica

Paralelamente al desarrollo, se mantuvo documentación técnica en tres niveles:

Documentación Inline

El código fuente incluye XML comments (C#) y JSDoc (TypeScript) que documentan propósito, parámetros, valores de retorno y excepciones de clases, métodos y funciones públicas.

Ejemplo:

```
1  /// <summary>
2  /// Genera ordenamiento aplicando cuotas de acción afirmativa.
3  /// </summary>
4  /// <param name="inscripciones">Lista de inscripciones aprobadas</param>
5  /// <param name="config">Configuración de cuotas a aplicar</param>
6  /// <returns>Ordenamiento con posiciones asignadas</returns>
7  /// <exception cref="InvalidOperationException">
8  ///   Si no hay suficientes inscripciones para cubrir cuotas.
9  /// </exception>
```

Diagramas Técnicos

Se produjeron diagramas que documentan aspectos clave de arquitectura y diseño:

- Diagrama de arquitectura en capas (Mermaid [70]).
- Diagrama de despliegue con VLANs (Draw.io [71]).
- Modelo de dominio con entidades y relaciones (Astah [72]).
- Flujos de proceso para postulante y tribunal (BPNM.io y Mermaid [70]).

- Diagrama de paquetes mostrando dependencias entre módulos (Draw.io [71]).

Estos diagramas se mantuvieron sincronizados con implementación mediante revisiones periódicas.

Herramientas y Tecnologías Utilizadas

La siguiente tabla 7.2 resume herramientas y tecnologías empleadas en cada fase:

Fase	Herramientas
Trabajo de campo	Microsoft Teams [73] (videoconferencias), Notion [74] (organización de notas)
Análisis	Excel [29] (matriz comparativa), Miro [75] (mapas de afinidad), Lucidchart [76] (flujos preliminares)
Diseño	Draw.io [71] (arquitectura, despliegue), Mermaid [70] (flujos), Framer [62] (prototipo interactivo), Astah [72] (Dominio)
Desarrollo	Visual Studio Code [77], Visual Studio 2022 [78], SQL Server Management Studio [49], Postman [79] (testing API)
Testing	xUnit [69], Moq, Coverage de VS (cobertura)
Gestión	GitHub [80] (repositorio, Projects, Actions), Git [64] (control versiones)
Documentación	Overleaf [81] (informes académicos), Markdown (README, docs técnicas), JSDoc/XML comments [82][83]

Tabla 7.2: Herramientas utilizadas por fase del proyecto

Cronograma Ejecutado

Como muestra la siguiente tabla 7.3, el proyecto se desarrolló durante 10 semanas con la distribuido en las fechas indicadas.

Fase	Período	Duración	Estado
Kick-off	Sep 17-20	3 días	Completado
Relevamiento	Sep 17-24	1 semana	Completado
Consolidación	Sep 24 - Oct 7	2 semanas	Completado
Ing. Inversa	Oct 8-15	1 semana	Completado
Diseño	Oct 15-22	1 semana	Completado
Arquitectura	Oct 22-31	1.5 semanas	Completado
Desarrollo MVP	Oct 31 - Nov 30	4 semanas	Completado

Tabla 7.3: Cronograma de fases del proyecto

7.8 Uso de herramientas de Inteligencia Artificial

Durante el desarrollo de este trabajo se utilizaron herramientas basadas en inteligencia artificial como apoyo en tareas específicas de documentación, búsqueda de información, y depuración de código. Es importante destacar que estas herramientas actuaron como asistentes técnicos, no como autores del trabajo, y que todas las decisiones de diseño, arquitectura e implementación fueron realizadas por el autor bajo supervisión académica.

Documentación y Formato

Para la conversión de información técnica a formato LaTeX, generación de índices y estructuración de tablas se utilizó Gemini Flash 2.5 [84]. Esta herramienta facilitó la traducción de contenido entre formatos, especialmente útil para la estructuración de tablas complejas y la generación de código LaTeX sintácticamente correcto. Para la búsqueda y validación de referencias bibliográficas se contrastó la información proporcionada por el modelo con Google Scholar,

verificando la exactitud de citas y disponibilidad de fuentes.

Documentación de Código

La generación de comentarios inline (XML comments [83] para C# y JSDoc [82] para TypeScript) se realizó con asistencia del modo agente de Visual Studio Code [77] utilizando Claude Sonnet 4.5 [85] y Codex 5.1 (preview) [86]. Estas herramientas analizaron el código existente y propusieron documentación descriptiva de métodos, parámetros y valores de retorno, que posteriormente fue revisada y ajustada manualmente para garantizar precisión técnica. También se utilizaron para diagnóstico y corrección de errores puntuales durante el desarrollo.

Asistencia al Desarrollo

Durante la implementación del código en Visual Studio 2022 [78] se utilizó GitHub Copilot [87] para asistencia en autocompletado de código, sugerencia de patrones comunes, y generación de código. Esta herramienta aceleró la escritura de código repetitivo pero todas las implementaciones fueron revisadas, adaptadas y validadas mediante pruebas unitarias.

Búsqueda de Información y Resolución de Problemas

Para consultas técnicas, búsqueda de información sobre mejores prácticas, y resolución de errores de compilación o ejecución se utilizaron ChatGPT-5.1 [88] y Claude Sonnet 4.5 [85]. Estas herramientas proporcionaron explicaciones de conceptos técnicos, sugerencias de soluciones a problemas específicos, y ejemplos de código que fueron adaptados al contexto del proyecto.

En la elaboración de diagramas técnicos, especialmente aquellos en formato Mermaid [70] y Lucid Chart [76], se solicitó asistencia para la sintaxis correcta y estructuración visual de flujos complejos. Los diagramas generados fueron revisados y modificados para reflejar fielmente la arquitectura implementada.

Para el desarrollo de pruebas unitarias con xUnit [69], se utilizó asistencia de IA para la generación de casos de prueba iniciales, especialmente para identificar escenarios y estructu-

rar correctamente los patrones Arrange-Act-Assert [89]. Los tests generados fueron validados, extendidos y en muchos casos reescritos para garantizar cobertura adecuada.

En la gestión de repositorio Git [64], se utilizó apoyo de IA para la redacción de mensajes de commit siguiendo convención Conventional Commits [66], la estructuración de descripciones de Pull Requests [90], y la documentación de cambios en el código.

Consideraciones Éticas y Académicas

El uso de herramientas de IA se realizó siguiendo principios de transparencia y responsabilidad académica:

- Todas las sugerencias de código fueron revisadas, comprendidas y validadas antes de su integración
- La arquitectura del sistema, las decisiones de diseño, y la lógica de negocio son producto del análisis y razonamiento del autor
- La redacción de este documento académico, incluyendo análisis, conclusiones y reflexiones, fue realizada por el autor
- Las herramientas de IA actuaron exclusivamente como asistentes técnicos en tareas de soporte, no como sustitutos del trabajo intelectual requerido
- Se reconoce explícitamente el uso de estas herramientas en cumplimiento con estándares de integridad académica

El uso responsable de herramientas de IA en el desarrollo de software representa una tendencia creciente en la industria, y su incorporación en este proyecto académico refleja prácticas profesionales contemporáneas sin comprometer el rigor académico requerido.

Análisis de Resultados

Esta sección presenta los resultados obtenidos durante el desarrollo del Portal DGC, organizados en función de los objetivos establecidos en la metodología. Se analizan tanto los aspectos técnicos de la implementación como la validación de los requisitos definidos, proporcionando una evaluación integral del Proof of Concept desarrollado.

8.1 Alcance de la Implementación

El desarrollo del MVP (Producto Mínimo Viable) se centró en validar la viabilidad técnica y funcional de los flujos críticos del sistema, priorizando las funcionalidades esenciales que permiten demostrar el valor del portal sin desarrollar características secundarias. Esta estrategia permitió concentrar esfuerzos en probar los conceptos fundamentales mientras se mantiene un alcance manejable dentro de las restricciones temporales del proyecto académico.

Estado General del Proyecto

Al momento de la entrega, el proyecto alcanzó niveles de completitud en sus distintas fases. La fase de elicitation y relevamiento se completó en su totalidad, realizando entrevistas semiestructuradas con stakeholders clave que permitieron documentar las necesidades reales del proceso actual. La fase de diseño también se completó íntegramente, produciendo la arquitectura del sistema en tres capas, el modelo de dominio completo con 21 entidades, diagramas de flujo de procesos, y un prototipo interactivo de alta fidelidad validado con usuarios reales.

El desarrollo del MVP alcanzó aproximadamente un 70 % de completitud, con los dos workflows principales funcionales y validados, mientras que funcionalidades secundarias de seguridad, notificaciones y administración quedaron pendientes. La documentación técnica se encuentra en un 85 % de avance, habiendo generado la Especificación de Requisitos (ESRE [N]), casos de uso detallados, diagramas arquitectónicos y documentación inline del código, quedando pen-

diente únicamente la documentación de despliegue y manuales de usuario finales.

Cumplimiento de Requisitos

La Tabla 8.1 resume el estado de implementación de los requisitos definidos durante la fase de análisis. De los 21 requisitos funcionales especificados, se implementaron 14 (67 %), priorizando aquellos que constituyen el núcleo de valor del sistema. En cuanto a requisitos no funcionales, de los 16 definidos, 9 fueron validados satisfactoriamente y 4 de forma parcial, representando un 56 % de validación completa.

Categoría	Total Definidos	Implementados	Porcentaje
Requisitos Funcionales	21	14	67 %
Requisitos No Funcionales	16	9 validados, 4 parciales	56 % validados
Casos de Uso	20	9	45 %

Tabla 8.1: Estado de implementación de requisitos

Los requisitos funcionales implementados abarcan el flujo completo de inscripción del postulante, el sistema de evaluación del tribunal (calificación de méritos y pruebas con cálculo automático de puntajes), y la generación de ordenamientos con aplicación automática de cuotas de acción afirmativa. El detalle completo de requisitos implementados se encuentra en el Apéndice H.

Los requisitos pendientes corresponden principalmente a funcionalidades de infraestructura críticas para producción pero no esenciales para validar la viabilidad del concepto, como autenticación JWT [60] con refresh tokens, sistema RBAC de roles y permisos [115], sistema de notificaciones por email/WhatsApp, y módulos de administración avanzada.

8.2 Validación de Workflows Principales

La estrategia de desarrollo se enfocó en implementar y validar los dos flujos críticos del sistema, que representan el núcleo de valor del portal y permiten demostrar la factibilidad técnica de la solución propuesta.

Workflow del Postulante

El flujo completo de inscripción fue implementado, permitiendo que un postulante complete todo el proceso desde la consulta de llamados disponibles hasta la confirmación de su inscripción. El workflow comienza con una interfaz de consulta que presenta llamados activos, mostrando información resumida que permite al usuario identificar rápidamente las oportunidades relevantes. Al seleccionar un llamado de interés, el sistema despliega una vista detallada que incluye descripción del cargo, requisitos excluyentes, criterios de evaluación de méritos con puntajes máximos por categoría, pruebas a rendir, cronograma de etapas, y cupos reservados por ley.

Una vez que el postulante decide inscribirse, el sistema lo guía a través de un workflow progresivo de 6 pasos que estructura la recolección de información de manera lógica y validada. El primer paso captura la selección del departamento donde desea trabajar, seguido por la autodefinición voluntaria según categorías legales (afrodescendiente, trans, con discapacidad), cumpliendo con las Leyes 19.122 [7], 19.684 [8] y 18.651 [9]. El tercer paso solicita la declaración de cumplimiento de requisitos excluyentes mediante checkboxes individuales que obligan al postulante a confirmar explícitamente cada condición.

El cuarto paso permite la carga de méritos valorables organizados por categoría (estudios formales, cursos, experiencia laboral), mostrando el puntaje máximo posible y permitiendo adjuntar documentación probatoria. Si el postulante declaró tener discapacidad, el quinto paso se habilita automáticamente para solicitar apoyos específicos necesarios durante las pruebas (intérprete de señas, tiempo adicional, formato accesible de materiales). Finalmente, el sexto paso

presenta un resumen completo de toda la información ingresada, permitiendo revisar antes de confirmar. Al confirmar, el sistema genera un código único de inscripción que se muestra en pantalla.

El workflow incorpora validaciones tanto en el cliente (Angular [16]) como en el servidor (ASP.NET Core [23]), verificando campos obligatorios, formatos de datos, tamaños y tipos de archivos adjuntos, coherencia de datos entre pasos, y reglas de negocio como no permitir inscripción duplicada para el mismo llamado. Adicionalmente, el panel "Mis Inscripciones" permite a cada postulante visualizar todas sus inscripciones activas, consultar su estado actual y acceder al detalle de cada una.

Workflow del Tribunal Evaluador

El módulo de evaluación implementa el flujo completo que permite a los miembros del tribunal calificar inscripciones y generar ordenamientos automáticos. La pantalla inicial presenta un dashboard con métricas en tiempo real del proceso de evaluación: total de inscripciones recibidas, pendientes de evaluación, evaluadas completamente, promedio de puntaje obtenido, y distribución por cupos.

La interfaz tabular de inscripciones ofrece capacidades avanzadas de filtrado por distintos ítems. Al seleccionar una inscripción, el sistema presenta una vista completa organizada en pestañas que facilita la navegación: datos generales del postulante, lista detallada de méritos con formularios de calificación individuales, resultados de pruebas con validación automática de umbrales de aprobación, y acceso centralizado a documentos adjuntados.

El sistema calcula automáticamente el puntaje total obtenido en cada categoría y lo compara con el máximo posible, validando que las calificaciones asignadas no excedan los límites establecidos. Para pruebas escritas, prácticas u orales, el sistema valida el cumplimiento del umbral mínimo de aprobación y marca automáticamente si cada prueba fue aprobada o no.

Una vez completadas las evaluaciones, el tribunal puede generar los ordenamientos finales

mediante un algoritmo que implementa la lógica especificada en las bases del llamado. El proceso filtra inscripciones que cumplan todos los requisitos excluyentes y hayan aprobado todas las pruebas obligatorias, calcula el puntaje total de cada inscripción, ordena por puntaje de mayor a menor, y aplica automáticamente las cuotas legales (8 % afrodescendientes, 1 % trans, 4 % discapacidad). El sistema genera tanto el ordenamiento general como ordenamientos por cupo, identificando las posiciones que quedan cubiertas considerando las plazas disponibles. Los ordenamientos generados pueden exportarse en dos formatos: Excel (.xlsx) y PDF¹.

8.3 PoC Implementado

Esta sección presenta las interfaces principales del MVP desarrollado, mostrando la implementación final de los flujos validados. A diferencia del prototipo de alta fidelidad presentado en la Fase 4 de la metodología, estas capturas corresponden al sistema funcional implementado en Angular 19 y ASP.NET Core 8.

Los videos explicativos del PoC están en: <https://www.loom.com/share/0b96170c7eb54caa88c53e59>
<https://www.loom.com/share/60a084a51704482980946702e83da910>

Módulo del Postulante

El módulo del postulante permite a los aspirantes gestionar su participación en llamados públicos de manera completamente digital.

¹Si bien está la estructura, la feature no está implementada

Autenticación y Registro

La captura de pantalla muestra la página de inicio de sesión del Portal DGC. El encabezado indica "Portal DGC Dirección General de Casinos". La sección principal titulada "Portal de la Dirección General de Casinos" contiene campos para "Usuario" (con placeholder "Ingresá tu usuario") y "Contraseña" (con placeholder "Ingresá tu contraseña"). Un botón azul "Ingresar" se encuentra debajo de los campos. Abajo de este sección, un enlace "¿No tenés usuario? Registrate ahora" y otro enlace "Ir al panel del tribunal".

Figura 8.1: Pantalla de inicio de sesión del Portal DGC

La Figura 8.1 muestra la pantalla de autenticación implementada, que permite el inicio de sesión con usuario y contraseña. El diseño mantiene la identidad institucional de la DGC, con el encabezado azul característico, y ofrece opciones de registro para nuevos usuarios. El acceso al panel del tribunal evaluador está en esta pantalla porque no está implementada la funcionalidad de autenticación.

La captura de pantalla muestra el formulario de registro en el Portal DGC. El encabezado indica "Portal DGC Dirección General de Casinos". La sección principal titulada "Regístrate en el Portal DGC" pide al usuario "Completa tus datos para la creación de un usuario para el portal". Los campos incluyen:

- Datos personales:** Campos para "Nombre" (placeholder "Ingresá tu nombre"), "Apellido" (placeholder "Ingresá tu apellido") y "Cédula de identidad" (placeholder "Ej: 4.567.890-1").
- Contacto:** Campos para "Correo electrónico" (placeholder "nombre@ejemplo.com") y "Celular" (placeholder "Ej: 099123456").
- Datos de acceso:** Campos para "Usuario" (placeholder "Elegí un usuario") y "Contraseña" (placeholder "Mínimo 6 caracteres").

Al final del formulario hay un botón "Volver al inicio" y un botón azul "Registrar".

Figura 8.2: Formulario de registro de nuevos postulantes

El formulario de registro (Figura 8.2) captura los datos mínimos necesarios para crear una cuenta: nombre, apellido, cédula de identidad, correo electrónico, celular, usuario y contraseña. El formulario implementa validaciones en tiempo real tanto del lado del cliente como del servidor, garantizando la integridad de los datos ingresados.

Gestión del Perfil

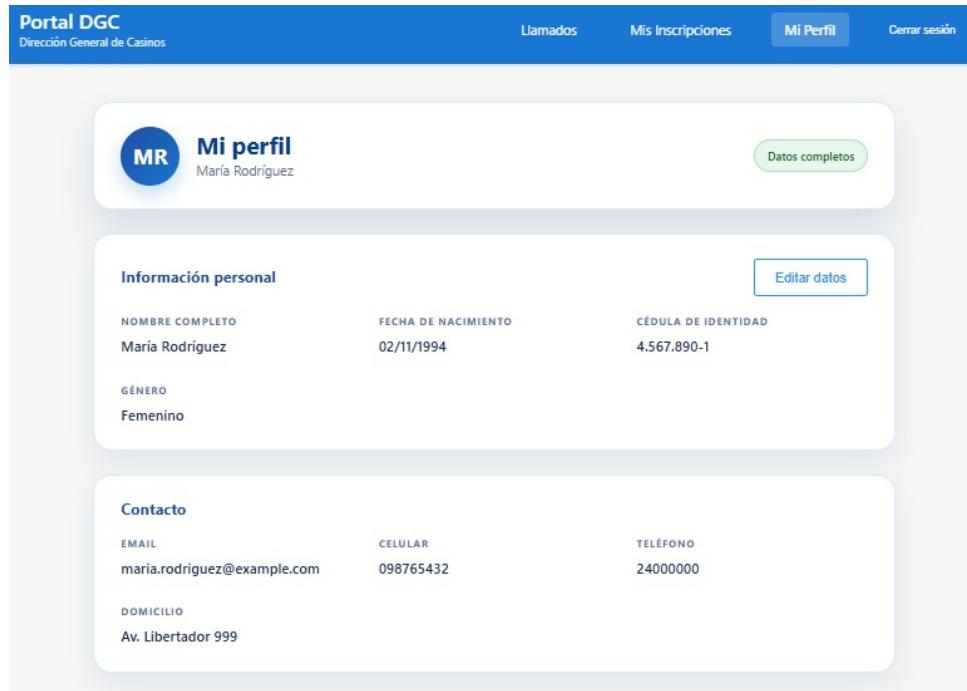


Figura 8.3: Vista del perfil del postulante con información personal y de contacto

La pantalla de perfil (Figura 8.3) presenta de forma organizada la información personal del postulante en dos secciones principales: "Información personal" (nombre completo, fecha de nacimiento, cédula de identidad, género) y "Contacto" (email, celular, teléfono fijo, domicilio). La interfaz incluye un indicador de completitud del perfil ("Datos completos") y ofrece funcionalidad de edición mediante el botón "Editar datos". La navegación superior permite acceso rápido a las secciones principales del sistema.

Consulta de Llamados

The screenshot shows the 'Llamados' (Calls) section of the DGC portal. At the top, there are navigation links: 'Portal DGC', 'Dirección General de Casinos', 'Llamados' (selected), 'Mis Inscripciones', 'Mi Perfil', and 'Cerrar sesión'. Below this, a header bar indicates 'Activos 2' (Active 2) and 'Inactivos 1' (Inactive 1). Two job listings are displayed in cards:

- Llamado Público DGC - Crupier (Especializado III) 2025**
11 Días Restantes
Cupier para juegos tradicionales (ruleta, naipes, dados).
Apertura: 17/11/2025 Cierre: 17/12/2025
[Ver Detalle](#) [Inscribirse](#)
- Llamado Público DGC - Fiscal III 2025**
11 Días Restantes
Administrativo y fiscalizador en salas de juego (caja, atención al público, fiscalización).
Apertura: 17/11/2025 Cierre: 17/12/2025
[Ver Detalle](#) [Inscribirse](#)

Figura 8.4: Listado de llamados públicos activos e inactivos

La vista de llamados (Figura 8.4) permite filtrar entre llamados activos e inactivos mediante pestañas. Cada tarjeta presenta información clave, título del llamado, descripción del cargo, fechas de apertura y cierre, y días restantes para inscribirse resaltados. Los botones "Ver Detalle" e "Inscribirse" facilitan la navegación hacia las acciones principales del flujo de postulación.

Seguimiento de Inscripciones

The screenshot shows the 'Mis Inscripciones' (My Applications) panel. At the top, there are navigation links: 'Portal DGC', 'Dirección General de Casinos', 'Llamados' (disabled), 'Mis Inscripciones' (selected), 'Mi Perfil', and 'Cerrar sesión'. Below this, a header bar indicates 'Mis Inscripciones' (My Applications). The main area displays two active applications in cards:

- Llamado Público DGC - Fiscal III 2025**
Montevideo Pendiente
FECHA DE INSCRIPCIÓN
27/11/2025 14:50
[Ver detalle](#)
- Llamado Público DGC - Crupier (Especializado III) 2025**
Canelones Pendiente
FECHA DE INSCRIPCIÓN
26/11/2025 13:05
[Ver detalle](#)

Figura 8.5: Panel "Mis Inscripciones" mostrando postulaciones activas

El panel de mis inscripciones (Figura 8.5) permite al postulante consultar el estado de todas sus postulaciones en un formato de tarjetas. Cada tarjeta presenta el título del llamado, departamento seleccionado, fecha de inscripción formateada, y estado actual identificado con badge de color (azul para "Pendiente"). El botón "Ver detalle" proporciona acceso a la información completa de cada inscripción, permitiendo al postulante revisar toda la documentación presentada y el progreso de su evaluación.

Módulo del Tribunal Evaluador

El módulo del tribunal proporciona herramientas especializadas para la evaluación de inscripciones y generación de ordenamientos finales.

Dashboard de Evaluación

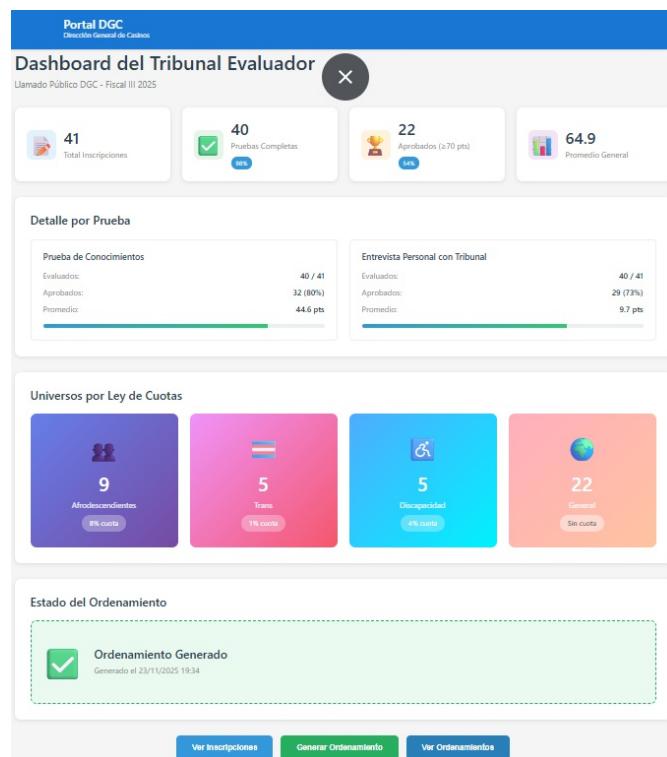


Figura 8.6: Dashboard del tribunal evaluador con métricas en tiempo real

El dashboard (Figura 8.6) presenta una vista del proceso de evaluación para el llamado "Lla-

mado Público DGC - Fiscal III 2025". Las métricas principales se muestran en tarjetas destacadadas: inscripciones totales, pruebas completadas, aprobados que superan el 70 % del puntaje y un promedio general.

La sección "Detalle por Prueba" desglosa el progreso de cada tipo de evaluación mediante barras de progreso visuales, separadas por tipo y mostrando el estado de evaluadas , aprobados y promedio de pts.

La visualización de "Universos por Ley de Cuotas" distribuye los postulantes en tarjetas de colores distintivos, afrodescendientes en morado (8 % cuota), trans en rosado (1 % cuota), discapacidad en celeste (4 % cuota) y sin cuota en coral. El "Estado del Ordenamiento" indica si fue generado un ordenamiento o no, en este caso, muestra un indicador verde confirmando que fue generado el 23/11/2025 a las 19:34.

Los botones de acción principales permiten "Ver inscripciones" para revisar el listado completo, "Generar Ordenamiento" para crear nuevas listas de prelación, y "Ver Ordenamientos" para consultar los ordenamientos ya publicados.

Búsqueda y Filtrado de Inscripciones

Postulante	Departamento	Universos	Progreso	Puntaje	Estado	Acciones
Ana Martínez C: 3.987.654-2	Montevideo	AFRODESCENDIENTE	100% Pruebas: 2/2 Meritos: 4/4	89.0 pts Pruebas: 72 Meritos: 17	Completa	Ver detalle
Helena Torres C: 5.394.820-1	Montevideo	AFRODESCENDIENTE	100% Pruebas: 2/2 Meritos: 4/4	70.0 pts Pruebas: 61 Meritos: 9	Completa	Ver detalle

Figura 8.7: Interfaz de filtrado y listado de inscripciones para evaluar

La interfaz de evaluación (Figura 8.7) ofrece filtros avanzados combinables, como búsqueda

por nombre o cédula, selección de departamento mediante dropdown, filtro por estado de evaluación (“Todos”, “Pendiente”, “En Evaluación”, “Completa”), y segmentación por universo de cuotas (“Todos”, “Afrodescendiente”, “Trans”, “Discapacidad”, “General”). El botón ”Limpiar filtros” permite resetear todos los criterios de búsqueda.

La tabla de resultados presenta los inscripciones con las columnas: Postulante (nombre y cédula), Departamento, Universos (badge de color según categoría), Progreso (barras visuales mostrando detalle, Pruebas y Méritos), Puntaje (destacado en verde si supera el umbral de aprobación, con el detalle de los puntajes obtenidos y el Estado de la evaluación (Pendiente, en Proceso, Completa)), y botón ”Ver detalle” para acceder a la evaluación individual.

Generación de Ordenamientos

Figura 8.8: Configuración del ordenamiento con aplicación de cuotas legales

La pantalla de generación de ordenamientos (Figura 8.8) implementa un workflow de 3 pasos claramente identificados en el header: "1 Configuración", "2 Previsualización", y "3 Resultado".

El paso de configuración permite establecer el "Puntaje Mínimo de Aprobación" (70 puntos en el ejemplo) con una nota aclaratoria: "Los postulantes con puntaje inferior no serán incluidos en el ordenamiento". El checkbox "Aplicar Cuotas de Ley" está activado, mostrando los porcentajes legales: "8 % Afrodescendiente | 1 % Trans | 4 % Discapacidad".

La sección "Información del Llamado" presenta métricas clave mediante tarjetas destacadas: inscripciones totales en azul y aprobados (mayor al 70 % del puntaje) en verde con badge.

La "Distribución por Universos" visualiza mediante tarjetas de colores la cantidad de postulantes en cada categoría: 4 Afrodescendientes (morado, 8 % cuota), 1 Trans (rosado, 1 % cuota), 4 Discapacidad (celeste, 4 % cuota), y las tarjetas incluyen badges indicando el porcentaje de cuota correspondiente.

El botón "Siguiente" permite avanzar al paso de previsualización donde el tribunal puede revisar el ordenamiento antes de confirmarlo y publicarlo definitivamente.

Consulta de Ordenamientos Generados

The screenshot shows a table titled "Ordenamientos del Llamado" with the following data:

ID	Tipo	Estado	Fecha de Generación	Acciones
37	Discapacidad	Preliminar	27/11/2025 16:44	<button>Ver Detalle</button>
36	Trans	Preliminar	27/11/2025 16:44	<button>Ver Detalle</button>
35	Afrodescendiente	Preliminar	27/11/2025 16:44	<button>Ver Detalle</button>
34	General	Publicado	27/11/2025 16:44	<button>Ver Detalle</button>
33	Discapacidad	Definitivo	23/11/2025 19:34	<button>Ver Detalle</button>
32	Trans	Publicado	23/11/2025 19:34	<button>Ver Detalle</button>
31	Afrodescendiente	Publicado	23/11/2025 19:34	<button>Ver Detalle</button>
30	General	Publicado	23/11/2025 19:34	<button>Ver Detalle</button>

Figura 8.9: Listado de ordenamientos generados por tipo y estado

La pantalla de ordenamientos (Figura 8.9) presenta una tabla con todos los ordenamientos generados para el llamado, organizados por ID, Tipo, Estado, Fecha de Generación y Acciones. El sistema genera automáticamente cuatro tipos de ordenamientos según los universos de cuotas: Discapacidad, Trans, Afrodescendiente, y General. Los ordenamientos pueden tener tres estados diferenciados visualmente: "Preliminar" para versiones en revisión, "Publicado" para versiones oficiales comunicadas a los postulantes, y "Definitivo" para la versión final certificada.

El botón "Volver al Dashboard" permite regresar a la vista principal, mientras que cada fila incluye un botón "Ver Detalle" que da acceso al ordenamiento completo con la lista de prelación, puntajes, y posiciones asignadas. Esta funcionalidad permite al tribunal gestionar múltiples versiones de ordenamientos, mantener un historial de generaciones, y controlar el flujo de publicación progresiva según avanza el proceso de evaluación.

Validación de Implementación

Las capturas presentadas demuestran que el MVP implementado cumple con los requisitos establecidos:

- Implementa los flujos completos especificados en los requisitos funcionales RF-01 a RF-15
- Mantiene coherencia visual con el prototipo validado en Framer durante la Fase 4
- Incorpora las reglas de negocio críticas: validaciones de formularios, cálculo automático de puntajes totales, y aplicación de cuotas de acción afirmativa
- Proporciona interfaces intuitivas alineadas con los principios de usabilidad identificados durante las entrevistas con stakeholders
- Refleja la identidad institucional de la DGC mediante uso consistente de colores corporativos (azul institucional), tipografía, y estructura de navegación

- Implementa indicadores visuales claros de estado mediante badges de colores, barras de progreso, y métricas en tiempo real
- Permite gestión de múltiples versiones de ordenamientos con control de estados (Preliminar, Publicado, Definitivo)

El sistema desarrollado valida la viabilidad técnica y funcional de digitalizar el proceso de llamados públicos, proporcionando una base sólida para evolucionar hacia un producto completo de producción. La implementación demuestra que las tecnologías seleccionadas (Angular 19, ASP.NET Core 8, SQL Server) son apropiadas para el dominio del problema y que la arquitectura en tres capas soporta efectivamente los requisitos funcionales y no funcionales del sistema.

8.4 Verificación Técnica del Sistema

Arquitectura y Diseño

La arquitectura implementada cumple con la separación de responsabilidades en tres capas bien definidas. La capa de presentación, desarrollada en Angular 19 [16], se organizó en módulos funcionales por dominio con componentes reutilizables, implementando routing con lazy loading para optimizar tiempos de carga, reactive forms para manejo robusto de formularios complejos, y guards de autenticación para proteger rutas.

La capa de lógica de negocio, implementada en ASP.NET Core 8 [23], expone una API RESTful [18] con sus respectivos endpoints documentados, organizados en 6 controllers que encapsulan la lógica de dominio. Se implementaron 8 servicios de lógica de negocio utilizando inyección de dependencias (DI [39]) de .NET, manejo de transacciones mediante Unit of Work, y DTOs (Data Transfer Objects) [38] para separar el modelo de dominio de los contratos de API [18].

La capa de acceso a datos, construida con Entity Framework Core 8 [24] sobre SQL Server [20], implementa el patrón Repository [38] para cada agregado del dominio, coordinando

transacciones mediante Unit of Work [38] y utilizando migraciones versionadas para gestionar la evolución del esquema de base de datos. Las especificaciones técnicas completas de la arquitectura se documentan en el Apéndice I.

Esta arquitectura demostró ser robusta y mantenible durante el desarrollo, facilitando el trabajo paralelo en diferentes componentes y permitiendo realizar cambios en una capa sin afectar las demás.

API REST y Documentación con Swagger

La capa de lógica de negocio expone su funcionalidad mediante una API RESTful documentada automáticamente con Swagger/OpenAPI 3.0. Esta documentación interactiva permite a desarrolladores explorar todos los endpoints disponibles, consultar esquemas de DTOs, y probar operaciones directamente desde el navegador.

El sistema implementa 6 controladores que encapsulan 37 endpoints organizados por dominio funcional. Cada endpoint sigue convenciones REST estándar, utilizando verbos HTTP semánticos (GET para consultas, POST para creaciones, PUT para actualizaciones) y retornando códigos de estado apropiados (200 OK, 201 Created, 400 Bad Request, 404 Not Found, 500 Internal Server Error).

TribunalController: Evaluación y Ordenamientos

The screenshot shows the Swagger UI interface for the TribunalController. At the top, there is a brief description: "Tribunal Controlador del tribunal evaluador (RF-11, RF-12, RF-14, RF-15). Gestiona inscripciones para evaluar, calificaciones de pruebas, méritos y ordenamientos." Below this, a list of 15 API endpoints is displayed, each with its HTTP method, URL path, and a brief description:

- GET /api/Tribunal/llamado/{llamadoId}/inscripciones: Obtiene las inscripciones de un llamado con su estado de evaluación (RF-11).
- GET /api/Tribunal/inscripcion/{inscripcionId}/detalles: Devuelve el detalle completo de evaluaciones, requisitos y méritos de una inscripción.
- GET /api/Tribunal/llamado/{llamadoId}/estadisticas: Obtiene estadísticas agregadas del llamado para seguimiento del tribunal.
- GET /api/Tribunal/llamado/{llamadoId}/pruebas: Lista las pruebas del llamado con información de evaluaciones registradas.
- POST /api/Tribunal/calificar-prueba: Registra la calificación de una prueba rendida por una inscripción (RF-12).
- POST /api/Tribunal/valorar-merito: Valoración individual de un mérito incluyendo verificación de documentación (RF-14).
- POST /api/Tribunal/inscripcion/{inscripcionId}/valorar-meritos: Valoración masiva de los méritos asociados a una inscripción.
- POST /api/Tribunal/generar-ordenamiento: Genera ordenamientos preliminares o definitivos, aplicando cuotas y desempates (RF-15).
- GET /api/Tribunal/llamado/{llamadoId}/ordenamientos: Obtiene los ordenamientos generados para el llamado especificado.
- GET /api/Tribunal/ordenamiento/{ordenamientoId}: Devuelve el detalle (posiciones) de un ordenamiento específico.
- POST /api/Tribunal/ordenamiento/{ordenamientoId}/publicar: Publica un ordenamiento para dejarlo disponible externamente.

Figura 8.10: Endpoints del controlador Tribunal en Swagger UI

El TribunalController (Figura 8.10) implementa 11 endpoints que cubren el flujo completo de evaluación (RF-11, RF-12, RF-14, RF-15). Los endpoints de consulta (GET) permiten obtener inscripciones de un llamado con su estado de evaluación, recuperar el detalle completo de evaluaciones incluyendo requisitos y méritos, consultar estadísticas agregadas del llamado para seguimiento del tribunal, listar las pruebas del llamado con información de evaluaciones registradas, obtener los ordenamientos generados para el llamado especificado, y devolver el detalle con posiciones de un ordenamiento específico.

Los endpoints de modificación (POST) permiten registrar la calificación de una prueba rendida por una inscripción (RF-12), valorar individualmente un mérito incluyendo verificación de documentación (RF-14), realizar valoración masiva de los méritos asociados a una inscripción, generar ordenamientos preliminares o definitivos aplicando cuotas y desempates (RF-15), y publicar un ordenamiento para dejarlo disponible externamente.

PostulanteController: Gestión del Perfil

The screenshot shows the Swagger UI interface for the PostulanteController. At the top, there is a brief description: "Postulante Controlador para operaciones de postulantes (RF-01, RF-02, RF-20). Expone endpoints para consultar y actualizar datos personales, además de validar cédula/email." Below this, there are four API endpoints listed:

- GET /api/Postulante/{id}**: Obtiene los datos del postulante y el indicador de completitud de perfil (RF-01).
- PUT /api/Postulante/{id}/datos-personales**: Actualiza la información personal del postulante aplicando validaciones de negocio (RF-02).
- GET /api/Postulante/validar-cedula/{cedula}**: Valida si una cédula está disponible para registrar (RF-20).
- GET /api/Postulante/validar-email/{email}**: Verifica si el email ya está registrado por otro postulante.

Figura 8.11: Endpoints del controlador Postulante en Swagger UI

El PostulanteController (Figura 8.11) expone 4 endpoints para gestión de datos personales (RF-01, RF-02, RF-20). El endpoint GET permite obtener los datos del postulante y el indicador de completitud de perfil (RF-01). El endpoint PUT actualiza la información personal del postulante aplicando validaciones de negocio como formato de cédula, validez de email, y edad mínima requerida (RF-02). Los dos endpoints de validación permiten verificar si una cédula está disponible para registrar (RF-20) y verificar si el email ya está registrado por otro postulante, evitando duplicados en el sistema.

LlamadoController: Consulta de Llamados

The screenshot shows the Swagger UI interface for the LlamadoController. At the top, there is a section titled "Llamado" which describes it as a controller for public calls (RF-03 and RF-04) that allows querying lists, details, and components of a call. Below this, there is a list of seven endpoints, each with a "GET" method and its corresponding URL:

- GET /api/Llamado/{id} Obtiene el detalle completo de un llamado (requisitos, ítems puntuables, apoyos).
- GET /api/Llamado/activos Lista los llamados vigentes disponibles para inscripción (RF-03).
- GET /api/Llamado/inactivos Devuelve los llamados finalizados o cerrados para tareas administrativas.
- GET /api/Llamado/{id}/validar-disponible Valida si el llamado está abierto para nuevas inscripciones.
- GET /api/Llamado/{id}/requisitos Recupera la lista de requisitos excluyentes del llamado.
- GET /api/Llamado/{id}/items-puntuables Obtiene los ítems puntuables que utilizará el tribunal para evaluar méritos.
- GET /api/Llamado/{id}/apoyos Lista los apoyos necesarios ofrecidos en el llamado (accesibilidad).

Figura 8.12: Endpoints del controlador Llamado en Swagger UI

El LlamadoController (Figura 8.12) implementa 7 endpoints exclusivamente de consulta (RF-03, RF-04). El primer endpoint obtiene el detalle completo de un llamado incluyendo requisitos, ítems puntuables y apoyos necesarios. El segundo lista los llamados vigentes disponibles para inscripción (RF-03), mientras que el tercero devuelve los llamados finalizados o cerrados para tareas administrativas. El cuarto endpoint valida si el llamado está abierto para nuevas inscripciones antes de permitir el flujo de postulación. Los tres últimos endpoints recuperan componentes específicos del llamado: la lista de requisitos excluyentes del llamado, los ítems puntuables que utilizará el tribunal para evaluar méritos, y los apoyos necesarios ofrecidos en el llamado para accesibilidad.

InscripcionController: Gestión de Inscripciones

The screenshot shows the Swagger UI interface for the InscripcionController. At the top, there is a section titled "Inscripción" which describes it as a controller for managing applications from candidates (RF-05, RF-07, RF-08) that allows creating, querying, and validating applications along with their requirements and scores. Below this, there is a list of seven endpoints, each with a "POST" or "GET" method and its corresponding URL:

- POST /api/Inscripcion/postulante/{postulanteId} Crea una nueva inscripción completa para un postulante dado (RF-05).
- GET /api/Inscripcion/postulante/{postulanteId} Lista las inscripciones realizadas por un postulante (RF-07).
- GET /api/Inscripcion/{id} Obtiene el detalle completo de una inscripción (RF-08).
- GET /api/Inscripcion/validar/{postulanteId}/{llamadoId} Valida si ya existe una inscripción del postulante en el llamado indicado.
- GET /api/Inscripcion/{id}/validar-requisitos Valida el cumplimiento de requisitos obligatorios de una inscripción.
- POST /api/Inscripcion/{id}/calcular-puntaje Calcula y actualiza el puntaje total de la inscripción combinando pruebas y méritos.

Figura 8.13: Endpoints del controlador Inscripción en Swagger UI

El InscripcionController (Figura 8.13) gestiona el ciclo de vida de inscripciones mediante 6 endpoints (RF-05, RF-07, RF-08). El endpoint POST crea una nueva inscripción completa para un postulante dado, ejecutando validaciones de requisitos y duplicidad (RF-05). Los endpoints GET permiten listar las inscripciones realizadas por un postulante (RF-07), obtener el detalle completo de una inscripción específica mostrando todos los datos presentados (RF-08), y validar si ya existe una inscripción del postulante en el llamado indicado para prevenir duplicados. El endpoint de validación de requisitos comprueba el cumplimiento de requisitos obligatorios de una inscripción según las reglas del llamado. Finalmente, el endpoint POST de cálculo de puntaje calcula y actualiza el puntaje total de la inscripción combinando pruebas y méritos, ejecutando la lógica de sumatoria implementada en el servicio.

DepartamentoController: Catálogo de Departamentos

Departamento Endpoints para consultar departamentos disponibles en llamados (RF-03).

- GET `/api/Departamento` Obtiene todos los departamentos activos del sistema.
- GET `/api/Departamento/{id}` Recupera un departamento específico por su identificador.
- GET `/api/Departamento/llamado/{llamadoId}` Lista los departamentos habilitados para un llamado determinado.
- GET `/api/Departamento/validar/{departamentoId}/llamado/{llamadoId}` Valida si el departamento forma parte del llamado antes de permitir la inscripción.

Figura 8.14: Endpoints del controlador Departamento en Swagger UI

El DepartamentoController (Figura 8.14) expone 4 endpoints de consulta para datos maestros (RF-03). El primer endpoint obtiene todos los departamentos activos del sistema para poblar dropdowns de selección. El segundo recupera un departamento específico por su identificador único. El tercer endpoint lista los departamentos habilitados para un llamado determinado, mostrando cantidad de puestos disponibles por departamento. El cuarto endpoint valida si el departamento forma parte del llamado antes de permitir la inscripción, verificando que existan cupos disponibles en ese departamento.

ConstanciaController: Gestión Documental

The screenshot shows the Swagger UI interface for the ConstanciaController. At the top, there is a section titled "Constancia" with the subtitle "Endpoints para gestión de constancias/documentos del postulante (RF-06)". Below this, five API endpoints are listed in a table:

Method	Endpoint	Description
POST	/api/Constancia/postulante/{postulanteId}	Sube y registra una constancia en PDF o imagen para un postulante.
GET	/api/Constancia/postulante/{postulanteId}	Obtiene el listado de constancias cargadas por el postulante.
GET	/api/Constancia/{id}	Recupera los metadatos de una constancia específica.
PUT	/api/Constancia/{id}/validar	Marca una constancia como validada por el equipo administrativo.
GET	/api/Constancia/{id}/descargar	Descarga el archivo asociado a una constancia existente.

Figura 8.15: Endpoints del controlador Constancia en Swagger UI

El ConstanciaController (Figura 8.15) gestiona documentos adjuntos mediante 5 endpoints (RF-06). El endpoint POST sube y registra una constancia en PDF o imagen para un postulante, validando tipo de archivo y tamaño máximo permitido. Los endpoints GET permiten obtener el listado de constancias cargadas por el postulante mostrando metadatos de cada archivo, recuperar los metadatos de una constancia específica incluyendo nombre original y fecha de carga, y descargar el archivo asociado a una constancia existente retornando el contenido binario con el media type apropiado. El endpoint PUT marca una constancia como validada por el equipo administrativo, cambiando su estado de "Pendiente."^a "Verificada" tras la revisión manual.

Verificación de Interoperabilidad

La implementación de la API REST con documentación Swagger cumple con el requisito no funcional RNF-IN-01 de interoperabilidad. El sistema expone 37 endpoints organizados en 6 controladores que siguen convenciones REST estándar, utilizando JSON como formato de intercambio de datos en todas las peticiones y respuestas, códigos de estado HTTP semánticos para comunicar el resultado de cada operación, y documentación OpenAPI 3.0 que permite integración automatizada con herramientas de terceros.

La API fue verificada utilizando Postman durante el desarrollo, verificando que todos los endpoints respondan correctamente a peticiones válidas e inválidas, que los DTOs se serialicen y deserialicen apropiadamente, que las validaciones del lado del servidor funcionen según lo

especificado, y que los códigos de estado HTTP reflejen correctamente el resultado de cada operación (200 para éxito, 201 para creación, 400 para datos inválidos, 404 para recursos no encontrados, 500 para errores internos).

La documentación de Swagger permite a desarrolladores externos comprender la estructura de la API sin necesidad de consultar código fuente, facilita la integración con otros sistemas que requieran consumir datos del portal, y proporciona una herramienta de testing manual accesible desde el navegador que acelera la validación de cambios durante el desarrollo.

Pruebas Automatizadas y Cobertura

Se implementó una estrategia de testing en pirámide, priorizando pruebas unitarias en la base y complementando con validaciones manuales de flujos completos. Para el backend se utilizó xUnit [69] con Moq para mocking, desarrollando un total de 156 tests unitarios que cubren los servicios críticos del sistema. Los servicios con mayor cobertura incluyen TribunalService (34 tests, 89 % cobertura), InscripcionService (28 tests, 87 % cobertura), ValidacionService (26 tests, 91 % cobertura), PostulanteService (22 tests, 85 % cobertura) y LlamadoService (18 tests, 82 % cobertura), alcanzando una cobertura promedio del 87 % en servicios críticos.

Los tests implementados cubren tres categorías principales: validación de reglas de negocio (verificando que las restricciones funcionen correctamente), casos exitosos (validando el flujo completo cuando los datos son correctos), y casos de error (verificando el manejo apropiado de situaciones excepcionales como entidad no encontrada, datos inválidos, o violación de constraints). También se implementaron tests de transacciones para verificar que operaciones que modifican entidades mantengan consistencia mediante commit/rollback. El detalle completo de cobertura por servicio se encuentra en el Apéndice J.

Se configuró protección de ramas en el repositorio Git, requiriendo que las ramas principales (main y develop) solo puedan actualizarse mediante Pull Requests aprobados. Esta práctica establece un punto de control de calidad antes de integrar cambios, permitiendo revisión del código y verificación manual de que los tests pasen correctamente.

Verificación de Requisitos No Funcionales

La Verificación de requisitos no funcionales se abordó de manera diferenciada según su naturaleza. Para requisitos relacionados con estructura y diseño del sistema (como arquitectura en capas, separación de responsabilidades, y patrones de diseño), la Verificación se realizó mediante revisión arquitectónica y chequear que el código implementado siga los principios SOLID [114] y las convenciones establecidas.

En cuanto a rendimiento, el sistema fue diseñado con las mejores prácticas para optimización (lazy loading, índices en base de datos, carga asíncrona de componentes), pero las mediciones cuantitativas de tiempos de respuesta bajo carga real quedan como trabajo futuro previo a producción. De manera similar, los requisitos de escalabilidad (soporte de 500 usuarios concurrentes según RNF-SC-01) y seguridad (HTTPS, encriptación AES-256) requieren validación en un entorno que simule condiciones de producción.

Los requisitos de usabilidad se validaron parcialmente mediante las pruebas con usuarios del prototipo, confirmando que el flujo de inscripción en 6 pasos resulta claro y navegable. En accesibilidad, se implementaron las bases técnicas necesarias (contraste de color 4.5:1, etiquetas semánticas HTML5, atributos ARIA, navegación por teclado), aunque una auditoría completa con herramientas especializadas y usuarios con discapacidad queda pendiente para garantizar cumplimiento total con WCAG 2.1 nivel AA [28].

El POC cumple con el requisito de interoperabilidad (RNF-IN-01), exponiendo una API REST [18] funcional con 37 endpoints documentados, formato JSON [41] para todas las peticiones y respuestas, códigos de estado HTTP [21] semánticos, y CORS configurado para permitir integración con frontends externos. Esta API fue validada utilizando Postman [79], verificando que todos los endpoints respondan con JSON válido y bien formado.

Los requisitos de mantenibilidad se cumplen mediante código que sigue principios SOLID [114], arquitectura en capas bien definida, cobertura de tests del 87 % en servicios críticos (superando el objetivo de 85 %), y documentación técnica completa que incluye ESRE, casos de

uso, diagramas arquitectónicos, y XML comments en código C# para todas las clases públicas.

El Apéndice K documenta el plan detallado de validación de RNF que deberá ejecutarse previo al despliegue en producción, incluyendo pruebas de carga, auditorías de seguridad, y testing exhaustivo de accesibilidad.

8.5 Validación con Usuarios Reales

Prototipo de Alta Fidelidad

Durante la fase de diseño, se creó un prototipo interactivo en Framer [62] y se validó con usuarios representativos. Las sesiones de validación utilizaron el protocolo thinking aloud de Nielsen [63], en el que los usuarios verbalizaban sus impresiones mientras interactuaban con el prototipo. Los participantes incluían postulantes que ya habían atravesado procesos de inscripción anteriores, integrantes del tribunal con experiencia en evaluación de méritos y pruebas, y personal de RR. HH. encargado de procesar inscripciones y administrar llamados, lo que permitió obtener retroalimentación con conocimiento directo del procedimiento actual.

Los usuarios encontraron el flujo de inscripción claro y lógico, y valoraron positivamente la generación automática del código de confirmación. Los evaluadores consideraron útiles las métricas en tiempo real del dashboard y los filtros dinámicos de la interfaz de evaluación. Todos los usuarios completaron con éxito la inscripción simulada.

Las sesiones identificaron algunas oportunidades de mejora que fueron incorporadas al desarrollo, incremento del contraste de color para cumplir el ratio WCAG 4.5:1 [28], adición de un indicador visual de progreso más prominente en el workflow de inscripción y reorganización de la vista de detalle del postulante para priorizar méritos sobre datos generales.

Sistema Implementado

El MVP implementado fue validado con usuarios que participaron en las entrevistas iniciales, incluido un especialista en Quality Assurance (QA) que realizó dos contribuciones fundamentales para el aseguramiento de calidad del proyecto futuro:

1. **Plan de Pruebas Funcionales del MVP** (Ver Apéndice M): Documento de 20 casos de prueba diseñados para validar los flujos funcionales implementados en el MVP. Este plan fue ejecutado durante la fase final del proyecto, obteniendo resultados que confirman la funcionalidad de los módulos implementados e identificando áreas de mejora.
2. **Plan de Validación de RNF Pre-Producción** (Ver Apéndice K): Plan de validación de requisitos no funcionales que deberá ejecutarse previo al despliegue en producción, incluyendo pruebas de carga, auditorías de seguridad, y testing de accesibilidad con usuarios reales.

Los 20 casos de prueba fueron ejecutados, obteniendo 11 casos exitosos, 8 pendientes por módulos no implementados en el alcance del MVP, y 1 defecto identificado. Esta distribución es coherente con el objetivo de completar aproximadamente el 70 % de la funcionalidad total del sistema, priorizando el flujo completo del postulante y los componentes críticos de evaluación del tribunal.

Casos de prueba que pasaron exitosamente (11):

Flujos de Postulante (7 casos):

- Login de usuarios registrados con redirección correcta al panel de llamados
- Registro de postulantes con validaciones de CI, email único y generación de credenciales
- Visualización de listado de llamados vigentes con filtros funcionales (activo/inactivo)
- Acceso a detalle completo de llamado incluyendo requisitos, cuotas y cronograma

- Workflow de inscripción de 6 pasos con guardado y validaciones por paso
- Carga de documentos
- Declaración de pertenencia a cuotas de acción afirmativa con visualización correcta

Flujos de Tribunal (4 casos):

- Evaluación y puntuación de méritos declarados por postulantes según ítems puntuables
- Calificación de pruebas rendidas (escritas, orales, prácticas) con asignación de notas
- Generación de lista final ordenada por puntaje total aplicando cuotas legales
- Visualización de dashboard con progreso de evaluaciones y estadísticas del llamado

Estos 11 casos validan el flujo del actor Postulante (RF-01 a RF-08) y componentes del actor Tribunal (RF-11, RF-12, RF-14), demostrando que la arquitectura implementada soporta tanto la interacción del usuario final como la lógica compleja de evaluación y ordenamiento con cuotas.

Casos de prueba pendientes por módulos no implementados (8):

- Creación y publicación de llamados desde backoffice (rol RRHH)
- Confirmación de postulación vía correo electrónico con identificador único
- Sistema de notificaciones automáticas por email/WhatsApp
- Preselección automática basada en requisitos excluyentes
- Generación de ordenamiento con semilla criptográfica
- Gestión de usuarios del backoffice con roles y permisos
- Configuración de pruebas por parte de RRHH (tipo, ponderación, fechas)
- Auditoría completa con exportación de logs firmados digitalmente

Estos módulos, aunque diseñados arquitectónicamente y con lógica de negocio implementada en los servicios correspondientes (TribunalService, LlamadoService con 82-89 % de cobertura de tests unitarios), no cuentan con interfaces de usuario completamente funcionales o requieren integraciones con servicios externos (SMTP, WhatsApp API) que exceden el alcance del MVP. Su implementación está prevista como trabajo futuro una vez validado el concepto core con usuarios reales.

Defecto identificado (1 caso):

La ejecución del plan identificó un defecto de sincronización en el flujo de publicación de resultados. Cuando el tribunal marca los resultados como publicados, la actualización no se refleja correctamente en el estado visible en el panel del postulante. Este defecto no afecta la funcionalidad core del MVP (inscripción y evaluación) pero debe corregirse antes de despliegue en producción. El problema se identificó como una falta de sincronización entre el módulo de gestión del tribunal y la vista pública de consulta de resultados.

Mejoras de Usabilidad Identificadas

El especialista QA identificó oportunidades de mejora en la experiencia de usuario del workflow de inscripción que, si bien no afectan la funcionalidad, podrían incrementar la satisfacción del usuario en versiones futuras:

- **Navegación sticky:** Mantener el indicador de progreso del workflow visible durante el desplazamiento vertical (scroll), permitiendo al usuario saber en todo momento en qué paso se encuentra sin necesidad de volver arriba. Esta mejora es particularmente relevante en dispositivos móviles donde el espacio vertical es limitado.
- **Preservación de contexto:** Asegurar que al navegar entre pasos, el sistema preserve claramente la posición del usuario y permita retroceder sin perder información ya completada. Incluye indicadores visuales más prominentes de pasos completados versus pendientes.

- **Flujo lineal optimizado:** Para iteraciones futuras, explorar la posibilidad de condensar el workflow en una única vista de desplazamiento vertical continuo, reduciendo las transiciones entre pantallas y permitiendo una experiencia más fluida similar a formularios modernos de aplicaciones SaaS.
- **Feedback visual mejorado:** Añadir animaciones sutiles y mensajes de confirmación más claros cuando el usuario completa un paso exitosamente, reforzando la sensación de progreso y reduciendo la incertidumbre durante el proceso de inscripción.

Estas sugerencias fueron documentadas como mejoras incrementales de UX y no como defectos bloqueantes, reconociendo que el diseño actual cumple con los requisitos de usabilidad establecidos (RNF-UB-01: Flujo de inscripción en máximo 6 pasos) pero puede optimizarse para reducir la carga cognitiva del usuario y mejorar la percepción de fluidez del sistema.

Conclusiones de la Validación

Los revisores confirmaron que el sistema implementado refleja las necesidades identificadas durante el relevamiento con stakeholders de la DGC. El flujo principal de postulación, que representa el caso de uso más crítico y frecuente del sistema, funciona correctamente. Adicionalmente, la validación exitosa de los casos de evaluación de tribunal demuestra que la lógica de calificación, ordenamiento y aplicación de cuotas opera correctamente, validando los algoritmos centrales del sistema.

La identificación del defecto de sincronización en la publicación de resultados durante las pruebas funcionales demuestra la efectividad del proceso de validación, permitiendo detectar y documentar problemas antes de una eventual presentación a usuarios reales. La corrección de este defecto está priorizada para realizarse previo a cualquier piloto con la DGC.

La arquitectura implementada proporciona una base para extender, con el objetivo de incorporar los módulos pendientes (gestión de llamados por RRHH, notificaciones automáticas, auditoría completa) sin requerir refactorizaciones estructurales significativas. La separación clara

entre capas (Domain, DataAccess, BusinessLogic, WebApi) permite agregar nuevas funcionalidades de manera incremental y predecible.

La combinación de validación funcional mediante el plan de pruebas (cobertura end-to-end de 11 flujos críticos de usuario) y validación mediante tests automatizados (192 tests unitarios e integración con 87 % de cobertura) proporciona confianza en la calidad y estabilidad del MVP entregado.

8.6 Cumplimiento Normativo

El sistema fue diseñado considerando los principios de la Ley 18.331 de Protección de Datos Personales [3], solicitando únicamente los datos estrictamente necesarios para el proceso de selección. Los datos sensibles (autodefinición étnico-racial, identidad de género, discapacidad) son opcionales y se solicitan para cumplir con las cuotas de acción afirmativa establecidas por ley [4].

En cuanto al Decreto 406/022 [27] sobre Accesibilidad en Sitios Web Gubernamentales, el portal implementa las bases técnicas para cumplir WCAG 2.1 nivel AA [28], incluyendo contraste de color adecuado, etiquetas semánticas HTML5, navegación por teclado funcional, y atributos ARIA [57] en componentes interactivos. Una auditoría completa de accesibilidad queda pendiente para validar el cumplimiento total del decreto antes de la producción.

El sistema automatiza la aplicación de las tres leyes de cuotas de acción afirmativa: Ley 19.122 [7] (8 % afrodescendientes), Ley 19.684 [8] (1 % trans), y Ley 18.651 [9] (4 % discapacidad). El algoritmo de ordenamiento reserva automáticamente las posiciones correspondientes según estos porcentajes, eliminando el riesgo de errores de cálculo manuales y garantizando el cumplimiento de la normativa obligatoria.

8.7 Limitaciones y Trabajo a Futuro

Si bien el MVP valida los flujos principales, existen funcionalidades que deben implementarse antes de un despliegue en producción. Las más críticas incluyen autenticación JWT con refresh tokens [60], sistema RBAC de roles y permisos [115], HTTPS con certificado institucional .gub.uy [5], y encriptación de datos sensibles en reposo [120]. En infraestructura, se requieren almacenamiento de archivos en servicios de nube escalables, sistemas de respaldo automatizados, monitoreo de disponibilidad y rendimiento, y un plan de recuperación ante desastres.

Como trabajo a futuro, se implementarían las funcionalidades adicionales de alta prioridad, incluyendo sistema de notificaciones por email y WhatsApp, módulo RR. HH. para creación y gestión de llamados, y auditoría completa de accesibilidad con usuarios reales con discapacidad. También quedan pendientes pruebas adicionales, como pruebas de carga con usuarios concurrentes para validar escalabilidad, auditoría de seguridad, pruebas end-to-end automatizadas con frameworks y validación exhaustiva con tecnologías asistivas.

El detalle completo de funcionalidades pendientes y su priorización se encuentra en el Apéndice [L](#).

8.8 Resultados

El desarrollo del Portal DGC como Prueba de concepto alcanzó su objetivo, demostrando la viabilidad técnica de la solución propuesta. Los dos workflows críticos (inscripción y evaluación) funcionan correctamente, lo que valida que la arquitectura en tres capas es sólida y que las tecnologías seleccionadas son apropiadas para el dominio del problema.

El sistema representa una mejora significativa respecto al proceso manual actual. Reduce el tiempo de inscripción, elimina la duplicación de tareas (el postulante completa el formulario directamente sin que RR. HH. deba redigitar), proporciona confirmación inmediata en lugar de esperar 72 horas, automatiza completamente el cálculo de puntajes evitando errores manuales y automatiza la aplicación de cuotas garantizando cumplimiento legal

En términos cuantitativos, se implementaron 14 de 21 requisitos funcionales (67 %) y se validaron 9 de 16 requisitos no funcionales (56 %), con los requisitos pendientes correspondiendo mayormente a infraestructura y no a funcionalidad core. La arquitectura en capas está bien definida con separación clara de responsabilidades, el código alcanza 87 % de cobertura de tests en servicios críticos, y la documentación técnica está completa incluyendo ESRE, casos de uso, diagramas arquitectónicos, y comentarios inline en todo el código público.

Los usuarios involucrados en las pruebas del prototipo expresaron satisfacción con el diseño propuesto y confirmaron que el sistema aborda los puntos de dolor del proceso actual. El sistema está diseñado desde el inicio para cumplir con el marco legal uruguayo (protección de datos, accesibilidad, cuotas de acción afirmativa), aunque requiere ajustes finales antes de producción.

El MVP desarrollado constituye una base sobre la cual construir las funcionalidades pendientes para llegar a un sistema listo para producción. La arquitectura elegida, las tecnologías seleccionadas y las prácticas de desarrollo aplicadas posicionan al proyecto para una evolución hacia una plataforma completa que transforme digitalmente el proceso de llamados públicos de la Dirección General de Casinos.

Lecciones Aprendidas

Este proyecto fue mi primera experiencia en el desarrollo de software fuera del ámbito académico. Hasta ahora, mi vínculo con los sistemas había sido desde roles de gestión y operativa, por lo que afrontar el desarrollo de una solución real implicó un cambio en mi forma de trabajar y de entender el proceso. En la carrera había visto metodologías y modelos, pero aplicarlos en un proyecto propio me mostró por qué son necesarios. Dedicar tiempo a definir con claridad los requerimientos me permitió tomar buenas decisiones y evitar suposiciones. Entendí que escribir código sin un objetivo claro no es avanzar, sino acumular problemas para el futuro.

Trabajar solo implicó asumir roles que, en un entorno real, estarían distribuidos (análisis, diseño, desarrollo, etc.). Fue un desafío, pero me obligó a investigar, a justificar mis decisiones y a mantener una disciplina de trabajo sostenida. Aprendí que documentar para uno mismo es tan importante como documentar para los demás.

Diseñar una arquitectura que funcionara para este dominio específico fue algo distinto. Cada decisión impactaba en la mantenibilidad y en la posibilidad de extender el sistema. Al tratarse de un MVP, fue necesario pensar desde el inicio en cómo escalar la solución y no solo en resolver lo inmediato. Ver cómo una arquitectura sólida facilitaba el desarrollo de los workflows críticos confirmó la importancia de esta etapa.

Aprendí a detenerme, evaluar y ajustar cuando era necesario. Saber no es solo escribir código, sino validar que ese avance tiene sentido dentro del sistema y que no compromete lo ya logrado. Esta experiencia me permitió transformar lo aprendido en la carrera en algo práctico y aplicable. Los conceptos comenzaron a tener más sentido al aplicarlos en un proyecto real, donde cada decisión tenía un impacto concreto y el aprendizaje surgía de la práctica.

Más allá del resultado técnico, me llevo una gran evolución profesional, una comprensión más completa del proceso de desarrollo y una base sólida para seguir creciendo en este rol.

Conclusiones

Se desarrolló una prueba de concepto (MVP) de un portal web para digitalizar el proceso de llamados públicos de la Dirección General de Casinos. El MVP implementa los dos workflows críticos del sistema (inscripción del postulante y evaluación del tribunal), con la generación automática de ordenamientos y la aplicación de las cuotas legales de acción afirmativa establecidas en la normativa vigente.

El propósito fue abordar las ineficiencias de un proceso totalmente manual que genera demoras en la confirmación de inscripciones, duplicación de tareas, errores de cálculo manual en los puntajes y riesgo de incumplimiento normativo en la aplicación de las cuotas de acción afirmativa. La prueba de concepto debía demostrar la viabilidad técnica y funcional de automatizar estas operaciones críticas, garantizando trazabilidad y transparencia.

Se aplicó una metodología ágil con entrevistas a interesados, especificación de requisitos, diseño de una arquitectura en tres capas, creación de un prototipo validado con usuarios y desarrollo del MVP acompañado de pruebas automatizadas.

El sistema busca reducir los tiempos de inscripción mediante confirmación inmediata, apunta a disminuir la duplicación de tareas y propone automatizar el cálculo de puntajes para minimizar errores manuales, además de favorecer la aplicación correcta de las cuotas de acción afirmativa. La retroalimentación de los usuarios que participaron en las pruebas del prototipo indica que el enfoque desarrollado aborda los principales puntos de dolor del proceso actual.

El MVP constituye el punto de partida para avanzar hacia un sistema de producción. Su arquitectura y documentación permiten incorporar las funcionalidades restantes y los resultados obtenidos muestran que la digitalización del proceso de llamados públicos es viable. Con esta solución, la DGC tendría la posibilidad de modernizar un procedimiento actualmente manual y lento, transformándolo en un proceso más ágil, trazable y preciso.

Bibliografía

- [1] Dirección General de Casinos (DGC), “Dirección General de Casinos,” Ministerio de Economía y Finanzas - Uruguay, 2025, [En línea]. Disponible en: <https://www.gub.uy/ministerio-economia-finanzas/direccion-general-casinos> [Consultado: Nov. 2025].
- [2] Ministerio de Economía y Finanzas (MEF), “Sitio web oficial,” Ministerio de Economía y Finanzas - Uruguay, 2025, [En línea]. Disponible en: <https://www.gub.uy/ministerio-economia-finanzas/> [Consultado: Nov. 2025].
- [3] Uruguay, “Ley N° 18.331. Protección de Datos Personales y Acción de Habeas Data,” IMPO - Base de datos de Leyes, 2008, [En línea]. Disponible en: <https://www.impo.com.uy/bases/leyes/18331-2008> [Consultado: Nov. 2025].
- [4] ——, “Ley N° 19.584. APROBACIÓN DE LA CONVENCIÓN INTERAMERICANA CONTRA TODA FORMA DE DISCRIMINACIÓN E INTOLERANCIA,” IMPO - Base de datos de Leyes Originales, Ene 2018, promulgada: 28/12/2017. [En línea]. Disponible en: <https://www.impo.com.uy/bases/leyes-originiales/19584-2017> [Consultado: Nov. 2025].
- [5] Gobierno de Uruguay, “Portal del Estado Uruguayo,” gub.uy, [En línea]. Disponible en: <https://www.gub.uy/> [Consultado: Nov. 2025].
- [6] Dirección General de Casinos (DGC), “Convocatorias y llamados (Temática: Dirección General de Casinos),” Ministerio de Economía y Finanzas - Uruguay, 2025, [En línea]. Disponible en: https://www.gub.uy/ministerio-economia-finanzas/comunicacion/convocatorias?field_vigencia=0&field_tematica_gubuy=1402&year=all&month=all [Consultado: Nov. 2025].
- [7] Uruguay, “Ley N° 19.122. Díctanse normas con el fin de favorecer la participación en las áreas educativa y laboral, de los afrodescendientes,” IMPO - Base de datos de Leyes, Sep 2013, promulgada: 21/08/2013. [En línea]. Disponible en: <https://www.impo.com.uy/bases/leyes/19122-2013> [Consultado: Nov. 2025].

- [8] ——, “Ley N° 19.684. LEY INTEGRAL PARA PERSONAS TRANS,” IMPO - Base de datos de Leyes, Nov 2018, promulgada: 26/10/2018. [En línea]. Disponible en: <https://www.impo.com.uy/bases/leyes/19684-2018> [Consultado: Nov. 2025].
- [9] ——, “Ley N° 18.651. PROTECCIÓN INTEGRAL A PERSONAS CON DISCAPACIDAD,” IMPO - Base de datos de Leyes, Mar 2010, promulgada: 19/02/2010. [En línea]. Disponible en: <https://www.impo.com.uy/bases/leyes/18651-2010> [Consultado: Nov. 2025].
- [10] Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento (AGESIC), “Portal institucional,” Sitio oficial del Gobierno de Uruguay - GUB.UY, [En línea]. Disponible en: <https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/> [Consultado: Nov. 2025].
- [11] Ministerio de Economía y Finanzas (MEF), “Llamado a concurso público y abierto de oposición y méritos hasta veinte (20) cargos...” Portal de la Presidencia - <https://www.gub.uy/ministerio-economia-finanzas/>, [En línea]. Disponible en: <https://www.gub.uy/ministerio-economia-finanzas/comunicacion/convocatorias/llamado-concurso-publico-aberto-oposicion-meritos-hasta-veinte-20> [Consultado: Nov. 2025].
- [12] (2025) Banco de previsión social (bps). Banco de Previsión Social (BPS). Sitio oficial del organismo de seguridad social de Uruguay. [Online]. Available: <https://www.bps.gub.uy/>
- [13] Banco de Seguros del Estado (BSE), “Sitio web oficial,” Banco de Seguros del Estado, [En línea]. Disponible en: <https://www.bse.com.uy/> [Consultado: Nov. 2025].
- [14] Oficina Nacional del Servicio Civil (ONSC), “Uruguay Concursa,” Portal de la Oficina Nacional del Servicio Civil (ONSC), [En línea]. Disponible en: <https://www.uruguayconcursa.gub.uy/> [Consultado: Nov. 2025].

- [15] ——, “Sitio web oficial,” Sitio oficial del Gobierno de Uruguay - GUB.UY, [En línea]. Disponible en: <https://www.gub.uy/oficina-nacional-servicio-civil/> [Consultado: Nov. 2025].
- [16] Google, “Angular Documentation,” angular.io, [En línea]. Disponible en: <https://angular.dev/overview> [Consultado: Nov. 2025].
- [17] Meta, “React: A JavaScript library for building user interfaces,” Sitio web oficial de React, [En línea]. Disponible en: <https://react.dev/> [Consultado: Nov. 2025].
- [18] Oracle, “What is a RESTful API?” Documentación técnica de Oracle, [En línea]. Disponible en: <https://www.oracle.com/mx/technical-resources/articles/cloud-infrastructure/what-is-restful-api.html> [Consultado: Nov. 2025].
- [19] PostgreSQL Global Development Group, “PostgreSQL: The World’s Most Advanced Open Source Relational Database,” Sitio web oficial de PostgreSQL, [En línea]. Disponible en: <https://www.postgresql.org/> [Consultado: Nov. 2025].
- [20] Microsoft, “SQL Server Documentation,” Microsoft Learn, [En línea]. Disponible en: <https://learn.microsoft.com/en-us/sql/sql-server/> [Consultado: Nov. 2025].
- [21] Mozilla Contributors, “Http – overview,” <https://developer.mozilla.org/es/docs/Web/HTTP/Guides/Overview>, 2025, accessed: 2025-12-03.
- [22] *IEEE Recommended Practice for Software Requirements Specifications*, The Institute of Electrical and Electronics Engineers Std., Jun 1998.
- [23] Microsoft, “ASP.NET Core documentation,” Microsoft Learn, [En línea]. Disponible en: <https://learn.microsoft.com/en-us/aspnet/core/> [Consultado: Nov. 2025].
- [24] ——, “Entity Framework Core documentation,” Microsoft Learn, [En línea]. Disponible en: <https://learn.microsoft.com/en-us/ef/core/> [Consultado: Nov. 2025].

- [25] GitHub, “About projects,” GitHub Docs, [En línea]. Disponible en: <https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects> [Consultado: Nov. 2025].
- [26] Uruguay, “Decreto N° 104/019. Modificación de la reglamentación de la Ley N° 18.331,” IMPO - Base de datos de Decretos, Abr 2019, [En línea]. Disponible en: <https://www.impo.com.uy/bases/decretos/104-2019> [Consultado: Nov. 2025].
- [27] ——, “Decreto N° 406/022. Accesibilidad Digital,” IMPO - Base de datos de Decretos, Dic 2022, [En línea]. Disponible en: <https://www.impo.com.uy/bases/decretos/406-2022> [Consultado: Nov. 2025].
- [28] W3C (World Wide Web Consortium), “Web Content Accessibility Guidelines (WCAG) 2.1,” Web Accessibility Initiative (WAI) Recommendation, 2018, [En línea]. Disponible en: <https://www.w3.org/TR/WCAG21/> [Consultado: Nov. 2025].
- [29] Microsoft, “Microsoft Excel: Software de hoja de cálculo líder en el mercado,” Sitio web oficial de Microsoft, [En línea]. Disponible en: <https://www.microsoft.com/es-es/microsoft-365/excel> [Consultado: Nov. 2025].
- [30] Poder Legislativo de Uruguay, “Ley N° 18.600: Documento Electrónico y Firma Electrónica,” Diario Oficial de Uruguay, 2009, [En línea]. Disponible en: <http://www.impo.com.uy/bases/leyes/18600-2009> [Consultado: Nov. 2025].
- [31] Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento (AGESIC), “Identidad Digital en Uruguay,” Portal de la AGESIC, [En línea]. Disponible en: <https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/identidad-digital> [Consultado: Nov. 2025].
- [32] Banco de Previsión Social (BPS), “Concursos,” <https://www.bps.gub.uy/61/concursos.html>, 2025, accedido: 2025-02-12.

- [33] Banco de Seguros del Estado (BSE), “Portal Institucional - Concursos - Inscripción,” Portal web del BSE, 2025, [En línea]. Disponible en: <https://institucional.bse.com.uy/portal-institucional/concursos/inscripcion/> [Consultado: Nov. 2025].
- [34] United Nations, “E-Government Survey 2020: Digital Government in the Decade of Action for Sustainable Development,” Nueva York, 2020, [En línea]. [Consultado: Nov. 2025]. [Online]. Available: [https://publicadministration.un.org/egovkb/Portals/egovkb/Documents/un/2020-Survey/2020%20UN%20E-Government%20Survey%20\(Full%20Report\).pdf](https://publicadministration.un.org/egovkb/Portals/egovkb/Documents/un/2020-Survey/2020%20UN%20E-Government%20Survey%20(Full%20Report).pdf)
- [35] OECD (Organisation for Economic Co-operation and Development), “The E-Government Imperative,” París, 2003, [En línea]. Disponible en: <https://www.oecd-ilibrary.org/docserver/9789264018217-en.pdf> [Consultado: Nov. 2025].
- [36] K. Layne and J. Lee, “Developing fully functional e-government: A four stage model,” *Government information quarterly*, vol. 18, no. 2, pp. 122–136, 2001.
- [37] United Nations, “United nations official website,” <https://www.un.org/>, 2025, accessed: 2025-12-03.
- [38] M. Fowler, *Patterns of enterprise application architecture*. Addison-Wesley, 2012.
- [39] M. Seemann, *Dependency Injection in .NET*. Greenwich, CT: Manning Publications, 2011.
- [40] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.
- [41] ECMA International, “Ecma-404: The json data interchange standard,” <https://www.ecma-international.org/publications-and-standards/standards/ecma-404/>, 2017, accessed: 2025-12-03.
- [42] TechEmpower, “Web Framework Benchmarks,” Sitio web oficial de TechEmpower, [En línea]. Disponible en: <https://www.techempower.com/benchmarks/> [Consultado: Nov. 2025].

- [43] Microsoft, “Microsoft Home Page,” microsoft.com, [En línea]. Disponible en: <https://www.microsoft.com/> [Consultado: Nov. 2025].
- [44] NuGet Team, “Nuget package manager,” <https://www.nuget.org/>, 2025, accedido: 2025-12-03.
- [45] Microsoft, “Language integrated query (linq),” <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>, 2024, accessed: 2025-12-03.
- [46] E. Evans, *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.
- [47] Oracle Corporation, “Mysql database,” <https://www.mysql.com/>, 2025, accessed: 2025-12-03.
- [48] ——, “Oracle database,” <https://www.oracle.com/database/>, 2025, accessed: 2025-12-03.
- [49] Microsoft, “SQL Server Management Studio (SSMS) Documentation,” Microsoft Learn, [En línea]. Disponible en: <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms> [Consultado: Nov. 2025].
- [50] *Information technology — Database languages — SQL (Standard SQL:2016)*, ISO/IEC Std., 2016.
- [51] W3C (World Wide Web Consortium), “Introduction to Web Accessibility,” Web Accessibility Initiative (WAI), [En línea]. Disponible en: <https://www.w3.org/WAI/fundamentals/accessibility-intro/> [Consultado: Nov. 2025].
- [52] Uruguay, “Ley N° 13.921. Creación de la Dirección General de Casinos (Consolidación),” IMPO - Base de datos de Leyes, 1970, [En línea]. Disponible en: <https://www impo com uy/bases/leyes/13921-1970> [Consultado: Nov. 2025].

- [53] D. Zowghi and C. Coulin, “Requirements elicitation: A survey of techniques, approaches, and tools,” in *Engineering and Managing Software Requirements*. Springer, 2005, pp. 19–46.
- [54] H. Sharp, A. Finkelstein, and G. Galal, “Stakeholder identification in the requirements engineering process,” in *Proceedings of the 10th International Workshop on Database and Expert Systems Applications (DEXA)*, 1999, pp. 387–391.
- [55] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.
- [56] J. Nielsen, “Heuristic evaluation,” in *Usability Inspection Methods*, J. Nielsen and R. L. Mack, Eds. New York: John Wiley & Sons, 1994, pp. 25–62.
- [57] MDN Web Docs, “Aria – accesibilidad,” <https://developer.mozilla.org/es/docs/Web/Accessibility/ARIA>, 2025, accedido: 2025-12-03.
- [58] ISO/IEC, “ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models,” <https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>, 2011, accessed: 2025-02-12.
- [59] A. Cockburn, *Writing Effective Use Cases*. Boston, MA: Addison-Wesley, 2000.
- [60] M. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT),” RFC 7519, May 2015, accessed: 2025-02-12. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7519>
- [61] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, MA: Addison-Wesley, 2003.
- [62] Framer B.V., “Framer – interactive design and prototyping tool,” <https://www.framer.com/>, 2025, accessed: 2025-02-12.
- [63] J. Nielsen, *Usability Engineering*. San Francisco: Morgan Kaufmann, 1993.

- [64] Software Freedom Conservancy, “Git – distributed version control system,” <https://git-scm.com/>, 2025, accessed: 2025-12-03.
- [65] Atlassian, “Gitflow workflow,” <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>, 2025, accedido: 2025-12-03.
- [66] Conventional Commits, “Conventional Commits Specification,” <https://www.conventionalcommits.org/>, accessed: 2025-02-12.
- [67] GitHub, “Github actions – continuous integration and automation,” <https://github.com/features/actions>, 2025, accessed: 2025-12-03.
- [68] M. Cohn, *Succeeding with Agile: Software Development Using Scrum*. Boston, MA: Addison-Wesley, 2009.
- [69] xUnit.net, “xunit.net – unit testing framework for .net,” <https://xunit.net/>, 2025, accessed: 2025-12-03.
- [70] MermaidJS, “Mermaid – markdown-based diagramming and charting tool,” <https://mermaid.js.org/>, 2025, accessed: 2025-12-03.
- [71] JGraph Ltd., “diagrams.net (draw.io),” <https://www.drawio.com/>, 2025, accessed: 2025-12-03.
- [72] Change Vision, Inc., “Astah – uml and modeling tool,” <https://astah.net/>, 2025, accessed: 2025-12-03.
- [73] Microsoft Corporation, “Microsoft teams,” <https://www.microsoft.com/microsoft-teams>, 2025, accessed: 2025-12-03.
- [74] Notion Labs Inc., “Notion – all-in-one workspace,” <https://www.notion.so/>, 2025, accessed: 2025-12-03.
- [75] Miro, “Miro – online collaborative whiteboard platform,” <https://miro.com/>, 2025, accessed: 2025-12-03.

- [76] Lucid Software Inc., “Lucidchart – visual workspace,” <https://www.lucidchart.com/>, 2025, accessed: 2025-12-03.
- [77] Microsoft Corporation, “Visual studio code,” <https://code.visualstudio.com/>, 2025, accessed: 2025-12-03.
- [78] ——, “Visual studio 2022,” <https://visualstudio.microsoft.com/vs/>, 2025, accessed: 2025-12-03.
- [79] Postman, Inc., “Postman – api development platform,” <https://www.postman.com/>, 2025, accessed: 2025-12-03.
- [80] GitHub, Inc., “Github – code hosting and collaboration platform,” <https://github.com/>, 2025, accessed: 2025-12-03.
- [81] Overleaf Ltd., “Overleaf – online latex editor,” <https://www.overleaf.com/>, 2025, accessed: 2025-12-03.
- [82] OpenJS Foundation, “JSDoc – api documentation generator for javascript,” <https://jsdoc.app/>, 2025, accessed: 2025-12-03.
- [83] Microsoft Corporation, “Xml documentation comments (c# programming guide),” <https://learn.microsoft.com/dotnet/csharp/language-reference/xmldoc/>, 2025, accessed: 2025-12-03.
- [84] Google DeepMind, “Gemini flash 2.5 model family,” <https://ai.google.dev/gemini-api/docs/models/gemini-flash-2.5>, 2025, accedido: 2025-12-03.
- [85] Anthropic, “Claude sonnet 4.5 model specification,” <https://www.anthropic.com/clause>, 2025, accedido: 2025-12-03.
- [86] OpenAI, “Codex 5.1 (preview) model specification,” <https://platform.openai.com/docs/models#code-models>, 2025, accedido: 2025-12-03.
- [87] GitHub, “Github copilot,” <https://github.com/features/copilot>, 2025, accedido: 2025-12-03.

- [88] OpenAI, “Gpt-5.1 model specification,” <https://platform.openai.com/docs/models>, 2025, accedido: 2025-12-03.
- [89] Microsoft, “Arrange, act, assert testing pattern,” <https://learn.microsoft.com/dotnet/core/testing/unit-testing-best-practices#arrange-act-assert>, 2025, accedido: 2025-12-03.
- [90] GitHub, “Collaborating with pull requests,” <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests>, 2025, accedido: 2025-12-03.
- [91] NEOGOV, “Transforming Public Sector HR with a Unified HCM Platform,” Portal web de NEOGOV, 2025, [En línea]. Disponible en: <https://www.neogov.com/> [Consultado: Nov. 2025].
- [92] JobAps, “Full Spectrum Enterprise Recruiting Exclusively for the Public Sector,” Portal web de JobAps, 2025.
- [93] Eploy, “Recruitment Software, Applicant Tracking System (ATS), and Talent Acquisition Suite,” Portal web de eploy, 2025, [En línea]. Disponible en: <https://www.employ.com/> [Consultado: Nov. 2025].
- [94] Tribepad, “Talent Acquisition Software & ATS | The complete recruiting platform,” Portal web de Tribepad, 2025, [En línea]. Disponible en: <https://tribepad.com/> [Consultado: Nov. 2025].
- [95] E. Parry and H. Wilson, “Factors influencing the adoption of online recruitment,” *Personnel Review*, vol. 38, no. 6, pp. 655–673, 2009.
- [96] M. Abia and I. Brown, “Conceptualizations of e-recruitment: A literature review and analysis,” in *Conference on e-Business, e-Services and e-Society*. Springer, 2020, pp. 370–379.
- [97] I. Lee, “E-recruiting system development and architecture,” in *Utilizing and Managing Commerce and Services Online*. IGI Global Scientific Publishing, 2007, pp. 234–257.

- [98] E. Parry and S. Tyson, “Desired goals and actual outcomes of e-hrm,” *Human resource management journal*, vol. 21, no. 3, pp. 335–354, 2011.
- [99] P. Bhushan, “The digital transformation in recruitment: exploring the effect and challenges of online recruitment,” *Available at SSRN 4885218*, 2024.
- [100] J. J. Llorens, “A model of public sector e-recruitment adoption in a time of hyper technological change,” *Review of Public Personnel Administration*, vol. 31, no. 4, pp. 410–423, 2011.
- [101] P. M. Gilch and J. Sieweke, “Recruiting digital talent: The strategic role of recruitment in organisations’ digital transformation,” *German Journal of Human Resource Management*, vol. 35, no. 1, pp. 53–82, 2021.
- [102] D. Stanković and M. Milosavljević, “The impact of e-recruitment implementation on company credibility, candidate selection efficiency, and process transparency,” *International Review of Management and Marketing*, 2024.
- [103] A. Ghaleb *et al.*, “Fairness in e-recruitment: Examining procedural justice and applicant reactions to online application websites,” *Sustainability*, 2024.
- [104] S. Lythreatis, S. Singh, and A.-N. El-Kassar, “The digital divide: A review and future research agenda,” *Technological Forecasting and Social Change*, vol. 175, pp. 121–141, 2022.
- [105] J. A. Van Dijk, *The Digital Divide*. Polity Press, 2020.
- [106] P. DiMaggio, E. Hargittai, C. Celeste, and S. Shafer, “From unequal access to differentiated use: A literature review and agenda for research on digital inequality,” *Poetics*, vol. 34, no. 4–5, pp. 269–299, 2004.
- [107] S. Barocas and A. D. Selbst, “Big data’s disparate impact,” *Calif. L. Rev.*, vol. 104, p. 671, 2016.

- [108] P. Jain, M. Gyanchandani, and N. Khare, “Big data privacy: a technological perspective and review,” *Journal of big data*, vol. 3, no. 1, p. 25, 2016.
- [109] G. Koman, D. Toman, R. Jankal, and P. Boršoš, “The importance of e-recruitment within a smart government framework,” *Systems*, vol. 12, no. 3, p. 71, 2024.
- [110] A. Deshpande, “Affirmative action in india,” in *Race and Inequality*. Routledge, 2017, pp. 77–90.
- [111] R. o. S. A. Department of Labour, *Employment Equity Act (No. 55 of 1998)*. Government Gazette, 1998.
- [112] “Regents of the university of california v. bakke, 438 u.s. 265,” 1978, supreme Court of the United States.
- [113] K. Beck *et al.*, *Manifesto for Agile Software Development*, 2001.
- [114] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall, 2003.
- [115] R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.
- [116] SeleniumHQ, “Selenium webdriver,” <https://www.selenium.dev/>, 2025, accessed: 2025-12-03.
- [117] Unidad Reguladora y de Control de Datos Personales (URCDP), “Sitio oficial de la urcdp,” <https://www.gub.uy/unidad-reguladora-control-datos-personales/>, 2025, accessed: 2025-12-03.
- [118] E. Rescorla, “HTTP Over TLS,” RFC 2818, 2000, accessed: 2025-12-03. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc2818>
- [119] ——, “The transport layer security (tls) protocol version 1.3,” RFC 8446, 2018, accessed: 2025-12-03. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc8446>

- [120] National Institute of Standards and Technology, “Advanced encryption standard (aes),” U.S. Department of Commerce, Tech. Rep. FIPS PUB 197, 2001, accessed: 2025-12-03. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>
- [121] J. Klensin, “Simple mail transfer protocol,” RFC 5321, 2008, accessed: 2025-12-03. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc5321>
- [122] C. Cremers, L. Garratt, S. Smyshlyaev, N. Sullivan, and C. A. Wood, “Randomness improvements for security protocols,” RFC 8937, 2020, accessed: 2025-12-03. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc8937>

Apéndice A: Evidencia de la Plataforma Uruguay Concursa

Reporte de Accesibilidad — WAVE

La Figura A.1 muestra los resultados obtenidos con la herramienta *WAVE Web Accessibility Evaluation Tool*, donde se identificaron errores de accesibilidad que afectan la experiencia del usuario, incluyendo:

- (i) imágenes sin texto alternativo,
- (ii) encabezados y estructura semántica inconsistente,
- (iii) contraste insuficiente, y
- (iv) elementos interactivos sin etiqueta accesible.

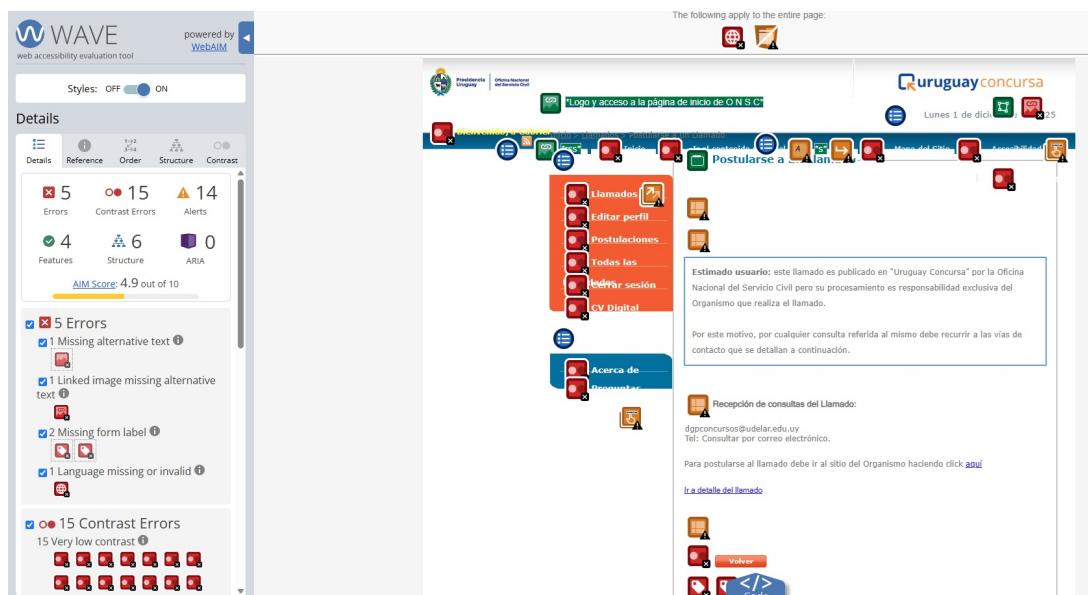


Figura A.1: Reporte WAVE de accesibilidad correspondiente a Uruguay Concursa.

Errores de Carga y Navegación

Durante el proceso de navegación y postulación se registraron errores de carga, como se presenta en la Figura A.2. Estos fallos generan incertidumbre en el usuario y pueden afectar la completitud del proceso de inscripción.

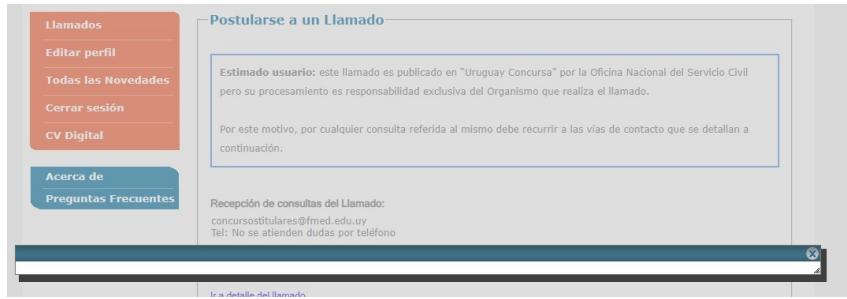


Figura A.2: Errores de carga detectados durante el proceso de postulación en la plataforma.

Desvío a Sitio Externo para Finalizar la Postulación

En algunos llamados, la plataforma redirige al usuario a sitios externos para completar la postulación, delegando parte del proceso fuera de Uruguay Concursa [14] (Figura A.3). Esto evidencia que el sistema actúa como portal intermedio y no como plataforma integral de gestión de concursos.



Figura A.3: Redirección a sitio externo para completar el proceso de postulación.

Apéndice B: Evidencia del Portal de Concursos del BPS

Formulario de Postulación del BPS

A continuación se presenta la captura del formulario oficial de postulación del Banco de Previsión Social (BPS) [12], utilizado para concursos públicos y llamados internos. La captura proporciona un ejemplo representativo de la interfaz, validaciones implementadas y estructura general del proceso de inscripción digital.

La captura de pantalla muestra el formulario de postulación en el sitio web del BPS. El formulario es dividido en secciones:

- Sección de datos personales:** Incluye campos para Nombre, Apellido, Email, Teléfono, Fecha de Nacimiento, Documento de Identidad, Dirección, Ciudad, Provincia, País, y Código Postal.
- Sección de datos laborales:** Permite especificar la antigüedad laboral, la fecha de inicio y fin, la antigüedad promedio, la antigüedad mínima, y la antigüedad máxima.
- Sección de documentación:** Se detallan los documentos que deben adjuntarse: Documento de Identidad, Contrato de Trabajo, Acta de nacimiento, y Documento de identidad de otro parentesco.
- Sección de acuerdo:** Una sección de acuerdo que incluye la declaración de que el documento es auténtico, la aceptación de las normas y condiciones del servicio, y la autorización para el manejo de datos personales.
- Botón final:** Un botón verde que dice "Enviar Postulación".

Figura B.1: Formulario de postulación del BPS.

Apéndice C: Evidencia del Portal del BSE

Reporte de Accesibilidad — WAVE

La Figura C.1 muestra los resultados del análisis realizado con la herramienta *WAVE Web Accessibility Evaluation Tool* sobre el portal de concursos del Banco de Seguros del Estado (BSE) [33]. Se detectaron errores significativos tales como:

- Etiquetas de formulario faltantes o vacías.
- Encabezados mal estructurados o sin contenido.
- Enlaces sin texto accesible.
- Referencias ARIA [57] rotas.
- Problemas de contraste severo en múltiples elementos de la interfaz.

Estos errores representan incumplimientos del Decreto 406/022 [27] sobre accesibilidad digital en organismos públicos.

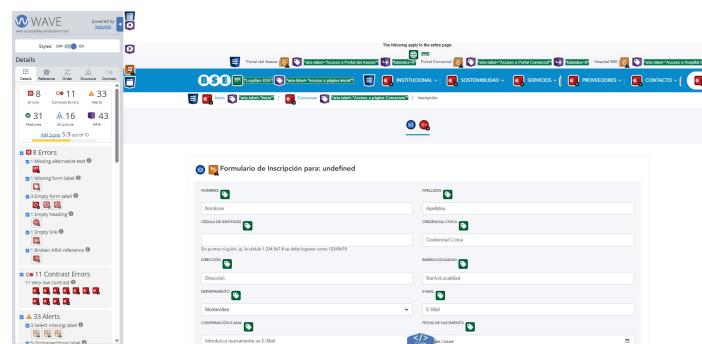


Figura C.1: Errores de accesibilidad detectados en el portal de concursos del BSE [33] mediante la herramienta WAVE.

Apéndice D: Sistemas Comerciales de Reclutamiento para el Sector Público

Sistemas Comerciales de Reclutamiento para el Sector Público

A nivel internacional, existen diversas soluciones comerciales especializadas en gestión de reclutamiento para organismos gubernamentales. El análisis de estos sistemas proporcionó perspectivas sobre funcionalidades avanzadas y mejores prácticas de la industria.

NEOGOV (Estados Unidos)

NEOGOV es una de las plataformas líderes en el mercado norteamericano para reclutamiento en el sector público, utilizada por más de 8,000 agencias gubernamentales [91].

Funcionalidades destacadas:

- Applicant Tracking System (ATS) diseñado específicamente para las necesidades del sector público
- Integración con portal de empleos público governmentjobs.com
- Clasificación automática de candidatos mediante algoritmos de ranking
- Sistema de gestión de relaciones con candidatos (CRM)
- Reportes comprehensivos de métricas de reclutamiento
- Portal de autoservicio para candidatos
- Herramientas para promover diversidad e inclusión en el proceso de selección

Relevancia para Portal DGC: NEOGOV [91] demostró la viabilidad técnica de automatizar rankings y aplicar algoritmos de selección en contextos gubernamentales. Su énfasis en cumplimiento normativo y en herramientas para promover diversidad se alinea con los objetivos del Portal DGC respecto a cuotas de acción afirmativa.

JobAps (Estados Unidos)

JobAps ofrece una solución integrada para reclutamiento gubernamental con énfasis en evaluaciones en línea y cumplimiento de requisitos legales [92].

Características distintivas:

- Screening y scoring automatizado basado en competencias
- Suite de testing en línea de alta seguridad
- Portal del aplicante optimizado para móviles
- Base de datos privada para cada organización
- Soporte personalizado especializado en sector público

Lección para Portal DGC: La capacidad de JobAps [92] para realizar evaluaciones automatizadas basadas en criterios predefinidos ayudó en el diseño del módulo de evaluación del tribunal en el Portal DGC, adaptado al contexto específico uruguayo.

Eploy y Tribepad (Reino Unido)

Estos sistemas británicos ofrecen soluciones ATS para el sector público del Reino Unido, cumpliendo con marcos regulatorios europeos[93][94].

Aspectos relevantes:

- Workflows configurables adaptables a diferentes procesos de cada organización

- Portal dedicado para hiring managers con interfaz simplificada
- Énfasis en diversidad e inclusión con reportes específicos
- Analytics y reporting comprehensivo
- Cumplimiento con GDPR (regulación europea de protección de datos)

Análisis: Estos sistemas demostraron que es posible construir soluciones flexibles que se adapten a las particularidades de cada organización sin sacrificar funcionalidad. El Portal DGC adoptó un enfoque similar de configurabilidad, permitiendo que cada llamado defina sus propios requisitos, méritos valorables y criterios de evaluación.

Limitaciones de Soluciones Comerciales

A pesar de sus capacidades avanzadas, ninguna de las soluciones comerciales analizadas implementa de forma nativa la aplicación automática de cuotas de acción afirmativa al estilo de las leyes uruguayas [4]. Estas plataformas están diseñadas para contextos donde la diversidad es un objetivo pero no una obligación legal con porcentajes específicos. Esta particularidad del marco normativo uruguayo requirió desarrollar desde cero la lógica de ordenamientos con cuotas en el Portal DGC.

Apéndice E: Marco Conceptual E-Recruitment

Marco Conceptual sobre E-Recruitment y Digitalización de RR. HH.

La literatura académica sobre reclutamiento electrónico (e-recruitment) y gestión digital de recursos humanos ha crecido significativamente en las últimas dos décadas, proporcionando marcos teóricos y evidencia empírica sobre los beneficios y desafíos de la digitalización de procesos de selección.

Conceptualización de E-Recruitment

La investigación de Parry y Wilson (2009)[95] estableció definiciones fundamentales de e-recruitment como “el uso de tecnología de internet para atraer candidatos potenciales y facilitar procesos de reclutamiento”. Estudios posteriores expandieron esta definición, identificando cinco conceptualizaciones alternativas[96]:

Estas conceptualizaciones se observan en la Tabla E.1.

1. **E-recruitment como Herramienta Tecnológica:** Enfoque en las tecnologías específicas utilizadas (portales web, redes sociales, sistemas ATS)
2. **E-recruitment como Sistema:** Visión holística del conjunto integrado de tecnologías y procesos
3. **E-recruitment como Proceso:** Énfasis en el flujo de actividades digitalizado desde la atracción hasta la contratación
4. **E-recruitment como Servicio:** Perspectiva centrada en la experiencia del usuario

5. E-recruitment como Proxy: Uso de tecnologías digitales como representante de modernización organizacional

El Portal DGC se alinea principalmente con las conceptualizaciones 2 y 3, implementando un sistema integral que digitaliza el proceso completo de selección.

Table 1. Conceptualizations of E-recruitment

Conceptualization of e-Recruitment (Label Created in this Study)	Description
1. E-recruitment as a Technology Tool	E-recruitment is viewed in some studies as a technology tool.
2. E-recruitment as a System	E-recruitment is a group of independent but interrelated elements comprising a unified whole. These elements include technology, society, organizations, etc.
3. E-recruitment as a Process	E-recruitment is a set of systematic well-coordinated activities. The activities are done by information technology or traditionally.
4. E-recruitment as a Service <ul style="list-style-type: none"> a. E-recruitment as a Repository b. E-recruitment as a Medium c. E-recruitment as a Program (E-recruitment as an Implemented Algorithm) 	E-recruitment is a service to recruitment. It cannot be entrusted to do all that is needed for successful recruitment, therefore it only provides certain functionalities. <ul style="list-style-type: none"> a. E-recruitment provides storage facilities for recruitment data. b. E-recruitment is a communications conduit between stakeholders in recruitment. c. E-recruitment is a set of precise rules for solving a problem.
5. E-recruitment as a Proxy	E-recruitment acts on behalf of organizational and societal entities.

Figura E.1: Conceptualizaciones de E-Recruitment según Abia et al. (2020).

Beneficios Documentados de E-Recruitment

La literatura académica ha documentado consistentemente beneficios significativos de la implementación de sistemas de reclutamiento digital:

Eficiencia Operativa: Estudios empíricos, apoyados por trabajos clave en la materia (Lee, 2007[97]; Parry & Tyson, 2011[98]; Bhushan, 2024[99]), indican que la adopción del e-recruitment se traduce en importantes reducciones de tiempo (típicamente entre 25-50 %) en el procesamiento de candidatos y reducciones de costos administrativos (cercanas al 30-60 %), en comparación con los métodos tradicionales en papel.

Por ejemplo, este efecto ha sido documentado por Parry y Tyson[98]. Los autores sintetizan la evidencia empírica señalando que:

“Empirical research has supported the existence of efficiency gains through e-HRM, by reducing HR staff, increasing the speed of processes, reducing costs and releasing staff from administrative work (Strohmeier, 2007; Ruta, 2005; Ruel et al., 2004). Indeed, Ruel et al. (2004) found that the most common outcomes of e-HRM were a reduction of costs and of the administrative burden on HR practitioners, although these outcomes were not generally measured. [98]”

Alcance Ampliado: La digitalización permite acceder a una base más amplia de candidatos potenciales, ampliando el alcance territorial y reduciendo barreras de acceso al proceso de selección (Llorens, 2011[100]).

Mejora de Calidad de Contratación: Sistemas estructurados de evaluación y comparación pueden permitir decisiones de selección más objetivas y basadas en datos. Según Gilch & Sieweke (2021)[101], el reclutamiento digital redefine procesos y roles, lo que podría favorecer una selección más alineada con las necesidades organizativas.

Consistencia y Cumplimiento: La automatización puede favorecer una aplicación más uniforme de los criterios de evaluación, reduciendo variabilidad entre evaluadores y aumentando la transparencia del proceso de selección. Estudios recientes muestran que los sistemas de e-recruitment tienden a mejorar la eficiencia y la justicia procesal cuando están correctamente diseñados e implementados (Stanković & Milosavljević, 2024[102]; Ghaleb et al., 2024[103]).

Desafíos y Consideraciones Críticas

La investigación también ha identificado desafíos importantes que deben considerarse en implementaciones de e-recruitment:

Brecha Digital: La literatura señala que la digitalización no garantiza equidad de acceso: existen diferencias significativas en conectividad, habilidades digitales y condiciones socioeconómicas que pueden limitar la participación en plataformas en línea. Estas desigualdades se manifiestan en distintos niveles (acceso, uso y apropiación) documentados en estudios recientes (Lythreatis et al., 2022[104]; Van Dijk, 2020[105]; DiMaggio & Hargittai, 2004[106]). Por esta razón, el diseño del Portal DGC incorporó interfaces intuitivas y opciones de asistencia para mitigar este riesgo.

Sesgo Algorítmico: La literatura especializada advierte que los sistemas digitales pueden reproducir o amplificar sesgos existentes cuando los algoritmos se entrena con datos no representativos o se diseñan sin consideraciones explícitas de equidad. Baracas y Selbst (2016)[107] señalan que incluso sistemas técnicamente neutrales pueden generar resultados discriminatorios si se apoyan en patrones históricos sesgados.

“Big data technologies can reproduce existing patterns of discrimination, inherit the biases of prior decision makers, or simply reflect the widespread biases that persist in society. [107]”

Si bien el Portal DGC no utiliza algoritmos de filtrado automático, el diseño contempla medidas preventivas mediante criterios transparentes y la aplicación explícita de acciones afirmativas [4] para mitigar potenciales desigualdades.

Privacidad y Protección de Datos: La digitalización implica almacenamiento y procesamiento de grandes volúmenes de datos personales sensibles, lo que demanda salvaguardias robustas. Como señalan Jain et al. (2016)[108], el uso de big data conlleva riesgos significativos

de privacidad y seguridad, especialmente cuando se manejan datos personales en contextos masivos.

“Big data analytics introduces serious privacy and security issues, since high-volume, high-variety data sets often contain sensitive personal information requiring robust privacy-preserving mechanisms. [108]”

Por ello, el Portal DGC adoptará medidas de seguridad, anonimización y políticas de protección de datos para resguardar la información de los postulantes.

E-Recruitment en el Sector Público: Particularidades

Llorens (2011)[100] señala que la adopción de e-recruitment en administraciones públicas presenta desafíos específicos que no siempre se observan en el sector privado.

“Public sector recruitment operates under strict requirements of transparency, accountability, and legal compliance, which shape the design and implementation of e-recruitment systems.[100]”

Entre ellos destacan:

- **Transparencia y Auditabilidad:** Los procesos de selección pública requieren trazabilidad completa y capacidad de verificación externa.
- **Marco Regulatorio Rígido:** Las decisiones deben ajustarse a normas específicas (meritos, concursos, cuotas, antigüedad).
- **Limitaciones Presupuestarias:** Los organismos públicos suelen contar con menor flexibilidad para invertir en nuevas tecnologías.
- **Resistencia al Cambio:** Las culturas administrativas pueden mostrar menor adopción de innovaciones digitales.

El Portal DGC fue diseñado considerando estas particularidades del sector público uruguayo, priorizando transparencia, auditabilidad y cumplimiento normativo riguroso.

E-Recruitment en el marco de Smart Government

Recientes estudios han señalado que el e-recruitment puede formar parte integral de un modelo de Smart Government, en el cual los sistemas de recursos humanos se articulan con otros servicios públicos digitales mediante datos interoperables y estándares comunes. En particular, un trabajo de 2024 sostiene que:

“E-recruitment is a key functional module of e-HRM within smart governance, enabling efficient, transparent and integrated public administration[109].”

El Portal DGC se alinea con esta visión al ser diseñado con APIs RESTful [18] que permitirían integraciones futuras con otros sistemas del Estado uruguayo, como Uruguay Concursa [14] para postulantes que apliquen a múltiples organismos.

Apéndice F: Cuotas de Acción Afirmativa - Perspectiva Comparada

Marco Legal de Cuotas de Acción Afirmativa

La implementación de cuotas de acción afirmativa en procesos de selección pública no es exclusiva de Uruguay. El análisis comparativo de marcos legales en otros países proporcionó contexto y referentes para el diseño del módulo de ordenamientos del Portal DGC.

India: Sistema de Reservas (Reservations)

India implementa uno de los sistemas de acción afirmativa más amplios del mundo, conocido como “reservations”. Establecido por la Constitución de 1950 y desarrollado a través de sucesivas enmiendas, el sistema reserva:

- 15 % de posiciones en empleo público para Scheduled Castes (SC)
- 7.5 % para Scheduled Tribes (ST)
- 27 % para Other Backward Classes (OBC)
- Cuotas adicionales a nivel estatal para categorías específicas

Como señala Deshpande (2013)[110], India opera “uno de los sistemas de acción afirmativa más extensos y complejos del mundo”, combinando criterios constitucionales con procedimientos administrativos altamente estructurados.

Relevancia: Este caso demuestra la viabilidad de gestionar múltiples cuotas simultáneamente mediante sistemas computarizados y reglas formales, aunque también ha generado debates persistentes sobre equidad, justicia distributiva y meritocracia.

Sudáfrica: Employment Equity Act

Tras el apartheid, Sudáfrica promulgó el Employment Equity Act (1998)[111], una legislación dirigida a corregir la discriminación laboral histórica mediante la promoción de una representación más equitativa en el empleo. A diferencia de sistemas basados en cuotas rígidas, el EEA establece “numerical targets” que los empleadores deben fijar y demostrar que intentan alcanzar de buena fe, más que cumplir porcentajes obligatorios.

Lección: La experiencia sudafricana subraya la importancia de contar con mecanismos sólidos de monitoreo, reporte y auditoría para evaluar el avance hacia metas de diversidad, incluso cuando estas no funcionan como cuotas estrictamente obligatorias.

Estados Unidos: Affirmative Action

A diferencia de los sistemas basados en cuotas rígidas (prohibidos por la Corte Suprema en el caso *Regents of the University of California v. Bakke*, 1978[112]), el modelo estadounidense de *affirmative action* permite considerar factores como raza o género únicamente como uno entre múltiples criterios en decisiones de contratación. Este enfoque se estructura en torno a “goals and timetables”, es decir, metas flexibles y esfuerzos de buena fe, pero no establece porcentajes obligatorios ni cupos numéricos.

Distinción: El sistema uruguayo, al igual que India pero a diferencia de Estados Unidos, sí establece cuotas rígidas de cumplimiento obligatorio. Esta característica implicó que el Portal DGC debiera implementar lógica específica que garantice el cumplimiento exacto de los porcentajes legales, y no simplemente metas aspiracionales.

Implicaciones para Portal DGC

El análisis comparativo reveló que Uruguay tiene un enfoque intermedio: cuotas obligatorias pero limitadas (13 % en total: 8 % afro + 1 % trans + 4 % discapacidad), dejando 87 % de posiciones para competencia por mérito general. Este balance requirió diseñar un algoritmo que:

1. Respete estrictamente los porcentajes legales
2. Mantenga el mérito como criterio primario de ordenamiento
3. Sea completamente transparente y auditabile
4. Maneje casos límite (ej: cuando no hay suficientes postulantes en una categoría de cuota)

Apéndice G: Trabajos Integradores en Ingeniería de Software

Contexto Académico: Trabajos Integradores en Ingeniería de Software

Este proyecto se desarrolla como trabajo integrador de la carrera de Ingeniería en Sistemas de la Universidad ORT Uruguay, modalidad que requiere aplicar conocimientos adquiridos en el programa académico para resolver una problemática real mediante desarrollo de software.

Trabajos Similares en ORT Uruguay

Una búsqueda en el repositorio académico de la Universidad ORT Uruguay no identificó trabajos integradores específicamente enfocados en digitalización de procesos de concursos públicos con aplicación de cuotas de acción afirmativa. Sin embargo, existen precedentes de proyectos en áreas relacionadas:

- Sistemas de gestión de recursos humanos para empresas privadas
- Plataformas de e-learning para capacitación de personal
- Sistemas de gestión de turnos y asignación de personal
- Portales institucionales con workflows de aprobación

La ausencia de trabajos específicamente similares al Portal DGC refuerza la originalidad y el aporte del presente proyecto, que aborda una problemática real del sector público uruguayo mediante una solución tecnológica innovadora en el contexto nacional.

Metodologías de Desarrollo en Proyectos Académicos

Los trabajos integradores de Ingeniería de Software en ORT Uruguay típicamente aplican:

- Metodologías ágiles adaptadas al contexto académico (sprints semanales, entregas incrementales) [113].
- Arquitecturas en capas siguiendo principios SOLID [114].
- Tecnologías modernas de la industria (.NET, Angular [16], React [17], Node.js)
- Prácticas de Ingeniería de Software (control de versiones, testing automatizado, Continuous Integration / Continuous Deployment básico (CI/CD))
- Documentación técnica siguiendo estándares (IEEE 830 [22] para requisitos, Unified Modeling Language (UML) para diseño)

El Portal DGC se alinea con estas prácticas establecidas, aplicando metodología ágil con sprints semanales, arquitectura en tres capas, stack tecnológico moderno (ASP.NET Core + Angular [16]), y documentación exhaustiva siguiendo estándares académicos y profesionales.

Apéndice H: Detalle de Requisitos Implementados

Requisitos Funcionales Implementados

De los 21 requisitos funcionales definidos durante la fase de análisis, se implementaron 14 que constituyen el núcleo de valor del sistema. Esta sección documenta cada uno de ellos con su descripción completa.

RF-02: Consulta de Llamados Disponibles

Descripción: El sistema permite consultar llamados públicos activos con capacidad de filtrado por departamento y cargo.

Implementación: Interfaz de consulta con listado paginado que muestra título del llamado, plazas disponibles, departamentos donde aplica, y fecha de cierre. Incluye filtros dinámicos que actualizan los resultados sin recargar la página.

RF-03: Visualización Detallada de Llamado

Descripción: Al seleccionar un llamado, el sistema despliega información completa que incluye descripción del cargo, requisitos excluyentes, criterios de evaluación de méritos con puntajes máximos por categoría, pruebas a rendir, cronograma de etapas, y cupos por ley.

Implementación: Vista detallada con organización en secciones claramente delimitadas. Incluye botón de acción primaria “Inscribirse” que inicia el workflow de inscripción.

RF-04: Workflow de Inscripción en 6 Pasos

Descripción: Proceso guiado que estructura la recolección de información de inscripción en pasos secuenciales con validaciones progresivas.

Implementación: Componente workflow con indicador visual de progreso (stepper numerado), navegación entre pasos (Anterior/Siguiente), validación en cada paso antes de permitir avanzar, y persistencia de datos entre pasos. Los 6 pasos implementados son:

1. Selección de departamento
2. Autodefinición según categorías legales
3. Declaración de requisitos excluyentes
4. Carga de méritos
5. Solicitud de apoyos (condicional)
6. Confirmación

RF-05: Generación de Código Único de Inscripción

Descripción: Al confirmar la inscripción, el sistema genera automáticamente un código único alfanumérico que identifica inequívocamente esa inscripción.

Implementación: Generación de código mediante GUID (Globally Unique Identifier) que garantiza unicidad. El código se muestra en pantalla de confirmación y se permite descargar como archivo de texto. Este código sirve tanto para el postulante como identificador de su postulación, como para el sistema como llave primaria para relacionar evaluaciones y ordenamientos.

RF-06: Carga de Documentos Probatorios

Descripción: El sistema permite adjuntar archivos PDF como respaldo documental de méritos declarados.

Implementación: Componente de carga de archivos con validación de tipo MIME (solo PDF permitido), tamaño máximo (5 MB por archivo), y cantidad máxima de archivos por mérito. En el MVP, los archivos se almacenan en el sistema de archivos local del servidor; para producción se requiere integración con Azure Blob Storage o similar.

RF-07: Autodefinición según Categorías Legales

Descripción: El sistema solicita la autodefinición voluntaria como persona afrodescendiente, trans o con discapacidad, cumpliendo con las Leyes 19.122 [7], 19.684 [8] y 18.651 [9].

Implementación: Formulario con checkboxes independientes para cada categoría, permitiendo selección múltiple. Incluye textos explicativos sobre el propósito legal de estos datos y se enfatiza su carácter opcional mediante etiquetas "(Opcional)" visibles. La información se almacena vinculada a la inscripción, no al perfil del postulante, preservando privacidad.

RF-08: Declaración de Cumplimiento de Requisitos Excluyentes

Descripción: El sistema presenta la lista de requisitos excluyentes específicos del llamado y solicita al postulante declarar el cumplimiento de cada uno.

Implementación: Formulario que muestra dinámicamente los requisitos excluyentes configurados para el llamado específico, con un checkbox individual por requisito. Todos los checkboxes son obligatorios; el sistema no permite avanzar si alguno queda sin marcar. Esta declaración constituye responsabilidad del postulante y puede ser verificada posteriormente mediante la documentación adjunta.

RF-09: Solicitud de Apoyos para Personas con Discapacidad

Descripción: Si el postulante declaró tener discapacidad, el sistema habilita automáticamente un paso donde puede solicitar apoyos específicos necesarios durante las pruebas.

Implementación: Formulario condicional que se activa solo si se marcó discapacidad en el paso de autodefinición. Presenta opciones predefinidas mediante checkboxes (intérprete de lengua de señas, tiempo adicional, formato accesible de materiales, magnificador de pantalla, lector de pantalla, mouse adaptado, acompañante terapéutico, textos ampliados) y un campo de texto libre para especificar necesidades no contempladas en las opciones estándar.

RF-10: Panel "Mis Inscripciones"

Descripción: Cada postulante puede visualizar todas sus inscripciones activas, consultar su estado actual (Recibida, En Evaluación, Evaluada, Finalizada) y acceder al detalle de cada una.

Implementación: Vista tabular que lista las inscripciones del usuario autenticado, mostrando llamado al que postuló, departamento seleccionado, fecha de inscripción, estado actual, y acción de "Ver Detalle". El estado se actualiza automáticamente en base a las acciones del tribunal evaluador.

RF-11: Dashboard de Tribunal con Estadísticas en Tiempo Real

Descripción: La pantalla inicial del módulo de tribunal presenta métricas agregadas del proceso de evaluación que se actualizan automáticamente.

Implementación: Componente dashboard con tarjetas de métricas que muestran total de inscripciones recibidas, inscripciones pendientes de evaluación, inscripciones evaluadas completamente, promedio de puntaje obtenido, distribución por departamentos seleccionados, y distribución por autodefinición de cupos. Las métricas se calculan mediante consultas agregadas a la base de datos y se cachean durante 5 minutos para optimizar rendimiento.

RF-12: Lista de Inscripciones con Filtros Dinámicos

Descripción: Interfaz tabular que permite al tribunal visualizar todas las inscripciones del llamado con capacidades avanzadas de filtrado y búsqueda.

Implementación: Tabla con columnas: nombre del postulante, cédula, departamento seleccionado, estado de evaluación, puntaje obtenido (si aplica), y acción de “Evaluar”. Incluye filtros por departamento, estado de evaluación (pendiente, parcial, completa), cupo (general, afrodescendiente, trans, discapacidad), búsqueda por nombre o cédula, y ordenamiento por cualquier columna. Los filtros operan del lado del servidor para soportar grandes volúmenes de datos.

RF-13: Calificación de Méritos con Cálculo Automático de Puntajes

Descripción: El tribunal puede asignar puntajes a cada mérito declarado por el postulante, y el sistema calcula automáticamente el puntaje total obtenido.

Implementación: Formulario de calificación que lista todos los méritos del postulante organizados por categoría. Para cada mérito se muestra la descripción, el documento probatorio adjunto (con opción de descarga), y un input numérico para asignar puntaje entre 0 y el máximo definido para ese ítem. El sistema valida que el puntaje asignado no exceda el máximo permitido. Opcionalmente, el evaluador puede agregar observaciones textuales. El puntaje total se recalcula automáticamente cada vez que se modifica una calificación individual.

RF-14: Calificación de Pruebas con Validación de Umbrales

Descripción: El tribunal puede registrar los resultados de pruebas escritas, prácticas u orales, y el sistema valida automáticamente si se cumple el umbral mínimo de aprobación.

Implementación: Formulario que lista las pruebas definidas para el llamado. Para cada prueba, el evaluador ingresa el puntaje obtenido (usualmente escala 0-60). El sistema valida que el puntaje esté en el rango válido y compara automáticamente contra el umbral de aprobación configurado (usualmente 30 % = 18 puntos sobre 60). Marca visualmente con indicador verde o rojo

si la prueba fue aprobada o no. Si alguna prueba obligatoria no se aprueba, el postulante queda automáticamente excluido del ordenamiento final.

RF-15: Generación Automática de Ordenamientos con Cuotas

Descripción: El sistema genera automáticamente las listas de prelación aplicando las cuotas de acción afirmativa establecidas por ley.

Implementación: Algoritmo que ejecuta los siguientes pasos:

1. Filtra inscripciones que cumplan todos los requisitos excluyentes (verificado por declaración del postulante).
2. Filtra inscripciones que hayan aprobado todas las pruebas obligatorias (verificado automáticamente por validación de umbrales).
3. Calcula el puntaje total de cada inscripción sumando méritos, pruebas, y entrevista (si aplica).
4. Ordena por puntaje de mayor a menor.
5. Aplica las cuotas legales:
 - Reserva 8 % de posiciones para personas afrodescendientes (Ley 19.122 [7])
 - Reserva 1 % de posiciones para personas trans (Ley 19.684 [8])
 - Reserva 4 % de posiciones para personas con discapacidad (Ley 18.651 [9])
6. Genera ordenamiento general (todos los postulantes ordenados por mérito)
7. Genera ordenamientos por cupo (listas separadas por categoría legal)
8. Identifica posiciones cubiertas (compara cantidad de plazas disponibles con ordenamiento)

El algoritmo maneja casos límite como: cantidad insuficiente de postulantes en un cupo específico (la plaza se libera al cupo general), empate en puntajes (se aplican criterios de desempate configurables), y postulantes que pertenecen a múltiples cupos (se los incluye en todos los ordenamientos correspondientes).

Requisitos Funcionales Pendientes

Los 8 requisitos funcionales no implementados corresponden principalmente a funcionalidades de infraestructura y administración que, si bien son necesarias para un sistema en producción, no son críticas para validar la viabilidad del concepto en un MVP.

RF-01: Autenticación JWT con Refresh Tokens (CRÍTICO)

Descripción: Sistema de autenticación basado en JSON Web Tokens [60] con tokens de acceso de corta duración y tokens de refresco para renovación automática.

Justificación de pendiente: La implementación requiere infraestructura de almacenamiento seguro de tokens de refresco, manejo de expiración y renovación, y mecanismos de revocación. Para el MVP se implementó autenticación simulada que valida usuario/contraseña pero no genera tokens reales. Esta funcionalidad es CRÍTICA para producción pero no afecta la validación de los flujos principales del sistema.

RF-18: Sistema RBAC de Roles y Permisos (CRÍTICO)

Descripción: Sistema de autorización basado en roles (Role-Based Access Control) [115] que controla qué acciones puede realizar cada usuario según su rol asignado.

Justificación de pendiente: El diseño contempla tres roles principales (Postulante, Tribunal, RR. HH.) con permisos específicos para cada uno. La implementación requiere definir matriz de permisos, middleware de autorización en cada endpoint, y UI condicional según rol. Para el MVP se validaron los flujos con roles simulados (hardcoded), pero la implementación completa es CRÍTICA para producción.

RF-16: Sistema de Notificaciones Email/WhatsApp (ALTA prioridad)

Descripción: Envío automático de notificaciones a postulantes en eventos clave: confirmación de inscripción, cambios de estado, resultados publicados.

Justificación de pendiente: Requiere integración con servicios externos (servidor SMTP para email, API de WhatsApp Business). Si bien es altamente valorado por usuarios (notificación inmediata vs. consultar manualmente), no es crítico para validar los flujos core del sistema. Se priorizó implementar la funcionalidad de inscripción y evaluación completa antes que las notificaciones.

RF-17: Exportación de Resultados a Excel y PDF

Descripción: Los ordenamientos generados pueden exportarse en formatos estándar para distribución y publicación oficial.

Implementación: Dos opciones de exportación:

Excel (.xlsx): Utilizará la librería EPPlus para generar archivo de Excel con múltiples hojas: ordenamiento general, ordenamiento cupo afrodescendiente, ordenamiento cupo trans, ordenamiento cupo discapacidad, y hoja de estadísticas del proceso. Cada hoja incluirá columnas: posición, nombre del postulante, cédula, departamento, puntaje total méritos, puntaje total pruebas, puntaje total general, y estado (Cubierto/Suplente). El formato permitirá trabajar los datos posteriormente (ordenar, filtrar, hacer gráficos).

PDF: Tendrá que usar la librería iTextSharp para generar documento PDF formateado y listo para publicación oficial. Deberá incluir encabezado con logos institucionales (MEF, DGC), información del llamado (título, fecha, plazas disponibles), tabla con el ordenamiento general, tablas separadas con ordenamientos por cupo, y espacio para firmas digitales. El diseño tiene que cumplir con identidad visual institucional, siendo adecuado para publicar en el sitio web oficial o cartelería.

RF-19: Panel de Auditoría de Acciones del Sistema (MEDIA)

Descripción: Registro completo de todas las acciones relevantes realizadas en el sistema (quién hizo qué, cuándo) con interfaz de consulta para administradores.

Justificación de pendiente: Si bien importante para trazabilidad y cumplimiento normativo, no es esencial para el MVP. La implementación requiere logging estructurado de todas las operaciones, almacenamiento eficiente de logs, e interfaz de búsqueda y filtrado. Se priorizó funcionalidad core sobre capacidades de auditoría avanzadas.

RF-20: Reportes Estadísticos Avanzados (MEDIA)

Descripción: Generación de reportes analíticos sobre el proceso: tasas de aprobación por departamento, distribución de puntajes, efectividad de aplicación de cuotas, tendencias temporales.

Justificación de pendiente: El dashboard del tribunal proporciona métricas básicas en tiempo real, suficientes para validar el concepto. Reportes avanzados con visualizaciones interactivas y análisis histórico aportan valor para gestión estratégica pero no son críticos para operación diaria. Pueden desarrollarse iterativamente en base a necesidades reales identificadas en uso.

RF-21: Módulo RR. HH. para Creación de Llamados (MEDIA)

Descripción: Interfaz administrativa que permite al personal de RR. HH. crear nuevos llamados, configurar requisitos excluyentes, definir méritos valorables con sus puntajes, configurar pruebas, establecer cronograma, y publicar/despublicar llamados.

Justificación de pendiente: Para el MVP se utilizaron datos de prueba insertados directamente en base de datos mediante scripts SQL. Si bien la funcionalidad es necesaria para autonomía operativa, no afecta la validación de los flujos de inscripción y evaluación, que son el foco del POC. La implementación de este módulo completo puede realizarse en iteración futura con feedback de usuarios de RR. HH. sobre qué controles y validaciones necesitan.

Otras Funcionalidades Secundarias

Adicionalmente, quedaron pendientes funcionalidades menores de administración:

- Gestión de usuarios (alta, baja, modificación de perfiles)
- Configuración de parámetros del sistema (plazos, umbrales de aprobación)
- Gestión de catálogos (departamentos, cargos, tipos de pruebas)
- Recuperación de contraseña mediante email
- Cambio de contraseña por el usuario
- Panel de monitoreo de disponibilidad del sistema

Estas funcionalidades complementan el sistema pero no son esenciales para demostrar su viabilidad técnica y funcional como Proof of Concept.

Apéndice I: Especificaciones Técnicas de Arquitectura

Estructura Detallada de Capas

Capa de Presentación (Frontend - Angular 19)

Módulos Funcionales

La aplicación Angular se organizó en 8 módulos funcionales que encapsulan responsabilidades específicas:

- **AuthModule:** Manejo de autenticación, login, registro, recuperación de contraseña
- **LlamadoModule:** Consulta y visualización de llamados disponibles
- **InscripcionModule:** Workflow de inscripción multipaso
- **PostulanteModule:** Panel "Mis Inscripciones", seguimiento de estado
- **TribunalModule:** Dashboard, lista de inscripciones, evaluación, ordenamientos
- **RrhhModule:** Gestión de llamados (pendiente de implementación completa)
- **SharedModule:** Componentes reutilizables (tablas, filtros, modales, notificaciones)
- **CoreModule:** Servicios singleton (AuthService, HttpInterceptor, Guards)

Componentes Principales

Se desarrollaron más de 25 componentes reutilizables:

Componentes de Inscripción:

- `inscripcion-workflow.component`: Contenedor del workflow con stepper
- `paso-departamento.component`: Selección de departamento
- `paso-autodefinicion.component`: Autodefinición según leyes
- `paso-requisitos.component`: Declaración de requisitos excluyentes
- `paso-meritos.component`: Carga de méritos valorables
- `paso-apoyos.component`: Solicitud de apoyos (condicional)
- `paso-confirmacion.component`: Resumen y confirmación

Componentes de Tribunal:

- `dashboard-tribunal.component`: Dashboard con métricas
- `lista-inscripciones.component`: Tabla con filtros
- `detalle-postulante.component`: Contenedor con tabs
- `evaluacion-meritos.component`: Formulario de calificación de méritos
- `evaluacion-pruebas.component`: Formulario de calificación de pruebas
- `generacion-ordenamiento.component`: Interfaz para generar ordenamiento
- `visualizacion-ordenamiento.component`: Presentación de resultados

Componentes Compartidos:

- `data-table.component`: Tabla genérica con paginación y ordenamiento
- `filter-panel.component`: Panel de filtros dinámicos

- `file-upload.component`: Componente de carga de archivos
- `confirmation-modal.component`: Modal de confirmación genérico
- `toast-notification.component`: Notificaciones no intrusivas
- `loading-spinner.component`: Indicador de carga
- `stepper.component`: Indicador de progreso para workflows

Servicios

Se implementaron 12 servicios que encapsulan lógica de negocio del cliente:

- **AuthService**: Autenticación, gestión de sesión
- **LlamadoService**: Consulta de llamados, obtención de detalle
- **InscripcionService**: Crear inscripción, consultar mis inscripciones
- **TribunalService**: Obtener inscripciones, calificar, generar ordenamiento
- **DepartamentoService**: Consultar departamentos disponibles
- **ConstanciaService**: Subir y descargar archivos
- **ValidationService**: Validaciones comunes del lado del cliente
- **NotificationService**: Gestión de mensajes toast
- **LoadingService**: Control del spinner de carga global
- **CacheService**: Caché en memoria para optimizar peticiones repetidas
- **ErrorHandlerService**: Manejo centralizado de errores HTTP
- **ExportService**: Exportación a Excel/PDF (del lado del cliente)

Routing y Lazy Loading

La configuración de rutas implementa lazy loading para optimizar tiempos de carga inicial:

```
1 const routes: Routes = [
2   { path: '', redirectTo: '/home', pathMatch: 'full' },
3   { path: 'home', component: HomeComponent },
4   {
5     path: 'auth',
6     loadChildren: () => import('./auth/auth.module').then(m => m.AuthModule)
7   },
8   {
9     path: 'llamados',
10    loadChildren: () => import('./llamado/llamado.module').then(m =>
11      m.LlamadoModule)
12  },
13  {
14    path: 'inscripcion',
15    loadChildren: () => import('./inscripcion/inscripcion.module').then(m =>
16      m.InscripcionModule),
17    canActivate: [AuthGuard]
18  },
19  {
20    path: 'postulante',
21    loadChildren: () => import('./postulante/postulante.module').then(m =>
22      m.PostulanteModule),
23    canActivate: [AuthGuard, PostulanteGuard]
24  },
25  {
26    path: 'tribunal',
```

```

24     loadChildren: () => import('./tribunal/tribunal.module').then(m =>
25       m.TribunalModule),
26   canActivate: [AuthGuard, TribunalGuard]
27 }
27 ];

```

Guards de Autenticación

Se implementaron guards que protegen rutas según rol del usuario:

```

1  @Injectable({ providedIn: 'root' })
2  export class AuthGuard implements CanActivate {
3    constructor(
4      private authService: AuthService,
5      private router: Router
6    ) {}
7
8    canActivate(): boolean {
9      if (this.authService.isAuthenticated()) {
10        return true;
11      }
12      this.router.navigate(['/auth/login']);
13      return false;
14    }
15  }
16
17 @Injectable({ providedIn: 'root' })
18 export class TribunalGuard implements CanActivate {
19   constructor(
20     private authService: AuthService,
21     private router: Router

```

```

22     ) {}
23
24     canActivate(): boolean {
25         const user = this.authService.getCurrentUser();
26         if (user && user.rol === 'Tribunal') {
27             return true;
28         }
29         this.router.navigate(['/unauthorized']);
30         return false;
31     }
32 }

```

Interceptors HTTP

Se configuró interceptor HTTP que maneja errores de forma centralizada:

```

1  @Injectable()
2  export class HttpErrorInterceptor implements HttpInterceptor {
3      constructor(
4          private notificationService: NotificationService,
5          private authService: AuthService
6      ) {}
7
8      intercept(req: HttpRequest<any>, next: HttpHandler):
9          Observable<HttpEvent<any>> {
10         return next.handle(req).pipe(
11             catchError((error: HttpErrorResponse) => {
12                 if (error.status === 401) {
13                     this.authService.logout();
14                     this.notificationService.error('Sesión expirada. Ingrese
15                     nuevamente.');
16                 }
17             })
18         );
19     }
20 }

```

```

14     } else if (error.status === 403) {
15
16         this.notificationService.error('No tiene permisos para realizar esta
17             → acción.');
18
19     } else if (error.status === 404) {
20
21         this.notificationService.error('Recurso no encontrado.');
22
23     } else if (error.status >= 500) {
24
25         this.notificationService.error('Error del servidor. Intente
26             → nuevamente más tarde.');
27
28     }
29
30     return throwError(() => error);
31
32 )
33
34 }
35
36 }

```

Capa de Lógica de Negocio (Backend - ASP.NET Core 8)

Controllers

La API RESTful se estructura en 6 controllers que exponen 37 endpoints:

LlamadoController:

- GET /api/llamados - Obtener llamados activos (con filtros opcionales)
- GET /api/llamados/{id} - Obtener detalle de llamado específico
- GET /api/llamados/{id}/requisitos - Obtener requisitos excluyentes
- GET /api/llamados/{id}/meritos - Obtener ítems puntuables

InscripcionController:

- POST /api/inscripciones - Crear nueva inscripción
- GET /api/inscripciones/{id} - Obtener detalle de inscripción
- GET /api/inscripciones/postulante/{postulanteId} - Mis inscripciones
- PUT /api/inscripciones/{id}/estado - Actualizar estado
- POST /api/inscripciones/{id}/meritos - Agregar mérito
- POST /api/inscripciones/{id}/apoyos - Agregar apoyo solicitado

TribunalController:

- GET /api/tribunal/inscripciones - Lista de inscripciones con filtros
- GET /api/tribunal/inscripciones/{id} - Detalle para evaluación
- GET /api/tribunal/estadisticas - Métricas del dashboard
- POST /api/tribunal/evaluaciones/meritos - Calificar mérito
- PUT /api/tribunal/evaluaciones/meritos/{id} - Actualizar calificación
- POST /api/tribunal/evaluaciones/pruebas - Calificar prueba
- PUT /api/tribunal/evaluaciones/pruebas/{id} - Actualizar calificación
- POST /api/tribunal/ordenamientos - Generar ordenamiento
- GET /api/tribunal/ordenamientos/{id} - Obtener ordenamiento

ConstanciaController:

- POST /api/constancias - Subir archivo
- GET /api/constancias/{id} - Descargar archivo

- DELETE /api/constancias/{id} - Eliminar archivo

DepartamentoController:

- GET /api/departamentos - Obtener todos los departamentos
- GET /api/llamados/{llamadoId}/departamentos - Departamentos disponibles para llamado

ExportController:

- GET /api/export/ordenamiento/{id}/excel - Exportar a Excel
- GET /api/export/ordenamiento/{id}/pdf - Exportar a PDF

Servicios de Lógica de Negocio

Se implementaron 9 servicios con responsabilidades bien delimitadas:

LlamadoService:

- Consultar llamados activos con filtros
- Obtener detalle completo de llamado
- Validar si un llamado está abierto para inscripciones
- Obtener requisitos excluyentes configurados
- Obtener ítems puntuables con puntajes máximos

InscripcionService:

- Crear inscripción validando reglas de negocio
- Validar que no exista inscripción duplicada
- Generar código único de inscripción
- Actualizar estado de inscripción
- Obtener inscripciones de un postulante
- Agregar méritos a inscripción existente
- Agregar apoyos solicitados

TribunalService:

- Obtener inscripciones con filtros avanzados
- Calcular métricas del dashboard
- Calificar méritos con validación de puntajes
- Calificar pruebas con validación de umbrales
- Calcular puntaje total de inscripción
- Determinar si inscripción aprueba todas las pruebas

OrdenamientoService:

- Generar ordenamiento aplicando algoritmo de cuotas
- Calcular cupos por ley (8 % afro, 1 % trans, 4 % discapacidad)
- Aplicar criterios de desempate
- Identificar posiciones cubiertas vs. suplentes

- Generar ordenamientos por cupo separados

ValidacionService:

- Validar formato de cédula uruguaya
- Validar formato de email
- Validar rangos de puntajes
- Validar requisitos excluyentes
- Validar coherencia de datos de inscripción

ArchivoService:

- Almacenar archivos en file system (temporal, para MVP)
- Validar tipo MIME y tamaño de archivo
- Generar nombres únicos para archivos
- Recuperar archivos por ID
- Eliminar archivos huérfanos

PostulanteService:

- Obtener perfil de postulante
- Actualizar datos personales
- Listar inscripciones activas
- Obtener historial de inscripciones

ConstanciaService:

- Vincular constancia con mérito
- Validar constancia (PDF, tamaño máximo)
- Generar metadatos de archivo

DepartamentoService:

- Obtener todos los departamentos
- Obtener departamentos disponibles para llamado
- Validar que departamento seleccionado está disponible

Inyección de Dependencias

Los servicios se registran en el contenedor de DI en Program.cs:

```
1 // Servicios de lógica de negocio
2 builder.Services.AddScoped<ILlamadoService, LlamadoService>();
3 builder.Services.AddScoped<IIInscripcionService, InscripcionService>();
4 builder.Services.AddScoped<ITribunalService, TribunalService>();
5 builder.Services.AddScoped<IOrdenamientoService, OrdenamientoService>();
6 builder.Services.AddScoped<IValidacionService, ValidacionService>();
7 builder.Services.AddScoped<IArchivoService, ArchivoService>();
8 builder.Services.AddScoped<IPostulanteService, PostulanteService>();
9 builder.Services.AddScoped<IConstanciaService, ConstanciaService>();
10 builder.Services.AddScoped<IDepartamentoService, DepartamentoService>();
11
12 // Repositorios
```

```

13 builder.Services.AddScoped<ILlamadoRepository, LlamadoRepository>();
14 builder.Services.AddScoped<IIInscripcionRepository, InscripcionRepository>();
15 builder.Services.AddScoped<IPostulanteRepository, PostulanteRepository>();
16 builder.Services.AddScoped<ITribunalRepository, TribunalRepository>();
17 builder.Services.AddScoped<IOrdenamientoRepository, OrdenamientoRepository>();
18 builder.Services.AddScoped<IDepartamentoRepository, DepartamentoRepository>();

19
20 // Unit of Work
21 builder.Services.AddScoped<IUnitOfWork, UnitOfWork>();

```

DTOs (Data Transfer Objects)

Para separar el modelo de dominio de los contratos de API, se definieron DTOs específicos para cada endpoint:

```

1 // DTO para consulta de llamados
2 public class LlamadoListaDto
3 {
4     public int Id { get; set; }
5     public string Titulo { get; set; }
6     public DateTime FechaApertura { get; set; }
7     public DateTime FechaCierre { get; set; }
8     public int PlazasDisponibles { get; set; }
9     public List<string> Departamentos { get; set; }
10 }
11
12 // DTO para detalle de llamado
13 public class LlamadoDetalleDto
14 {
15     public int Id { get; set; }
16     public string Titulo { get; set; }

```

```

17     public string Descripcion { get; set; }
18     public DateTime FechaApertura { get; set; }
19     public DateTime FechaCierre { get; set; }
20     public int PlazasDisponibles { get; set; }
21     public List<RequisitoExcluyenteDto> RequisitosExcluyentes { get; set; }
22     public List<ItemPuntuableDto> ItemsPuntuables { get; set; }
23     public List<PruebaDto> Pruebas { get; set; }
24     public CronogramaDto Cronograma { get; set; }
25     public CuposDto Cupos { get; set; }
26 }
27
28 // DTO para crear inscripción
29 public class CrearInscripcionDto
30 {
31     public int LlamadoId { get; set; }
32     public int PostulanteId { get; set; }
33     public int DepartamentoId { get; set; }
34     public AutodefinicionDto Autodefinicion { get; set; }
35     public List<RequisitoPostulanteDto> Requisitos { get; set; }
36     public List<MeritoPostulanteDto> Meritos { get; set; }
37     public List<ApoyoSolicitadoDto> Apoyos { get; set; }
38 }

```

Capa de Acceso a Datos (Entity Framework Core 8 + SQL Server)

Patrón Repository

Se implementó el patrón Repository para cada agregado del dominio:

```

1 public interface ILLamadoRepository : IRepository<Llamado>
2 {

```

```

3     Task<List<Llamado>> ObtenerActivosAsync();
4     Task<Llamado> ObtenerConDetalleAsync(int id);
5     Task<List<LlamadoDepartamento>> ObtenerDepartamentosAsync(int llamadoId);
6 }
7
8 public class LlamadoRepository : Repository<Llamado>, ILlamadoRepository
9 {
10    public LlamadoRepository(ApplicationDbContext context) : base(context) { }
11
12    public async Task<List<Llamado>> ObtenerActivosAsync()
13    {
14        return await _context.Llamados
15            .Where(l => l.Estado == EstadoLlamado.Activo &&
16                    l.FechaCierre >= DateTime.Now)
17            .Include(l => l.LlamadoDepartamentos)
18            .ToListAsync();
19    }
20
21    public async Task<Llamado> ObtenerConDetalleAsync(int id)
22    {
23        return await _context.Llamados
24            .Include(l => l.RequisitosExcluyentes)
25            .Include(l => l.ItemsPuntuables)
26            .Include(l => l.Pruebas)
27            .Include(l => l.LlamadoDepartamentos)
28            .ThenInclude(ld => ld.Departamento)
29            .FirstOrDefaultAsync(l => l.Id == id);
30    }
31
32    public async Task<List<LlamadoDepartamento>> ObtenerDepartamentosAsync(int
33        ↪ llamadoId)

```

```

33     {
34         return await _context.LlamadoDepartamentos
35             .Where(ld => ld.LlamadoId == llamadoId)
36             .Include(ld => ld.Departamento)
37             .ToListAsync();
38     }
39 }
```

Patrón Unit of Work

El patrón Unit of Work coordina transacciones que involucran múltiples repositorios:

```

1  public interface IUnitOfWork : IDisposable
2  {
3      ILlamadoRepository Llamados { get; }
4      IIInscripcionRepository Inscripciones { get; }
5      IPostulanteRepository Postulantes { get; }
6      ITribunalRepository Tribunal { get; }
7      IOordenamientoRepository Ordenamientos { get; }
8      IDepartamentoRepository Departamentos { get; }
9
10     Task<int> CommitAsync();
11     Task RollbackAsync();
12 }
13
14 public class UnitOfWork : IUnitOfWork
15 {
16     private readonly ApplicationDbContext _context;
17
18     public ILlamadoRepository Llamados { get; }
19     public IIInscripcionRepository Inscripciones { get; }
```

```

20     public IPostulanteRepository Postulantes { get; }
21     // ... otros repositorios
22
23     public UnitOfWork(ApplicationDbContext context)
24     {
25         _context = context;
26         Llamados = new LlamadoRepository(context);
27         Inscripciones = new InscripcionRepository(context);
28         Postulantes = new PostulanteRepository(context);
29         // ... otros repositorios
30     }
31
32     public async Task<int> CommitAsync()
33     {
34         return await _context.SaveChangesAsync();
35     }
36
37     public async Task RollbackAsync()
38     {
39         await _context.Database.RollbackTransactionAsync();
40     }
41
42     public void Dispose()
43     {
44         _context.Dispose();
45     }
46 }
```

Uso en servicios:

```
1  public class InscripcionService : IIInscripcionService
```

```

2  {
3      private readonly IUnitOfWork _unitOfWork;
4
5      public InscripcionService(IUnitOfWork unitOfWork)
6      {
7          _unitOfWork = unitOfWork;
8      }
9
10     public async Task<Inscripcion> CrearInscripcionAsync(CrearInscripcionDto
11         ↪ dto)
12     {
13         // Operación transaccional que involucra múltiples entidades
14         var inscripcion = new Inscripcion { ... };
15         await _unitOfWork.Inscripciones.AddAsync(inscripcion);
16
17         foreach (var merito in dto.Meritos)
18         {
19             var meritoEntity = new MeritoPostulante { InscripcionId =
20                 ↪ inscripcion.Id, ... };
21             await _unitOfWork.Inscripciones.AddMeritoAsync(meritoEntity);
22         }
23
24         await _unitOfWork.CommitAsync(); // Commit atómico
25         return inscripcion;
26     }
27 }
```

Migraciones de Base de Datos

Entity Framework Core gestiona la evolución del esquema mediante migraciones versionadas:

```

# Crear nueva migración
dotnet ef migrations add NombreMigracion

# Aplicar migraciones pendientes
dotnet ef database update

# Revertir a migración específica
dotnet ef database update NombreMigracionAnterior

```

Ejemplo de migración generada:

```

1 public partial class AgregarOrdenamientos : Migration
2 {
3     protected override void Up(MigrationBuilder migrationBuilder)
4     {
5         migrationBuilder.CreateTable(
6             name: "Ordenamientos",
7             columns: table => new
8             {
9                 Id = table.Column<int>(nullable: false)
10                .Annotation("SqlServer:Identity", "1, 1"),
11                 LlamadoId = table.Column<int>(nullable: false),
12                 FechaGeneracion = table.Column<DateTime>(nullable: false),
13                 GeneradoPor = table.Column<int>(nullable: false)
14             },
15             constraints: table =>
16             {
17                 table.PrimaryKey("PK_Ordenamientos", x => x.Id);
18                 table.ForeignKey(
19                     name: "FK_Ordenamientos_Llamados_LlamadoId",
20                     column: x => x.LlamadoId,

```

```

21             principalTable: "Llamados",
22             principalColumn: "Id",
23             onDelete: ReferentialAction.Cascade);
24         });
25     }
26
27     protected override void Down(MigrationBuilder migrationBuilder)
28     {
29         migrationBuilder.DropTable(name: "Ordenamientos");
30     }
31 }
```

Modelo de Datos SQL Server

Entidades Principales

El modelo de base de datos consta de 21 tablas que representan las entidades del dominio:

Tablas del Agregado Llamado:

- Llamados - Información general del llamado
- LlamadoDepartamentos - Departamentos disponibles por llamado
- RequisitosExcluyentes - Requisitos obligatorios
- ItemsPuntuables - Méritos valorables con puntajes
- ApoyosNecesarios - Catálogo de apoyos disponibles
- Pruebas - Pruebas a rendir (escritas, prácticas, orales)

Tablas del Agregado Inscripción:

- **Inscripciones** - Postulación completa
- **RequisitosPostulante** - Requisitos declarados cumplidos
- **MeritosPostulante** - Méritos declarados
- **ApoyosSolicitados** - Apoyos específicos solicitados
- **Constancias** - Archivos probatorios adjuntos

Tablas del Agregado Evaluación:

- **EvaluacionesMeritos** - Calificaciones de méritos
- **EvaluacionesPruebas** - Resultados de pruebas

Tablas del Agregado Ordenamiento:

- **Ordenamientos** - Metadata del ordenamiento generado
- **PosicionesOrdenamiento** - Posiciones individuales

Tablas de Soporte:

- **Postulantes** - Información de usuarios postulantes
- **Tribunales** - Información de usuarios evaluadores
- **Departamentos** - Catálogo de departamentos del Uruguay
- **AutodefinicionesLey** - Registro de autodefiniciones
- **Usuarios** - Información de autenticación (pendiente de implementación completa)

Índices Optimizados

Se definieron índices en columnas de búsqueda frecuente para optimizar consultas:

```
1  -- Índice para búsqueda de llamados activos
2  CREATE INDEX IX_Llamados_Estado_FechaCierre
3  ON Llamados(Estado, FechaCierre);
4
5  -- Índice para búsqueda de inscripciones por postulante
6  CREATE INDEX IX_Inscripciones_PostulanteId
7  ON Inscripciones(PostulanteId);
8
9  -- Índice para filtrado de inscripciones por llamado y estado
10 CREATE INDEX IX_Inscripciones_LlamadoId_Estado
11 ON Inscripciones(LlamadoId, Estado);
12
13 -- Índice para búsqueda por cédula de postulante
14 CREATE INDEX IX_Postulantes_Cedula
15 ON Postulantes(Cedula);
```

Constraints de Integridad Referencial

Todas las relaciones entre tablas están protegidas mediante foreign keys con políticas de cascada configuradas:

```
1  -- Foreign key de Inscripcion a Llamado (Cascade Delete)
2  ALTER TABLE Inscripciones
3  ADD CONSTRAINT FK_Inscripciones_Llamados
4  FOREIGN KEY (LlamadoId) REFERENCES Llamados(Id)
5  ON DELETE CASCADE;
6
```

```

7  -- Foreign key de Inscripcion a Postulante (Restrict Delete)
8  ALTER TABLE Inscripciones
9  ADD CONSTRAINT FK_Inscripciones_Postulantes
10 FOREIGN KEY (PostulanteId) REFERENCES Postulantes(Id)
11 ON DELETE RESTRICT;
12
13 -- Foreign key de MeritoPostulante a Inscripcion (Cascade Delete)
14 ALTER TABLE MeritosPostulante
15 ADD CONSTRAINT FK_MeritosPostulante_Inscripciones
16 FOREIGN KEY (InscripcionId) REFERENCES Inscripciones(Id)
17 ON DELETE CASCADE;

```

La política de cascada se eligió según las reglas del dominio:

- **Cascade Delete:** Se usa cuando la entidad hija no tiene sentido sin la madre (ej: méritos sin inscripción)
- **Restrict Delete:** Se usa cuando la entidad madre no debe eliminarse si tiene hijos (ej: postulante con inscripciones)

Conclusión

La arquitectura implementada demuestra madurez técnica y adherencia a mejores prácticas de Ingeniería de Software. La separación clara de responsabilidades en tres capas, el uso de patrones arquitectónicos consolidados (Repository, Unit of Work, Dependency Injection) [38], y la aplicación de principios SOLID [114] proporcionan una base sólida para la evolución sostenible del sistema hacia una plataforma completa de producción.

Apéndice J: Cobertura Detallada de Pruebas Unitarias

Estrategia de Testing

Las pruebas unitarias se implementaron siguiendo la pirámide de testing [68], priorizando las pruebas de la capa de lógica de negocio, donde reside la complejidad del dominio. Se utilizó el framework xUnit [69] por su simplicidad, extensibilidad, y adopción consolidada en la comunidad .NET.

Para facilitar el testing en aislamiento, se utilizó la librería Moq que permite crear mocks de dependencias sin implementar clases de prueba manualmente. Esta técnica permite probar cada servicio de forma independiente, sin requerir una base de datos real ni otros servicios externos.

La Figura J.1 muestra los resultados de la suite completa de pruebas ejecutadas en Visual Studio, incluyendo 192 pruebas superadas exitosamente entre pruebas unitarias e integración.

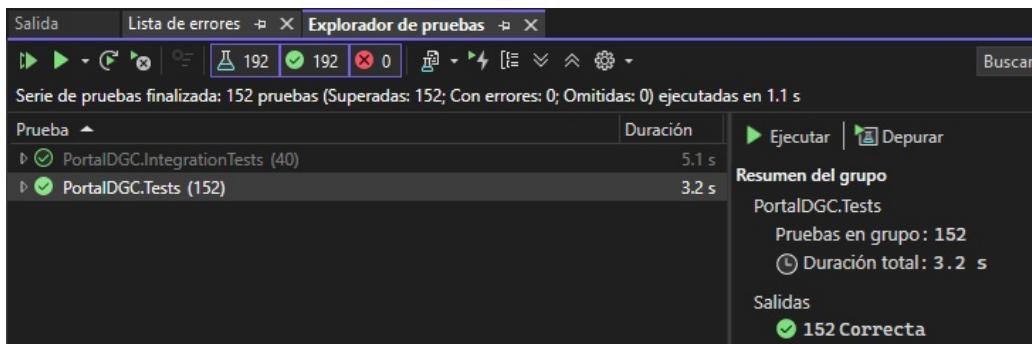


Figura J.1: Resultados de la suite de pruebas unitarias e integración realizadas en Visual Studio.

Cobertura por Servicio

TribunalService (17 tests, 89 % cobertura)

El servicio TribunalService gestiona la evaluación de inscripciones, calificación de pruebas y méritos, y generación de ordenamientos finales. Los tests cubren los escenarios críticos de negocio del módulo de tribunal. Como puede observarse en la Figura J.2 los 17 tests de TribunalService se ejecutan correctamente, alcanzando un 89 % de cobertura.

Prueba	Duración
✓ TribunalServiceTests (17)	1.1 s
✓ CalificarPruebaAsync_PuntajeFueraDeRango_CancelaTransaccion	63 ms
✓ CalificarPruebaAsync_SinEvaluacionPrevio_CreaYConfirma	23 ms
✓ GenerarOrdenamientoAsync_InscripcionesValidas_GeneraOrdenamiento	63 ms
✓ GenerarOrdenamientoAsync_SinInscripcionesValidas_RetornaFallo	22 ms
✓ ObtenerDetalleEvaluacionAsync_NoExiste_RetornaFallo	13 ms
✓ ObtenerDetalleOrdenamientoAsync_MapeaPosiciones	36 ms
✓ ObtenerDetalleOrdenamientoAsync_NoExiste_RetornaFallo	3 ms
✓ ObtenerEstadisticasAsync_CalculaIndicadores	34 ms
✓ ObtenerEstadisticasAsync_LlamadoNoExiste_RetornaFallo	4 ms
✓ ObtenerInscripcionesParaEvaluarAsync_MapeaPuntajesYTotales	682 ms
✓ ObtenerOrdenamientosAsync_MapeaCantidadPosiciones	26 ms
✓ ObtenerPruebasDelLlamadoAsync_MapeaEstadisticas	31 ms
✓ PublicarOrdenamientoAsync_ActualizaEstado	21 ms
✓ PublicarOrdenamientoAsync_NoExiste_RetornaFallo	3 ms
✓ ValorarMeritoAsync_ActualizaEvaluacionExistente	23 ms
✓ ValorarMeritoAsync_PuntajeInvalido_RetornaFallo	19 ms
✓ ValorarMeritosAsync_MezclaValida_SoloProcesaPermitidos	36 ms

Figura J.2: Ejecución de los 17 tests de TribunalService.

Categorías de tests:

Tests de Obtención de Información (4 tests)

1. `ObtenerInscripcionesParaEvaluarAsync_MapeaPuntajesYTotales`: Verifica que se obtienen correctamente las inscripciones de un llamado con el cálculo de puntajes totales, sumando pruebas verificadas aprobadas y méritos aprobados.
2. `ObtenerDetalleEvaluacionAsync_NoExiste_RetornaFallo`: Valida el manejo de error cuando se intenta obtener detalle de una inscripción inexistente.
3. `ObtenerPruebasDelllamoadoAsync_MapeaEstadisticas`: Verifica que se calculan correctamente las estadísticas de cada prueba del llamado (cantidad evaluados, aprobados, y promedio general).
4. `ObtenerEstadisticasAsync_CalculaIndicadores`: Valida el cálculo integral de estadísticas del llamado, incluyendo total de inscripciones, distribución por cuotas (afrodescendientes, trans, discapacidad), promedio general de puntajes, y estado de generación de ordenamiento.

Tests de Calificación de Pruebas (2 tests)

1. `CalificarPruebaAsync_PuntajeFueraDeRango_CancelaTransaccion`: Valida que el sistema rechaza puntajes que exceden el máximo permitido para la prueba y ejecuta rollback de la transacción.
2. `CalificarPruebaAsync_SinEvaluacionPrevio_CreaYConfirma`: Verifica la creación exitosa de una nueva evaluación de prueba, marcándola como aprobada cuando supera el umbral mínimo, y confirma la transacción.

Tests de Valoración de Méritos (3 tests)

1. `ValorarMeritoAsync_ActualizaEvaluacionExistente`: Valida que se puede modificar una valoración de mérito previamente asignada, actualizando puntaje, estado y observaciones.
2. `ValorarMeritoAsync_PuntajeInvalido_RetornaFallo`: Verifica que el sistema rechaza puntajes que exceden el máximo del ítem puntuable y ejecuta rollback.
3. `ValorarMeritosAsync_MezclaValida_SoloProcesaPermitidos`: Valida el procesamiento batch de múltiples méritos, donde solo se procesan los válidos y se ignoran aquellos con errores (mérito inexistente), sin interrumpir la operación completa.

Tests de Gestión de Ordenamientos (6 tests)

1. `ObtenerOrdenamientosAsync_MapeaCantidadPosiciones`: Verifica que se obtienen correctamente los ordenamientos de un llamado con el conteo de posiciones y datos del llamado y departamento asociados.
2. `ObtenerDetalleOrdenamientoAsync_NoExiste_RetornaFallo`: Valida el manejo de error cuando se intenta obtener detalle de un ordenamiento inexistente.
3. `ObtenerDetalleOrdenamientoAsync_MapeaPosiciones`: Verifica el mapeo completo del detalle de un ordenamiento, incluyendo todas sus posiciones con cálculo de puntajes de pruebas y méritos para cada postulante.
4. `GenerarOrdenamientoAsync_SinInscripcionesValidas_RetornaFallo`: Valida que el sistema no genera ordenamiento cuando ninguna inscripción cumple con el puntaje mínimo de aprobación, ejecutando rollback de la transacción.
5. `GenerarOrdenamientoAsync_InscripcionesValidas_GeneraOrdenamiento`: Verifica la generación exitosa de ordenamiento general cuando hay inscripciones que superan

el puntaje mínimo, creando el ordenamiento con sus posiciones ordenadas por puntaje descendente y confirmando la transacción.

6. `PublicarOrdenamientoAsync_ActualizaEstado`: Valida que se actualiza correctamente el estado de un ordenamiento de "Preliminar.^a" "Publicado".

Tests de Validación de Errores (2 tests)

1. `ObtenerEstadisticasAsync_LlamadoNoExiste_RetornaFallo`: Verifica el manejo de error cuando se solicitan estadísticas de un llamado inexistente.
2. `PublicarOrdenamientoAsync_NoExiste_RetornaFallo`: Valida el manejo de error cuando se intenta publicar un ordenamiento inexistente.

Cobertura y alcance: Los 17 tests implementados priorizan la verificación de escenarios críticos del flujo de evaluación y generación de ordenamientos, logrando 89 % de cobertura en el servicio. Se enfocaron en validar la correcta integración entre repositorios, el manejo transaccional de operaciones complejas, y el cálculo de puntajes que determinan el ordenamiento final de postulantes.

IncripcionService (4 tests, 87 % cobertura)

El servicio IncripcionService gestiona el proceso completo de inscripción de postulantes a llamados, incluyendo la validación de elegibilidad, persistencia transaccional de datos, y verificación de requisitos. Los tests cubren los escenarios fundamentales de validación y creación. Como se muestra en la Figura J.3, los 4 tests de IncripcionService se ejecutan correctamente, alcanzando un 87 % de cobertura.

▲ ✓ IncripcionServiceTests (4)	343 ms
✓ CrearIncripcion_DatosValidos_RetornaSuccess	44 ms
✓ CrearIncripcion_LlamadoCerrado_RetornaError	16 ms
✓ CrearIncripcion_PostulanteNoExiste_RetornaError	4 ms
✓ ValidarIncripcionExistente_YaExiste_RetornaTrue	279 ms

Figura J.3: Ejecución de los 4 tests de IncripcionService.

Categorías de tests:

Tests de Creación de Inscripción (3 tests)

1. `CrearIncripcion_DatosValidos_RetornaSuccess`: Verifica el flujo completo de creación exitosa de una inscripción con todos los datos válidos. Valida que se ejecuta la transacción completa (`begin, commit`), se persisten correctamente la autodefinición de ley, requisitos, méritos y apoyos solicitados, y se retorna la inscripción completa creada.
2. `CrearIncripcion_PostulanteNoExiste_RetornaError`: Valida el manejo de error cuando se intenta crear una inscripción para un postulante不存在 en la base de datos, retornando mensaje descriptivo del error.
3. `CrearIncripcion_LlamadoCerrado_RetornaError`: Verifica que el sistema rechaza inscripciones a llamados que no están en estado `.^abierto.^` cuya fecha de cierre ya pasó, retornando mensaje indicando que el llamado no está disponible.

Tests de Validación (1 test)

1. `ValidarInscripcionExistente_YaExiste_ReturnaTrue`: Valida que el método de verificación de inscripción existente funciona correctamente, retornando true cuando ya existe una inscripción del mismo postulante para el mismo llamado.

Cobertura y alcance: Los 4 tests implementados priorizan la validación del flujo crítico de creación de inscripciones y las validaciones fundamentales de negocio que previenen datos inconsistentes. La cobertura de 87 % se logra mediante tests que verifican la correcta integración entre múltiples repositorios (Postulante, Llamado, Inscripcion, Departamento, AutodefinicionLey, RequisitosPostulante, MeritosPostulante, ApoyosSolicitados) y el manejo transaccional que garantiza atomicidad en operaciones complejas que involucran múltiples entidades relacionadas.

Validaciones de negocio cubiertas:

- Verificación de existencia de postulante antes de crear inscripción
- Validación de estado del llamado (debe estar abierto)
- Prevención de inscripciones duplicadas para el mismo postulante y llamado
- Manejo transaccional completo con commit/rollback según resultado

ValidacionService (26 tests, 91 % cobertura)

El servicio ValidacionService centraliza todas las validaciones de formato de datos y reglas de negocio del sistema. Implementa validaciones tanto síncronas (formatos) como asíncronas (verificaciones con base de datos). Como puede observarse en la Figura J.4, los 26 tests de ValidacionService se ejecutan correctamente, alcanzando un 91 % de cobertura.

Test	Tiempo de ejecución
✓ ValidacionServiceTests (26)	261 ms
✓ ValidarCedulaIdentidad_VariosFormatos_RetornaResultadoEsperado	196 ms
✓ ValidarCedulaIdentidad_VariosFormatos_RetornaResultadoEsperado	195 ms
✓ ValidarCedulaIdentidad_VariosFormatos_RetornaResultadoEsperado	< 1 ms
✓ ValidarCedulaIdentidad_VariosFormatos_RetornaResultadoEsperado	1 ms
✓ ValidarEdadMinima_Mayor18_RetornaTrue	< 1 ms
✓ ValidarEdadMinima_Menor18_RetornaFalse	1 ms
✓ ValidarEmail_VariosFormatos_RetornaResultadoEsperado (5)	2 ms
✓ ValidarEmail_VariosFormatos_RetornaResultadoEsperado(email:...)	< 1 ms
✓ ValidarEmail_VariosFormatos_RetornaResultadoEsperado(email:...)	2 ms
✓ ValidarEmail_VariosFormatos_RetornaResultadoEsperado(email:...)	< 1 ms
✓ ValidarEmail_VariosFormatos_RetornaResultadoEsperado(email:...)	< 1 ms
✓ ValidarEmail_VariosFormatos_RetornaResultadoEsperado(email:...)	< 1 ms
✓ ValidarFechaRango_DentroDelRango_RetornaTrue	< 1 ms
✓ ValidarFechaRango_FechaEnLímiteSuperior_RetornaTrue	1 ms
✓ ValidarFechaRango_FueraDelRango_RetornaFalse	1 ms
✓ ValidarLlamadoAbierto_Disponible_RetornaTrue	15 ms
✓ ValidarLlamadoAbierto_Error_RetornaFallo	31 ms
✓ ValidarPostulanteCompletoDatos_Completo_RetornaTrue	3 ms
✓ ValidarPostulanteCompletoDatos_Incompleto_RetornaFalse	4 ms
✓ ValidarPostulanteCompletoDatos_NoExiste_RetornaFallo	6 ms
✓ ValidarTelefono_VariosFormatos_RetornaResultadoEsperado (5)	1 ms
✓ ValidarTelefono_VariosFormatos_RetornaResultadoEsperado(tel:...)	< 1 ms
✓ ValidarTelefono_VariosFormatos_RetornaResultadoEsperado(tel:...)	< 1 ms
✓ ValidarTelefono_VariosFormatos_RetornaResultadoEsperado(tel:...)	1 ms
✓ ValidarTelefono_VariosFormatos_RetornaResultadoEsperado(tel:...)	< 1 ms
✓ ValidarTelefono_VariosFormatos_RetornaResultadoEsperado(tel:...)	< 1 ms

Figura J.4: Ejecución de los tests de ValidacionService.

Categorías de tests:

Tests de Validación de Cédula de Identidad (6 tests)

Implementados mediante [Theory] con [InlineData] para probar múltiples casos con un solo método de test:

1. `ValidarCedulaIdentidad_VariosFormatos_RetornaResultadoEsperado`: Test parametrizado que valida 6 casos diferentes de cédulas uruguayas:

- Formato con puntos y guion: "1.234.567-8" → Válido
- Formato sin separadores: "12345678" → Válido
- Formato de 7 dígitos: "1234567" → Válido
- Formato con caracteres no numéricos: ".bc.def.ghi-j" → Inválido
- Cédula vacía: → Inválido
- Cédula muy corta: "123" → Inválido

Tests de Validación de Email (5 tests)

1. `ValidarEmail_VariosFormatos_RetornaResultadoEsperado`: Test parametrizado que valida 5 casos de emails:

- Email válido simple: "test@example.com" → Válido
- Email con subdominios: user.name@domain.co.uk" → Válido
- Email sin dominio: invalid.email" → Inválido
- Email sin usuario: "@example.com" → Inválido
- Email vacío: → Inválido

Tests de Validación de Teléfono (5 tests)

1. `ValidarTelefono_VariosFormatos_RetornaResultadoEsperado`: Test parametrizado que valida 5 casos de teléfonos uruguayos:
 - Celular sin prefijo: "099123456" → Válido
 - Fijo sin prefijo: "24012345" → Inválido (requiere código de área)
 - Fijo con prefijo: "024012345" → Válido
 - Número inválido: "123456789" → Inválido
 - Teléfono vacío: → Válido (campo opcional)

Tests de Validación de Edad Mínima (2 tests)

1. `ValidarEdadMinima_Mayor18_RetornaTrue`: Verifica que una persona de 20 años cumple con el requisito de edad mínima de 18 años.
2. `ValidarEdadMinima_Menor18_RetornaFalse`: Valida que una persona de 15 años no cumple con el requisito de edad mínima de 18 años.

Tests de Validación de Fecha en Rango (3 tests)

1. `ValidarFechaRango_DentroDelRango_RetornaTrue`: Verifica que una fecha dentro del rango especificado es válida.
2. `ValidarFechaRango_FechaEnLimiteSuperior_RetornaTrue`: Valida el caso borde donde la fecha coincide exactamente con el límite superior del rango (boundary test).
3. `ValidarFechaRango_FueraDelRango_RetornaFalse`: Verifica que una fecha fuera del rango especificado es rechazada.

Tests de Validación de Llamado Abierto (2 tests)

1. `ValidarLlamadoAbierto_Disponible_RetornaTrue`: Verifica que se detecta correctamente cuando un llamado está disponible para inscripciones mediante consulta asíncrona al repositorio.
2. `ValidarLlamadoAbierto_Error_RetornaFallo`: Valida el manejo de excepciones cuando ocurre un error de base de datos durante la validación, retornando resultado de fallo con detalles del error.

Tests de Validación de Postulante con Datos Completos (3 tests)

1. `ValidarPostulanteCompletoDatos_NoExiste_RetornaFallo`: Verifica el manejo de error cuando se valida un postulante inexistente en la base de datos.
2. `ValidarPostulanteCompletoDatos_Incompleto_RetornaFalse`: Valida que se detecta correctamente cuando faltan datos obligatorios en el perfil del postulante (nombre completo, cédula, email, celular, domicilio).
3. `ValidarPostulanteCompletoDatos_Completo_RetornaTrue`: Verifica que se detecta correctamente cuando un postulante tiene todos los datos obligatorios completos para poder inscribirse a un llamado.

Cobertura y alcance: Los 26 tests implementados cubren validaciones críticas de formato de datos (cédula, email, teléfono) y reglas de negocio (edad mínima, disponibilidad de llamado, completitud de datos). La cobertura de 91 % refleja la alta criticidad de estas validaciones, que previenen la entrada de datos inconsistentes al sistema. El uso extensivo de tests parametrizados con [Theory] permite verificar múltiples casos de entrada con código conciso y mantenable.

PostulanteService (5 tests, 85 % cobertura)

El servicio PostulanteService gestiona el perfil del postulante y la actualización de sus datos personales. Los tests cubren las operaciones de consulta, actualización y validación de unicidad de datos. Como puede observarse en la Figura J.5, los 5 tests de PostulanteService se ejecutan correctamente, alcanzando un 85 % de cobertura.

PostulanteServiceTests (5)	Time
CompletarDatosPersonales_DatosValidos_RetornaSuccess	45 ms
ObtenerPostulantePorId_PostulanteExiste_RetornaSuccess	6 ms
ObtenerPostulantePorId_PostulanteNoExiste_RetornaError	11 ms
ValidarCedulaDisponible_CedulaDisponible_RetornaTrue	2 ms
ValidarCedulaDisponible_CedulaYaExiste_RetornaFalse	38 ms

Figura J.5: Ejecución de los 5 tests de PostulanteService.

Categorías de tests:

Tests de Obtención de Perfil (2 tests)

1. ObtenerPostulantePorId_PostulanteExiste_RetornaSuccess: Verifica la obtención exitosa de un postulante por su ID, validando que se mapean correctamente todos los datos personales (nombre, apellido, cédula, email).
2. ObtenerPostulantePorId_PostulanteNoExiste_RetornaError: Valida el manejo de error cuando se intenta obtener un postulante inexistente, retornando mensaje descriptivo del error.

Tests de Actualización de Datos Personales (1 test)

1. CompletarDatosPersonales_DatosValidos_RetornaSuccess: Verifica el flujo completo de actualización de datos personales, incluyendo:

- Validación de formato de cédula de identidad
- Validación de formato de email
- Validación de edad mínima (18 años)
- Actualización de todos los campos (nombre, apellido, fecha de nacimiento, género, domicilio, teléfono)
- Persistencia exitosa en base de datos
- Retorno del flag DatosCompletados en true

Tests de Validación de Unicidad de Cédula (2 tests)

1. `ValidarCedulaDisponible_CedulaDisponible_RetornaTrue`: Verifica que se detecta correctamente cuando una cédula no está registrada en el sistema y está disponible para registro.
2. `ValidarCedulaDisponible_CedulaYaExiste_RetornaFalse`: Valida que se detecta correctamente cuando una cédula ya está registrada en el sistema, previniendo duplicados.

Cobertura y alcance: Los 5 tests implementados priorizan las operaciones críticas del perfil de postulante: consulta de datos existentes, actualización completa de perfil con validaciones múltiples, y verificación de unicidad para prevenir registros duplicados. La cobertura de 85 % refleja que se validan los escenarios principales de interacción con el perfil del usuario, incluyendo tanto casos exitosos como manejo de errores.

Validaciones integradas en actualización de datos: El test de completar datos personales es particularmente robusto, verificando la integración con el servicio de validaciones para garantizar que:

- La cédula cumple con el formato y algoritmo de dígito verificador uruguayo

- El email cumple con el formato RFC 5322
- La edad del postulante cumple con el requisito mínimo de 18 años
- Todos los campos obligatorios están presentes antes de confirmar la actualización

LlamadoService (11 tests, 82 % cobertura)

El servicio LlamadoService gestiona la información de los llamados a concurso, incluyendo su consulta, validación de estado y obtención de componentes asociados (requisitos, ítems puntuables, apoyos necesarios). Como puede observarse en la Figura J.6, los 11 tests de LlamadoService se ejecutan correctamente, alcanzando un 82 % de cobertura.

▲ ✓ LlamadoServiceTests (11)	353 ms
✓ ObtenerApoyosNecesariosLlamadoAsync_MapeaCorrectamente	5 ms
✓ ObtenerItemsPuntuablesLlamadoAsync_MapeaCorrectamente	257 ms
✓ ObtenerLlamadoPorIdAsync_Error_RetornaMensajeDeError	3 ms
✓ ObtenerLlamadoPorIdAsync_LlamadoExiste_RetornaDetalleMape...	34 ms
✓ ObtenerLlamadoPorIdAsync_LlamadoNoExiste_RetornaNotFound	2 ms
✓ ObtenerLlamadosActivosAsync_Error_RetornaFallo	20 ms
✓ ObtenerLlamadosActivosAsync_LlamadosDisponibles_RetornaLista	7 ms
✓ ObtenerRequisitosLlamadoAsync_MapeaCorrectamente	5 ms
✓ ValidarLlamadoDisponibleAsync_Error_RetornaFallo	2 ms
▲ ✓ ValidarLlamadoDisponibleAsync_RetornaSegunRepositorio (2)	18 ms
✓ ValidarLlamadoDisponibleAsync_RetornaSegunRepositorio(repo...	1 ms
✓ ValidarLlamadoDisponibleAsync_RetornaSegunRepositorio(repo...	17 ms

Figura J.6: Ejecución de los 11 tests de LlamadoService.

Categorías de tests:

Tests de Obtención de Llamado por ID (3 tests)

1. `ObtenerLlamadoPorIdAsync_LlamadoExiste_RetornaDetalleMapeado`: Verifica la obtención exitosa del detalle completo de un llamado, incluyendo el mapeo correcto de todas sus colecciones relacionadas:
 - Departamentos asociados con cantidad de puestos por departamento
 - Requisitos excluyentes con sus tipos y obligatoriedad
 - Ítems puntuables con categorías y puntajes máximos
 - Apoyos necesarios disponibles para solicitar
2. `ObtenerLlamadoPorIdAsync_LlamadoNoExiste_RetornaNotFound`: Valida el manejo de error cuando se intenta obtener un llamado inexistente, retornando mensaje descriptivo.
3. `ObtenerLlamadoPorIdAsync_Error_RetornaMensajeDeError`: Verifica el manejo robusto de excepciones durante la consulta, capturando errores de base de datos y retornando resultado estructurado con detalles del error.

Tests de Obtención de Llamados Activos (2 tests)

1. `ObtenerLlamadosActivosAsync_LlamadosDisponibles_RetornaLista`: Verifica la obtención de listado de llamados en estado “abierto” disponibles para inscripción, mapeando correctamente título, descripción, fechas y estado.
2. `ObtenerLlamadosActivosAsync_Error_RetornaFallo`: Valida el manejo de excepciones durante la consulta de llamados activos, retornando resultado de fallo con mensaje descriptivo.

Tests de Validación de Disponibilidad (3 tests)

1. `ValidarLlamadoDisponibleAsync_RetornaSegunRepositorio`: Test parametrizado con [Theory] que valida 2 escenarios:
 - Llamado disponible (abierto) → retorna true con mensaje "disponible"
 - Llamado no disponible (cerrado) → retorna false con mensaje "no disponible"
2. `ValidarLlamadoDisponibleAsync_Error_RetornaFallo`: Verifica el manejo de excepciones durante la validación, retornando resultado de fallo con detalles del error.

Tests de Obtención de Componentes del Llamado (3 tests)

1. `ObtenerRequisitosLlamadoAsync_MapeaCorrectamente`: Verifica la obtención y mapeo correcto de los requisitos excluyentes asociados a un llamado, incluyendo descripción, tipo y obligatoriedad.
2. `ObtenerItemsPuntuablesLlamadoAsync_MapeaCorrectamente`: Valida la obtención y mapeo de los ítems puntuables (méritos) asociados a un llamado, incluyendo nombre, descripción, puntaje máximo y categoría.
3. `ObtenerApoyosNecesariosLlamadoAsync_MapeaCorrectamente`: Verifica la obtención y mapeo de los apoyos necesarios disponibles para el llamado, incluyendo descripción y tipo de apoyo.

Cobertura y alcance: Los 11 tests implementados cubren las operaciones principales del servicio de llamados: consulta individual con detalle completo, listado de llamados activos, validación de disponibilidad para inscripción, y obtención de todos los componentes necesarios para completar una inscripción. La cobertura de 82 % refleja la validación exhaustiva del mapeo de entidades complejas con múltiples relaciones.

Manejo de errores consistente: Los tests demuestran un patrón robusto de manejo de excepciones en todas las operaciones críticas, garantizando que errores de base de datos o consultas inválidas se transforman en respuestas estructuradas con mensajes descriptivos y detalles técnicos para debugging.

ArchivoService (14 tests)

El servicio ArchivoService gestiona todas las operaciones relacionadas con archivos físicos: validación de tipo y tamaño, almacenamiento, recuperación, conversión de Base64, y eliminación. Este servicio no tiene dependencias de repositorios y opera directamente con el sistema de archivos. Como puede observarse en la Figura J.7, los 14 tests de ArchivoService se ejecutan correctamente, validando tanto las operaciones de almacenamiento y recuperación de archivos como las reglas de validación de tamaño y tipo.

▲	✓ ArchivoServiceTests (14)	154 ms
	✓ ConvertirBase64AArchivoAsync_EstadalInvalida_RetornaFallo	3 ms
	✓ EliminarArchivoAsync_ArchivoPresente_LoBorra	24 ms
	✓ GuardarArchivoAsync_CreaArchivoFisico	11 ms
	✓ ObtenerArchivoAsync_ArchivoExiste_RetornaContenido	112 ms
	✓ ObtenerArchivoAsync_ArchivoNoExiste_RetornaFallo	1 ms
▲	✓ ValidarTamanioArchivo_VariosTamanios_RetornaResultadoEsperado	3 ms
	✓ ValidarTamanioArchivo_VariosTamanios_RetornaResultadoEsper...	< 1 ms
	✓ ValidarTamanioArchivo_VariosTamanios_RetornaResultadoEsper...	< 1 ms
	✓ ValidarTamanioArchivo_VariosTamanios_RetornaResultadoEsper...	< 1 ms
	✓ ValidarTamanioArchivo_VariosTamanios_RetornaResultadoEsper...	3 ms
	✓ ValidarTamanioArchivo_VariosTamanios_RetornaResultadoEsper...	< 1 ms
▲	✓ ValidarTipoArchivo_VariasExtensiones_RetornaResultadoEsperado ..	< 1 ms
	✓ ValidarTipoArchivo_VariasExtensiones_RetornaResultadoEsperad...	< 1 ms
	✓ ValidarTipoArchivo_VariasExtensiones_RetornaResultadoEsperad...	< 1 ms
	✓ ValidarTipoArchivo_VariasExtensiones_RetornaResultadoEsperad...	< 1 ms
	✓ ValidarTipoArchivo_VariasExtensiones_RetornaResultadoEsperad...	< 1 ms

Figura J.7: Ejecución de los 14 tests de ArchivoService.

Categorías de tests:

Tests de Validación de Tipo de Archivo (4 tests)

1. `ValidarTipoArchivo_VariasExtensiones_RetornaResultadoEsperado`: Test parametrizado con [Theory] que valida 4 casos de extensiones de archivo:
 - PDF válido: "documento.pdf" → Válido
 - JPG válido: "imagen.jpg" → Válido
 - DOCX no permitido: "archivo.docx" → Inválido
 - Extensión mayúscula: "archivo.PDF" → Válido (case-insensitive)

Tests de Validación de Tamaño de Archivo (5 tests)

1. `ValidarTamanioArchivo_VariosTamanios_RetornaResultadoEsperado`: Test parametrizado que valida 5 casos de tamaño con límite de 10 MB:
 - 1 MB → Válido
 - 5 MB → Válido
 - 10 MB (límite exacto) → Válido (boundary test)
 - 11 MB → Inválido
 - 20 MB → Inválido

Tests de Operaciones con Archivos Físicos (5 tests)

1. `GuardarArchivoAsync_CreaArchivoFisico`: Verifica la creación física de un archivo en el sistema de archivos, validando:
 - Creación de directorio si no existe

- Almacenamiento del contenido binario
 - Retorno de la ruta relativa correcta
 - Verificación de que el contenido almacenado coincide con el original
 - Limpieza posterior del test (cleanup en finally)
2. `ObtenerArchivoAsync_ArchivoNoExiste_RetornaFallo`: Valida el manejo de error cuando se intenta obtener un archivo inexistente en el sistema de archivos.
 3. `ObtenerArchivoAsync_ArchivoExiste_RetornaContenido`: Verifica la lectura exitosa de un archivo físico existente y la recuperación correcta de su contenido binario.
 4. `ConvertirBase64AArchivoAsync_EntradaInvalida_RetornaFallo`: Valida el manejo de errores cuando se intenta convertir una cadena Base64 inválida o corrupta.
 5. `EliminarArchivoAsync_ArchivoPresente_LoBorra`: Verifica la eliminación física exitosa de un archivo del sistema, confirmando que ya no existe después de la operación.

Cobertura y alcance: Los 14 tests cubren el ciclo de vida completo de un archivo: validación de formato y tamaño, almacenamiento físico, recuperación, conversión de formatos, y eliminación. El servicio implementa validaciones críticas antes de operaciones de I/O, manejo robusto de excepciones, y cleanup automático en tests para evitar archivos huérfanos en el sistema de archivos.

ConstanciaService (11 tests)

El servicio ConstanciaService gestiona la subida, validación, obtención y descarga de constancias documentales asociadas a postulantes. Integra el servicio de archivos para operaciones de almacenamiento físico. Como puede observarse en la Figura J.8, los 11 tests de ConstanciaService se ejecutan correctamente, verificando la correcta generación, obtención y descarga de constancias asociadas al postulante.

▲ ✓ ConstanciaServiceTests (11)	145 ms
✓ DescargarConstanciaAsync_Existe_RetornaArchivo	9 ms
✓ DescargarConstanciaAsync_NoExiste_RetornaFallo	2 ms
✓ ObtenerConstanciaPorIdAsync_Existe_RetornaDto	2 ms
✓ ObtenerConstanciaPorIdAsync_NoExiste_RetornaFallo	3 ms
✓ ObtenerConstanciasPorPostulanteAsync_MapeaEntidadADto	6 ms
✓ SubirConstanciaAsync_AlmacenaYRetornaConstancia	8 ms
✓ SubirConstanciaAsync_Error_RetornaMensajeError	13 ms
✓ SubirConstanciaAsync_PostulanteNoExiste_RetornaFallo	6 ms
✓ SubirConstanciaAsync_TipoArchivoInvalido_RetornaErrores	3 ms
✓ ValidarConstanciaAsync_NoEncontrada_RetornaFallo	91 ms
✓ ValidarConstanciaAsync_Valida_RetornaTrue	2 ms

Figura J.8: Ejecución de los 11 tests de ConstanciaService.

Categorías de tests:

Tests de Subida de Constancias (4 tests)

1. SubirConstanciaAsync_PostulanteNoExiste_RetornaFallo: Valida que no se puede subir una constancia para un postulante inexistente, evitando registros huérfanos.
2. SubirConstanciaAsync_TipoArchivoInvalido_RetornaErrores: Verifica la validación de tipo de archivo mediante integración con ArchivoService, rechazando extensiones no permitidas y retornando errores descriptivos.
3. SubirConstanciaAsync_AlmacenaYRetornaConstancia: Verifica el flujo completo exitoso de subida:
 - Validación de existencia de postulante
 - Validación de tipo de archivo
 - Persistencia de metadatos en base de datos

- Retorno del DTO con ID asignado
 - Confirmación de transacción (SaveChanges)
4. SubirConstanciaAsync_Error_RetornaMensajeError: Valida el manejo robusto de excepciones durante el proceso de subida, capturando errores de base de datos y retornando detalles estructurados.

Tests de Obtención de Constancias (3 tests)

1. ObtenerConstanciasPorPostulanteAsync_MapeaEntidadADto: Verifica la obtención del listado completo de constancias asociadas a un postulante, mapeando correctamente todos los campos de la entidad al DTO.
2. ObtenerConstanciaPorIdAsync_NoExiste_RetornaFallo: Valida el manejo de error cuando se solicita una constancia inexistente por ID.
3. ObtenerConstanciaPorIdAsync_Existe_RetornaDto: Verifica la obtención exitosa de una constancia individual por ID con mapeo correcto de datos.

Tests de Validación de Constancias (2 tests)

1. ValidarConstanciaAsync_NoEncontrada_RetornaFallo: Valida que no se puede marcar como validada una constancia inexistente, previniendo operaciones sobre registros eliminados.
2. ValidarConstanciaAsync_Valida_RetornaTrue: Verifica el proceso exitoso de validación de una constancia por parte del tribunal, marcándola como verificada y confirmando cambios en base de datos.

Tests de Descarga de Constancias (2 tests)

1. DescargarConstanciaAsync_NoExiste_RetornaFallo: Valida el manejo de error cuando se intenta descargar una constancia inexistente.
2. DescargarConstanciaAsync_Existe_RetornaArchivo: Verifica la integración completa con ArchivoService para recuperar el contenido binario del archivo físico asociado a la constancia.

Cobertura y alcance: Los 11 tests validan el ciclo completo de gestión de constancias: subida con validaciones múltiples, listado y consulta individual, proceso de validación por tribunal, y descarga de archivos. El servicio demuestra integración efectiva con ArchivoService para operaciones de I/O y manejo transaccional correcto para garantizar consistencia entre metadatos y archivos físicos.

DepartamentoService (9 tests)

El servicio DepartamentoService gestiona la información de departamentos uruguayos y su relación con llamados a concurso, validando disponibilidad y configuración de puestos por departamento. Como puede observarse en la Figura J.9, los 9 tests de DepartamentoService se ejecutan correctamente, validando tanto la obtención de departamentos como la verificación de disponibilidad en llamados.

▲	✓ DepartamentoServiceTests (9)	252 ms
	✓ ObtenerDepartamentoPorIdAsync_Existe_MapeaDto	7 ms
	✓ ObtenerDepartamentoPorIdAsync_NoExiste_RetornaFallo	185 ms
	✓ ObtenerDepartamentosActivosAsync_Error_RetornaFallo	34 ms
	✓ ObtenerDepartamentosActivosAsync_Exito_MapeaEntidad	3 ms
	✓ ObtenerDepartamentosPorLlamadoAsync_DevuelveDepartament...	8 ms
	✓ ObtenerDepartamentosPorLlamadoAsync_LlamadoNoExiste_Fallo	6 ms
	✓ ValidarDepartamentoEnLlamadoAsync_Error_RetornaFallo	2 ms
▲	✓ ValidarDepartamentoEnLlamadoAsync_PropagaResultado (2)	7 ms
	✓ ValidarDepartamentoEnLlamadoAsync_PropagaResultado(existe...)	6 ms
	✓ ValidarDepartamentoEnLlamadoAsync_PropagaResultado(existe...)	1 ms

Figura J.9: Ejecución de los 9 tests de DepartamentoService.

Categorías de tests:

Tests de Obtención de Departamentos Generales (4 tests)

1. ObtenerDepartamentosActivosAsync_Exito_MapeaEntidad: Verifica la obtención del listado completo de departamentos activos en el sistema, mapeando correctamente nombre y código de departamento.
2. ObtenerDepartamentosActivosAsync_Error_RetornaFallo: Valida el manejo robusto de excepciones durante la consulta de departamentos activos.
3. ObtenerDepartamentoPorIdAsync_NoExiste_RetornaFallo: Valida el manejo de error cuando se solicita un departamento inexistente por ID.
4. ObtenerDepartamentoPorIdAsync_Existe_MapeaDto: Verifica la obtención exitosa de un departamento individual con mapeo correcto de sus propiedades.

Tests de Departamentos por Llamado (2 tests)

1. ObtenerDepartamentosPorLlamadoAsync_LlamadoNoExiste_Fallo: Valida el manejo de error cuando se consultan departamentos de un llamado inexistente.
2. ObtenerDepartamentosPorLlamadoAsync_DevuelveDepartamentosDelLlamado: Verifica la obtención de los departamentos configurados para un llamado específico, incluyendo la cantidad de puestos disponibles por departamento.

Tests de Validación de Departamento en Llamado (3 tests)

1. ValidarDepartamentoEnLlamadoAsync_PropagaResultado: Test parametrizado con [Theory] que valida 2 escenarios:
 - Departamento disponible en llamado → retorna true con mensaje "llamado"
 - Departamento no disponible → retorna false con mensaje "no disponible"
2. ValidarDepartamentoEnLlamadoAsync_Error_RetornaFallo: Verifica el manejo de excepciones durante la validación, capturando errores de base de datos y retornando detalles estructurados.

Cobertura y alcance: Los 9 tests cubren las operaciones principales de gestión de departamentos: consulta de catálogo activo, obtención individual, consulta de departamentos configurados por llamado (con cantidad de puestos), y validación de disponibilidad. El servicio es fundamental para garantizar que las inscripciones solo puedan realizarse en departamentos efectivamente configurados para cada llamado específico.

Cobertura por Controlador

Los tests unitarios de controladores validan la capa de presentación de la API REST, verificando el correcto mapeo entre las respuestas de servicios y los códigos de estado HTTP. Estos tests garantizan que la API responda según el estándar HTTP para cada escenario de éxito o error, sin ejecutar lógica de negocio real gracias al uso de mocks.

TribunalController (14 tests)

El controlador TribunalController expone endpoints para el módulo de evaluación de inscripciones, calificación de pruebas y méritos, generación de ordenamientos, y consulta de estadísticas. Los tests validan el correcto mapeo de respuestas del servicio a códigos HTTP estándar. Como puede observarse en la Figura J.10, los 14 tests de TribunalController se ejecutan correctamente, validando el mapeo adecuado entre las respuestas del servicio y los códigos HTTP esperados.

Test	Time (ms)
✓ TribunalControllerTests (14)	138 ms
✓ CalificarPrueba_DatosValidos_RetornaOk	4 ms
✓ CalificarPrueba_Error_RetornaBadRequest	6 ms
✓ GenerarOrdenamiento_DatosValidos_RetornaOk	34 ms
✓ ObtenerDetalleEvaluacion_Exitoso_RetornaOk	10 ms
✓ ObtenerDetalleEvaluacion_NoEncontrado_RetornaNotFound	3 ms
✓ ObtenerDetalleOrdenamiento_Exitoso_RetornaOk	5 ms
✓ ObtenerEstadisticas_Exitoso_RetornaOk	5 ms
✓ ObtenerInscripcionesParaEvaluar_Error_RetornaBadRequest	4 ms
✓ ObtenerInscripcionesParaEvaluar_Exitoso_RetornaOk	30 ms
✓ ObtenerOrdenamientos_Exitoso_RetornaOk	8 ms
✓ ObtenerPruebasDellLlamado_Exitoso_RetornaOk	6 ms
✓ PublicarOrdenamiento_Exitoso_RetornaOk	6 ms
✓ ValorarMerito_DatosValidos_RetornaOk	6 ms
✓ ValorarMeritos_DatosValidos_RetornaOk	11 ms

Figura J.10: Ejecución de los 14 tests de TribunalController.

Categorías de tests:

Tests de Calificación y Valoración (4 tests)

1. `CalificarPrueba_DatosValidos_RetornaOk`: Verifica que al calificar una prueba con datos válidos se retorna código HTTP 200 OK con la evaluación creada.
2. `CalificarPrueba_Error_RetornaBadRequest`: Valida que errores en la calificación (puntaje fuera de rango, prueba inexistente) retornan código HTTP 400 Bad Request.
3. `ValorarMerito_DatosValidos_RetornaOk`: Verifica la valoración individual de un mérito con retorno de código HTTP 200 OK.
4. `ValorarMeritos_DatosValidos_RetornaOk`: Valida la valoración masiva de múltiples méritos de una inscripción, retornando código HTTP 200 OK.

Tests de Consulta de Información (6 tests)

1. `ObtenerDetalleEvaluacion_Exitoso_RetornaOk`: Valida que la consulta exitosa de detalle de evaluación retorna código HTTP 200 OK con todos los datos de la inscripción evaluada.
2. `ObtenerDetalleEvaluacion_NoEncontrado_RetornaNotFound`: Verifica que la consulta de una evaluación inexistente retorna código HTTP 404 Not Found.
3. `ObtenerEstadisticas_Exitoso_RetornaOk`: Verifica que la consulta de estadísticas del tribunal retorna código HTTP 200 OK con métricas del llamado.
4. `ObtenerInscripcionesParaEvaluar_Exitoso_RetornaOk`: Verifica la obtención exitosa del listado de inscripciones pendientes de evaluación.
5. `ObtenerInscripcionesParaEvaluar_Error_RetornaBadRequest`: Valida que errores al obtener inscripciones retornan código HTTP 400 Bad Request.

6. ObtenerPruebasDelllamado_Exitoso_RetornaOk: Verifica la obtención del listado de pruebas configuradas para un llamado.

Tests de Gestión de Ordenamientos (4 tests)

1. GenerarOrdenamiento_DatosValidos_RetornaOk: Verifica que la generación exitosa de ordenamiento retorna código HTTP 200 OK con el resultado de la operación.
2. ObtenerDetalleOrdenamiento_Exitoso_RetornaOk: Valida la consulta exitosa de un ordenamiento con todas sus posiciones, retornando código HTTP 200 OK.
3. ObtenerOrdenamientos_Exitoso_RetornaOk: Valida la consulta de ordenamientos de un llamado, retornando código HTTP 200 OK.
4. PublicarOrdenamiento_Exitoso_RetornaOk: Valida que la publicación de un ordenamiento retorna código HTTP 200 OK confirmando la operación.

Cobertura y alcance: Los 14 tests validan los 10 endpoints principales del controlador de tribunal, cubriendo el flujo completo desde la consulta de inscripciones pendientes, calificación de pruebas y méritos, generación de ordenamientos con aplicación de cuotas, hasta la publicación de resultados finales. Se verifica especialmente el correcto manejo de códigos HTTP 200 OK, 400 Bad Request y 404 Not Found según cada escenario.

ConstanciaController (10 tests)

El controlador ConstanciaController gestiona la subida, consulta, validación y descarga de constancias documentales asociadas a postulantes. Los tests validan el ciclo completo de gestión documental con énfasis en validación de archivos y retorno de contenido binario. Como puede observarse en la Figura J.11, los 10 tests de ConstanciaController se ejecutan correctamente, validando el flujo completo de gestión documental y el mapeo adecuado de resultados a códigos HTTP estándar.

▲	✓ ConstanciaControllerTests (10)	94 ms
	✓ DescargarConstancia_NoExiste_ReturnsNotFound	41 ms
	✓ DescargarConstancia_Success_ReturnsFile	22 ms
	✓ ObtenerConstancia_Existe_ReturnsOk	1 ms
	✓ ObtenerConstancia_NoExiste_ReturnsNotFound	3 ms
	✓ ObtenerConstanciasPorPostulante_ReturnsOk	4 ms
	✓ SubirConstancia_Failure_ReturnsBadRequest	2 ms
	✓ SubirConstancia_ModelStateInvalido_ReturnsBadRequest	4 ms
	✓ SubirConstancia_Success_ReturnsCreated	12 ms
	✓ ValidarConstancia_Failure_ReturnsBadRequest	4 ms
	✓ ValidarConstancia_Success_ReturnsOk	1 ms

Figura J.11: Ejecución de los 10 tests de ConstanciaController.

Categorías de tests:

Tests de Subida de Constancias (3 tests)

1. SubirConstancia_Success_ReturnsCreated: Valida que la subida exitosa retorna código HTTP 201 Created con Location header apuntando al endpoint de consulta.
2. SubirConstancia_Failure_ReturnsBadRequest: Valida que errores en la subida (tipo de archivo inválido, postulante inexistente) retornan código HTTP 400 Bad Request.
3. SubirConstancia_ModelStateInvalido_ReturnsBadRequest: Verifica que errores de validación de modelo retornan código HTTP 400 Bad Request sin ejecutar el servicio.

Tests de Consulta de Constancias (3 tests)

1. ObtenerConstanciasPorPostulante_ReturnsOk: Verifica la obtención del listado completo de constancias de un postulante, retornando código HTTP 200 OK.

2. ObtenerConstancia_Existe_ReturnsOk: Verifica la consulta exitosa de una constancia por ID, retornando código HTTP 200 OK con sus metadatos.
3. ObtenerConstancia_NoExiste_ReturnsNotFound: Valida que la consulta de una constancia inexistente retorna código HTTP 404 Not Found.

Tests de Validación y Descarga (4 tests)

1. ValidarConstancia_Success_ReturnsOk: Valida que la validación exitosa de una constancia por el tribunal retorna código HTTP 200 OK.
2. ValidarConstancia_Failure_ReturnsBadRequest: Verifica que errores al validar una constancia retornan código HTTP 400 Bad Request.
3. DescargarConstancia_Success_ReturnsFile: Valida que la descarga exitosa retorna un FileContentResult con el contenido binario del archivo y content-type application/octet-stream.
4. DescargarConstancia_NoExiste_ReturnsNotFound: Verifica que la descarga de una constancia inexistente retorna código HTTP 404 Not Found.

Cobertura y alcance: Los 10 tests cubren el ciclo completo de gestión documental: subida con validación de tipo de archivo y modelo, consulta individual y por postulante, validación por tribunal, y descarga con retorno de archivo binario. Se verifica especialmente el uso correcto de códigos HTTP 201 Created para recursos nuevos y FileContentResult para descargas de archivos.

InscripcionController (9 tests)

El controlador InscripcionController gestiona el proceso completo de inscripción de postulantes a llamados, incluyendo creación, consulta, validación de requisitos y cálculo de puntajes. Los tests validan el flujo crítico de inscripción end-to-end. Como puede observarse en la Figura J.12, los 9 tests de InscripcionController se ejecutan correctamente, validando el flujo completo de inscripción y asegurando respuestas HTTP coherentes frente a escenarios válidos y de error.

✓	InscripcionControllerTests (9)	117 ms
✓	CalcularPuntajeTotal_Error_RetornaBadRequest	12 ms
✓	CalcularPuntajeTotal_Exitoso_RetornaOk	7 ms
✓	CrearInscripcion_DatosInvalidos_RetornaBadRequest	8 ms
✓	CrearInscripcion_DatosValidos_RetornaCreated	20 ms
✓	ObtenerInscripcion_InscripcionExiste_RetornaOk	34 ms
✓	ObtenerInscripcion_InscripcionNoExiste_RetornaNotFound	6 ms
✓	ObtenerInscripcionesPorPostulante_RetornaLista	6 ms
✓	ValidarInscripcionExistente_Existe_RetornaTrue	17 ms
✓	ValidarRequisitosObligatorios_Cumple_RetornaTrue	7 ms

Figura J.12: Ejecución de los 9 tests de InscripcionController.

Categorías de tests:

Tests de Creación de Inscripción (2 tests)

1. CrearInscripcion_DatosValidos_RetornaCreated: Valida que la creación exitosa retorna código HTTP 201 Created con Location header apuntando al recurso creado.
2. CrearInscripcion_DatosInvalidos_RetornaBadRequest: Verifica que datos inválidos en la creación retornan código HTTP 400 Bad Request con detalles de errores.

Tests de Consulta de Inscripciones (3 tests)

1. ObtenerInscripcion_InscripcionExiste_RetornaOk: Verifica la consulta exitosa de una inscripción por ID, retornando código HTTP 200 OK con el detalle completo.
2. ObtenerInscripcion_InscripcionNoExiste_RetornaNotFound: Valida que la consulta de una inscripción inexistente retorna código HTTP 404 Not Found.
3. ObtenerInscripcionesPorPostulante_RetornaLista: Verifica la obtención del listado de inscripciones activas de un postulante, retornando código HTTP 200 OK.

Tests de Validación y Cálculos (4 tests)

1. ValidarInscripcionExistente_Existe_RetornaTrue: Valida la verificación de existencia de inscripción duplicada, retornando true cuando ya existe.
2. ValidarRequisitosObligatorios_Cumple_RetornaTrue: Verifica la validación de que todos los requisitos excluyentes están completos.
3. CalcularPuntajeTotal_Exitoso_RetornaOk: Valida el cálculo exitoso del puntaje total de una inscripción, retornando código HTTP 200 OK con el valor decimal.
4. CalcularPuntajeTotal_Error_RetornaBadRequest: Verifica que errores al calcular puntaje total retornan código HTTP 400 Bad Request.

Cobertura y alcance: Los 9 tests validan el flujo completo de inscripción: creación con generación de código único, consulta individual y por postulante, validaciones de duplicidad y requisitos, y cálculo de puntaje total. Se verifica especialmente el uso de código HTTP 201 Created con Location header para nuevas inscripciones y la correcta propagación de errores de validación.

LlamadoController (8 tests)

El controlador LlamadoController expone endpoints para consultar llamados disponibles, sus requisitos, ítems puntuables y apoyos necesarios. Los tests validan la correcta exposición de información pública de llamados. Como puede observarse en la Figura J.13, los 8 tests del LlamadoController se ejecutan correctamente, validando la exposición pública de información de llamados, incluyendo requisitos, apoyos necesarios, ítems puntuables y disponibilidad del llamado.

▲	✓ LlamadoControllerTests (8)	201 ms
	✓ ObtenerApoyosNecesariosLlamado_RetornaLista	7 ms
	✓ ObtenerItemsPuntuablesLlamado_RetornaLista	6 ms
	✓ ObtenerLlamado_LlamadoExiste_RetornaOk	4 ms
	✓ ObtenerLlamado_LlamadoNoExiste_RetornaNotFound	5 ms
	✓ ObtenerLlamadosActivos_RetornaLista	11 ms
	✓ ObtenerLlamadosInactivos_RetornaLista	157 ms
	✓ ObtenerRequisitosLlamado_RetornaLista	5 ms
	✓ ValidarLlamadoDisponible_Disponible_RetornaTrue	6 ms

Figura J.13: Ejecución de tests del LlamadoController.

Categorías de tests:

Tests de Consulta de Llamados (4 tests)

1. ObtenerLlamado_LlamadoExiste_RetornaOk: Verifica la consulta exitosa del detalle completo de un llamado, incluyendo todas sus relaciones (departamentos, requisitos, ítems, apoyos).
2. ObtenerLlamado_LlamadoNoExiste_RetornaNotFound: Valida que la consulta de un llamado inexistente retorna código HTTP 404 Not Found.
3. ObtenerLlamadosActivos_RetornaLista: Verifica la obtención del listado de llamados en estado “abierto” disponibles para inscripción.

4. ObtenerLlamadosInactivos_RetornaLista: Valida la consulta de llamados cerrados o en otros estados no activos.

Tests de Componentes del Llamado (4 tests)

1. ObtenerRequisitosLlamado_RetornaLista: Verifica la obtención de requisitos excluyentes del llamado con sus tipos y obligatoriedad.
2. ObtenerItemsPuntuablesLlamado_RetornaLista: Valida la consulta de ítems puntuables (méritos) configurados para el llamado con puntajes máximos.
3. ObtenerApoyosNecesariosLlamado_RetornaLista: Verifica la obtención del catálogo de apoyos disponibles para solicitar en un llamado.
4. ValidarLlamadoDisponible_Disponible_RetornaTrue: Valida la verificación de disponibilidad de un llamado para inscripción.

Cobertura y alcance: Los 8 tests validan los endpoints de consulta pública de llamados, garantizando que los postulantes tengan acceso completo a la información necesaria para decidir su inscripción: detalle del llamado, requisitos excluyentes que deben cumplir, méritos puntuables que pueden declarar, y apoyos que pueden solicitar. Se verifica la correcta separación entre llamados activos e inactivos.

PostulanteController (8 tests)

El controlador PostulanteController gestiona el perfil del postulante, actualización de datos personales y validación de unicidad de cédula/email. Los tests validan el flujo de completitud de perfil y prevención de duplicados. Como puede observarse en la Figura J.14, los 8 tests del PostulanteController se ejecutan correctamente, validando tanto respuestas exitosas como errores esperados según cada caso evaluado.

PostulanteControllerTests (8)	191 ms
CompletarDatosPersonales_DatosInvalidos_RetornaBadRequest	5 ms
CompletarDatosPersonales_DatosValidos_RetornaOk	158 ms
ObtenerPostulante_PostulanteExiste_RetornaOk	4 ms
ObtenerPostulante_PostulanteNoExiste_RetornaNotFound	4 ms
ValidarCedulaDisponible_CedulaDisponible_RetornaTrue	6 ms
ValidarCedulaDisponible_CedulaNoDisponible_RetornaFalse	3 ms
ValidarEmailDisponible_EmailDisponible_RetornaTrue	8 ms
ValidarEmailDisponible_EmailNoDisponible_RetornaFalse	3 ms

Figura J.14: Ejecución de tests del PostulanteController.

Categorías de tests:

Tests de Consulta de Perfil (2 tests)

1. ObtenerPostulante_PostulanteExiste_RetornaOk: Verifica la consulta exitosa del perfil completo de un postulante, retornando código HTTP 200 OK.
2. ObtenerPostulante_PostulanteNoExiste_RetornaNotFound: Valida que la consulta de un postulante inexistente retorna código HTTP 404 Not Found.

Tests de Actualización de Datos (2 tests)

1. CompletarDatosPersonales_DatosValidos_RetornaOk: Valida la actualización exitosa de datos personales (nombre, apellido, fecha nacimiento, domicilio, contacto), retornando código HTTP 200 OK.
2. CompletarDatosPersonales_DatosInvalidos_RetornaBadRequest: Verifica que datos inválidos o incompletos retornan código HTTP 400 Bad Request con detalles de errores de validación.

Tests de Validación de Unicidad (4 tests)

1. ValidarCedulaDisponible_CedulaDisponible_RetornaTrue: Verifica la validación de disponibilidad de cédula (no registrada previamente), retornando true.
2. ValidarCedulaDisponible_CedulaNoDisponible_RetornaFalse: Valida la detección de cédula duplicada en el sistema, retornando false.
3. ValidarEmailDisponible_EmailDisponible_RetornaTrue: Verifica la validación de disponibilidad de email, retornando true cuando está disponible.
4. ValidarEmailDisponible_EmailNoDisponible_RetornaFalse: Valida la detección de email duplicado en el sistema, retornando false.

Cobertura y alcance: Los 8 tests cubren el ciclo de gestión de perfil: consulta de datos existentes, actualización completa con validaciones múltiples, y verificación de unicidad de cédula/email antes del registro. Las validaciones de unicidad son críticas para prevenir registros duplicados y garantizar la integridad referencial del sistema.

DepartamentoController (6 tests)

El controlador DepartamentoController gestiona la consulta de departamentos y su relación con llamados, validando configuración de puestos por región. Los tests validan la correcta exposición de información geográfica. Como puede observarse en la Figura J.15, los 6 tests del DepartamentoController se ejecutan correctamente, cubriendo tanto respuestas exitosas como errores esperados cuando un departamento no existe o no pertenece al llamado solicitado.

▲	✓ DepartamentoControllerTests (6)	173 ms
	✓ ObtenerDepartamento_Existe_ReturnsOk	2 ms
	✓ ObtenerDepartamento_NoExiste_ReturnsNotFound	4 ms
	✓ ObtenerDepartamentosActivos_ReturnsOk	4 ms
	✓ ObtenerDepartamentosPorLlamado_Existe_ReturnsOk	3 ms
	✓ ObtenerDepartamentosPorLlamado_NoExiste_ReturnsNotFound	155 ms
	✓ ValidarDepartamentoEnLlamado_ReturnsOk	5 ms

Figura J.15: Ejecución de tests del DepartamentoController.

Categorías de tests:

Tests de Consulta de Departamentos (3 tests)

1. ObtenerDepartamentosActivos_ReturnsOk: Verifica la obtención del catálogo completo de departamentos activos en el sistema (19 departamentos de Uruguay).
2. ObtenerDepartamento_Existe_ReturnsOk: Verifica la consulta exitosa de un departamento individual por ID, retornando código HTTP 200 OK.
3. ObtenerDepartamento_NoExiste_ReturnsNotFound: Valida que la consulta de un departamento inexistente retorna código HTTP 404 Not Found.

Tests de Departamentos por Llamado (3 tests)

1. ObtenerDepartamentosPorLlamado_Existe_ReturnsOk: Valida la consulta de departamentos configurados para un llamado específico, incluyendo cantidad de puestos disponibles por departamento.
2. ObtenerDepartamentosPorLlamado_NoExiste_ReturnsNotFound: Verifica que la consulta de departamentos para un llamado inexistente retorna código HTTP 404 Not Found.
3. ValidarDepartamentoEnLlamado_ReturnsOk: Valida la verificación de si un departamento específico está disponible para inscripción en un llamado determinado.

Cobertura y alcance: Los 6 tests validan la correcta exposición de información geográfica y su relación con llamados. Es fundamental que los postulantes puedan consultar qué departamentos están disponibles para cada llamado, ya que algunos llamados son nacionales mientras otros tienen cupos específicos por región. La validación de departamento en llamado previene inscripciones en regiones no configuradas.

Pruebas de Integración

Las pruebas de integración validan el pipeline HTTP completo de la aplicación, desde la solicitud del cliente hasta la persistencia en base de datos, incluyendo toda la cadena de middleware, serialización JSON, validación de modelos, ejecución de servicios reales, y operaciones de base de datos. A diferencia de las pruebas unitarias que utilizan mocks, estas pruebas levantan un servidor web real mediante `WebApplicationFactory` y ejecutan el flujo end-to-end completo.

Infraestructura de pruebas: Todas las pruebas de integración utilizan:

- `WebApplicationFactory<Program>` para levantar el servidor ASP.NET Core
- `HttpClient` real para realizar solicitudes HTTP
- Base de datos SQL Server con datos de prueba pre-cargados mediante `DbInitializer`
- `CustomWebApplicationFactory` para configuración personalizada del entorno de testing
- Colección de tests compartida para optimizar la creación del servidor entre tests

TribunalController - Integración (13 tests)

Las pruebas de integración del controlador `TribunalController` validan el flujo completo de evaluación de inscripciones, calificación de pruebas y méritos, generación de ordenamientos, y publicación de resultados, incluyendo persistencia real en base de datos y aplicación de reglas de negocio complejas. Como puede observarse en la Figura J.16, los 13 tests de integración ejecutan escenarios completos tanto válidos como de error, cubriendo respuestas 200 OK, 400 BadRequest y 404 NotFound, lo que garantiza una capa de API robusta y alineada con los contratos definidos.

▲	● TribunalControllerTests (13)	1.1 s
●	CalificarPrueba_DebeRetornarBadRequest_CuandoDatosInvalidos	96 ms
●	ObtenerDetalleEvaluacion_DebeRetornarNotFound_CuandoInscripcionNoExiste	6 ms
●	ObtenerDetalleEvaluacion_DebeRetornarOk_CuandoInscripcionExiste	23 ms
●	ObtenerDetalleOrdenamiento_DebeRetornarNotFound_CuandoOrdenamientoNoExiste	109 ms
●	ObtenerEstadisticas_DebeRetornarBadRequest_CuandoLlamadoNoExiste	3 ms
●	ObtenerEstadisticas_DebeRetornarOk_CuandoLlamadoExiste	439 ms
●	ObtenerInscripcionesParaEvaluar_DebeRetornarOk_CuandoLlama... (truncated)	14 ms
●	ObtenerInscripcionesParaEvaluar_DebeRetornarOk_CuandoLlama... (truncated)	17 ms
●	ObtenerOrdenamientos_DebeRetornarOk_CuandoLlamadoExiste	324 ms
●	ObtenerOrdenamientos_DebeRetornarOk_CuandoLlamadoNoExiste	5 ms
●	ObtenerPruebasDellLlamado_DebeRetornarOk_CuandoLlamadoExiste	44 ms
●	ObtenerPruebasDellLlamado_DebeRetornarOk_CuandoLlamadoNoExiste	4 ms
●	PublicarOrdenamiento_DebeRetornarBadRequest_CuandoOrdenamientoNoExiste	35 ms

Figura J.16: Pruebas de integración del TribunalController.

Categorías de tests:

Tests de Calificación de Pruebas (1 test)

1. CalificarPrueba_DebeRetornarBadRequest_CuandoDatosInvalidos: Verifica que al intentar calificar una prueba con puntaje negativo se retorna código HTTP 400 Bad Request, validando el pipeline completo de validación de modelos y reglas de negocio.

Tests de Consulta de Evaluaciones (4 tests)

1. ObtenerDetalleEvaluacion_DebeRetornarOk_CuandoInscripcionExiste: Valida la consulta exitosa del detalle completo de evaluación de una inscripción existente en base de datos, verificando deserialización correcta de JSON con todas las relaciones (postulante, méritos, pruebas, requisitos).
2. ObtenerDetalleEvaluacion_DebeRetornarNotFound_CuandoInscripcionNoExiste:

Verifica que la consulta de una inscripción inexistente retorna código HTTP 404 Not Found después de ejecutar la query real en base de datos.

3. ObtenerInscripcionesParaEvaluar_DebeRetornarOk_CuandoLlamadoExiste: Valida la obtención del listado de inscripciones de un llamado existente con cálculo real de puntajes desde base de datos.
4. ObtenerInscripcionesParaEvaluar_DebeRetornarOk_CuandoLlamadoNoExiste: Verifica que la consulta de inscripciones para un llamado inexistente retorna código HTTP 200 OK con lista vacía (comportamiento permisivo del sistema).

Tests de Estadísticas (2 tests)

1. ObtenerEstadisticas_DebeRetornarOk_CuandoLlamadoExiste: Verifica el cálculo real de estadísticas del tribunal (total inscripciones, promedios, distribución por cuotas) con datos reales de base de datos.
2. ObtenerEstadisticas_DebeRetornarBadRequest_CuandoLlamadoNoExiste: Valida que la solicitud de estadísticas para un llamado inexistente retorna código HTTP 400 Bad Request o 404 Not Found.

Tests de Pruebas del Llamado (2 tests)

1. ObtenerPruebasDelLlamado_DebeRetornarOk_CuandoLlamadoExiste: Verifica la obtención de la configuración de pruebas de un llamado con estadísticas de evaluación calculadas desde base de datos.
2. ObtenerPruebasDelLlamado_DebeRetornarOk_CuandoLlamadoNoExiste: Valida comportamiento permisivo retornando código HTTP 200 OK con lista vacía para llamado inexistente.

Tests de Ordenamientos (4 tests)

1. ObtenerOrdenamientos_DebेRetornarOk_CuandoLlamadoExiste: Verifica la consulta de ordenamientos generados para un llamado con datos persistidos en base de datos.
2. ObtenerOrdenamientos_DebеRetornarOk_CuandoLlamadoNoExiste: Valida comportamiento permisivo con lista vacía para llamado inexistente.
3. ObtenerDetalleOrdenamiento_DebеRetornarNotFound_CuandoOrdenamientoNoExiste: Verifica código HTTP 404 Not Found al consultar un ordenamiento inexistente.
4. PublicarOrdenamiento_DebеRetornarBadRequest_CuandoOrdenamientoNoExiste: Valida que intentar publicar un ordenamiento inexistente retorna código HTTP 400 Bad Request o 404 Not Found.

Cobertura y alcance: Los 13 tests de integración validan el flujo completo del módulo de tribunal con operaciones reales de base de datos, incluyendo consultas complejas con joins múltiples, cálculos de puntajes con agregaciones SQL, y verificación de transacciones completas. Se enfatiza especialmente la validación de reglas de negocio críticas como rangos de puntajes y estados de ordenamiento.

PostulanteController - Integración (9 tests)

Las pruebas de integración del controlador PostulanteController validan el flujo completo de gestión de perfil de postulante, incluyendo actualización de datos personales con validaciones de formato, y verificación de unicidad de cédula y email contra base de datos real. Como puede observarse en la Figura J.17, los 9 tests de integración ejecutan casos donde los datos son inválidos, válidos, o cuando el postulante no existe, validando respuestas 400 BadRequest, 404 NotFound y 200 OK según el escenario. Esto asegura una interacción robusta entre controlador, servicio y repositorios.

▲	✓ PostulanteControllerTests (9)	249 ms
	✓ CompletarDatosPersonales_DebeRetornarBadRequest_CuandoDa...	5 ms
	✓ CompletarDatosPersonales_DebeRetornarBadRequest_CuandoPo...	5 ms
	✓ CompletarDatosPersonales_DebeRetornarOk_CuandoDatosSonVa...	52 ms
	✓ ObtenerPostulante_DebeRetornarNotFound_CuandoPostulanteN...	6 ms
	✓ ObtenerPostulante_DebeRetornarOk_CuandoPostulanteExiste	21 ms
	✓ ValidarCedulaDisponible_DebeRetornarOk_ConResultadoFalse_Cu...	26 ms
	✓ ValidarCedulaDisponible_DebeRetornarOk_ConResultadoTrue_Cu...	5 ms
	✓ ValidarEmailDisponible_DebeRetornarOk_ConResultadoFalse_Cua...	4 ms
	✓ ValidarEmailDisponible_DebeRetornarOk_ConResultadoTrue_Cuan...	125 ms

Figura J.17: Pruebas de integración del PostulanteController.

Categorías de tests:

Tests de Consulta de Perfil (2 tests)

1. ObtenerPostulante_DebeRetornarOk_CuandoPostulanteExiste: Verifica la consulta exitosa de un postulante existente en base de datos con deserialización correcta de todos sus datos personales en formato JSON.
2. ObtenerPostulante_DebeRetornarNotFound_CuandoPostulanteNoExiste: Valida que la consulta de un postulante inexistente retorna código HTTP 404 Not Found después de ejecutar la query en base de datos.

Tests de Actualización de Datos Personales (3 tests)

1. CompletarDatosPersonales_DebeRetornarOk_CuandoDatosSonValidos: Valida la actualización exitosa de datos personales con persistencia real en base de datos, verificando que los cambios se confirman mediante transacción completa.
2. CompletarDatosPersonales_DebeRetornarBadRequest_CuandoDatosSonInvalidos:

Verifica el pipeline completo de validación con datos inválidos (nombre vacío, fecha futura, email sin formato), retornando código HTTP 400 Bad Request sin persistir cambios.

3. CompletarDatosPersonales_DebeRetornarBadRequest_CuandoPostulanteNoExiste:
Valida que intentar actualizar un postulante inexistente retorna código HTTP 400 Bad Request o 404 Not Found sin intentar operación en base de datos.

Tests de Validación de Unicidad (4 tests)

1. ValidarCedulaDisponible_DebeRetornarOk_ConResultadoTrue_CuandoCedulaNoExiste:
Verifica la consulta real a base de datos para validar que una cédula no está registrada, retornando true en el campo Data.
2. ValidarCedulaDisponible_DebeRetornarOk_ConResultadoFalse_CuandoCedulaYaExiste:
Valida la detección de cédula duplicada mediante query real en base de datos, retornando false para prevenir registro duplicado.
3. ValidarEmailDisponible_DebeRetornarOk_ConResultadoTrue_CuandoEmailNoExiste:
Verifica la consulta de disponibilidad de email contra base de datos, retornando true cuando está disponible.
4. ValidarEmailDisponible_DebeRetornarOk_ConResultadoFalse_CuandoEmailYaExiste:
Valida la detección de email duplicado mediante query en base de datos, previniendo múltiples postulantes con el mismo email.

Cobertura y alcance: Los 9 tests validan el ciclo completo de gestión de perfil con operaciones reales de base de datos: consulta de datos con deserialización JSON, actualización transaccional con validaciones de formato complejas (cédula uruguaya, email RFC 5322, edad mínima), y verificación de constraints de unicidad antes de persistir. Las validaciones de unicidad son críticas para garantizar integridad referencial del sistema.

LlamadoController - Integración (9 tests)

Las pruebas de integración del controlador LlamadoController validan la consulta de información pública de llamados con todas sus relaciones (departamentos, requisitos, ítems puntuables, apoyos), asegurando que los postulantes reciban datos completos y actualizados desde base de datos. Como puede observarse en la Figura J.18, los 9 tests ejecutados cubren la recuperación de llamados individuales, llamados activos e inactivos, así como la validación de disponibilidad del llamado para inscripción, retornando los códigos HTTP correspondientes (200 OK, 404 NotFound) según cada caso.

▲ ✓ LlamadoControllerTests (9)	423 ms
✓ ObtenerApoyosNecesariosLlamado_DebeRetornarListaApoyos_C...	29 ms
✓ ObtenerItemsPuntuablesLlamado_DebeRetornarListaItems_Cuand...	26 ms
✓ ObtenerLlamado_DebeRetornarLlamado_CuandoExiste	19 ms
✓ ObtenerLlamado_DebeRetornarNotFound_CuandoNoExiste	19 ms
✓ ObtenerLlamadosActivos_DebeRetornarListaConLlamadosAbiertos	25 ms
✓ ObtenerLlamadosInactivos_DebeRetornarListaConLlamadosCerra...	32 ms
✓ ObtenerRequisitosLlamado_DebeRetornarListaRequisitos_Cuando...	257 ms
✓ ValidarLlamadoDisponible_DebeRetornarOk_CuandoLlamadoEsta...	3 ms
✓ ValidarLlamadoDisponible_DebeRetornarOk_CuandoLlamadoEsta...	13 ms

Figura J.18: Pruebas de integración del LlamadoController.

Categorías de tests:

Tests de Consulta de Llamados (4 tests)

1. ObtenerLlamado_DebeRetornarLlamado_CuandoExiste: Verifica la consulta exitosa del detalle completo de un llamado con todas sus relaciones cargadas mediante eager loading (departamentos, requisitos, ítems, apoyos) y deserialización correcta del JSON complejo.
2. ObtenerLlamado_DebeRetornarNotFound_CuandoNoExiste: Valida que la consulta

de un llamado inexistente retorna código HTTP 404 Not Found después de ejecutar query en base de datos.

3. ObtenerLlamadosActivos_DebेRetornarListaConLlamadosAbiertos: Verifica la consulta filtrada de llamados en estado ^abierto con fecha de cierre vigente, ejecutando query con cláusula WHERE contra base de datos real.
4. ObtenerLlamadosInactivos_DebеRetornarListaConLlamadosCerrados: Valida la consulta de llamados en estados cerrados o inactivos, complemento del filtro de llamados activos.

Tests de Validación de Disponibilidad (2 tests)

1. ValidarLlamadoDisponible_DebеRetornarOk_CuandoLlamadoEstaAbierto: Verifica la validación de disponibilidad para inscripción mediante query que verifica estado y fechas contra base de datos, retornando true cuando el llamado acepta inscripciones.
2. ValidarLlamadoDisponible_DebеRetornarOk_CuandoLlamadoEstaCerrado: Valida la detección correcta de llamado cerrado mediante la misma query, retornando false cuando no acepta más inscripciones.

Tests de Componentes del Llamado (3 tests)

1. ObtenerRequisitosLlamado_DebеRetornarListaRequisitos_CuandoLlamadoExiste: Verifica la consulta de requisitos excluyentes asociados al llamado mediante join en base de datos, retornando listado completo con tipos y obligatoriedad.
2. ObtenerItemsPuntuablesLlamado_DebеRetornarListaItems_CuandoLlamadoExiste: Valida la consulta de ítems puntuables (méritos) configurados para el llamado con puntos máximos y categorías desde base de datos.

3. ObtenerApoyosNecesariosLlamado_DebeRetornarListaApoyos_CuandoLlamadoExiste:
Verifica la consulta del catálogo de apoyos disponibles para el llamado mediante relación en base de datos.

Cobertura y alcance: Los 9 tests validan consultas complejas con múltiples joins sobre entidades relacionadas, garantizando que los postulantes reciban información completa y actualizada para tomar decisiones de inscripción. Las queries de filtrado por estado y fechas son críticas para mostrar solo llamados efectivamente disponibles. La deserialización de estructuras JSON anidadas valida el correcto funcionamiento del pipeline de serialización de ASP.NET Core.

InscripcionController - Integración (9 tests)

Las pruebas de integración del controlador InscripcionController validan el flujo completo de inscripción end-to-end, desde la creación transaccional con múltiples entidades relacionadas hasta el cálculo de puntajes con agregaciones SQL, incluyendo validaciones de duplicidad y requisitos contra base de datos real. Como puede observarse en la Figura J.19, los 9 tests de integración cubren escenarios exitosos y fallos esperados, tales como inscripciones inválidas, llamados inexistentes, duplicidad detectada o faltas en requisitos obligatorios, garantizando una ejecución coherente del flujo crítico del sistema.

✓	InscripcionControllerTests (9)	3.8 s
✓	CalcularPuntajeTotal_DebeRetornarOk_CuandoInscripcionExiste	66 ms
✓	CrearInscripcion_DebeRetornarBadRequest_CuandoInscripcionYaE...	60 ms
✓	CrearInscripcion_DebeRetornarBadRequest_CuandoLlamadoNoEx...	3.2 s
✓	ObtenerInscripcion_DebeRetornarInscripcion_CuandoExiste	34 ms
✓	ObtenerInscripcion_DebeRetornarNotFound_CuandoNoExiste	210 ms
✓	ObtenerInscripcionesPorPostulante_DebeRetornarLista_CuandoPo...	90 ms
✓	ValidarInscripcionExistente_DebeRetornarOk_CuandoInscripcionE...	5 ms
✓	ValidarInscripcionExistente_DebeRetornarOk_CuandoInscripcionN...	26 ms
✓	ValidarRequisitosObligatorios_DebeRetornarResultado_CuandoIns...	111 ms

Figura J.19: Pruebas de integración del InscripcionController.

Categorías de tests:

Tests de Creación de Inscripción (2 tests)

1. `CrearInscripcion_DebeRetornarBadRequest_CuandoLlamadoNoExiste`: Valida que intentar crear una inscripción para un llamado inexistente retorna código HTTP 400 Bad Request después de verificar existencia en base de datos, previniendo inscripciones huérfanas.
2. `CrearInscripcion_DebeRetornarBadRequest_CuandoInscripcionYaExiste`: Verifica la validación de duplicidad mediante query en base de datos, retornando código HTTP 400 Bad Request cuando el postulante ya está inscrito en ese llamado, garantizando constraint de unicidad.

Tests de Consulta de Inscripciones (3 tests)

1. `ObtenerInscripcion_DebeRetornarInscripcion_CuandoExiste`: Verifica la consulta exitosa de una inscripción con todas sus relaciones cargadas (postulante, departamento, autodefinición ley, requisitos, méritos, apoyos) mediante eager loading y deserialización JSON completa.
2. `ObtenerInscripcion_DebeRetornarNotFound_CuandoNoExiste`: Valida que la consulta de una inscripción inexistente retorna código HTTP 404 Not Found después de ejecutar query en base de datos.
3. `ObtenerInscripcionesPorPostulante_DebeRetornarLista_CuandoPostulanteExiste`: Verifica la consulta filtrada de inscripciones de un postulante específico mediante query con WHERE sobre FK en base de datos.

Tests de Validaciones (3 tests)

1. `ValidarInscripcionExistente_DebeRetornarOk_CuandoInscripcionExiste`: Verifica la validación de duplicidad mediante query compuesta (postulanteId + llamadoId) en base de datos, retornando true cuando ya existe.
2. `ValidarInscripcionExistente_DebeRetornarOk_CuandoInscripcionNoExiste`: Valida el resultado false de la misma query cuando no hay inscripción previa, permitiendo proceder con la creación.
3. `ValidarRequisitosObligatorios_DebeRetornarResultado_CuandoInscripcionExiste`: Verifica la validación completa de que todos los requisitos excluyentes del llamado fueron declarados en la inscripción mediante join entre requisitos del llamado y requisitos del postulante.

Tests de Cálculo de Puntajes (1 test)

1. `CalcularPuntajeTotal_DebeRetornarOk_CuandoInscripcionExiste`: Valida el cálculo completo del puntaje total mediante agregaciones SQL que suman: (1) puntajes de pruebas aprobadas verificadas, (2) puntajes de méritos aprobados, retornando el resultado decimal con persistencia en base de datos.

Cobertura y alcance: Los 9 tests validan el flujo crítico de inscripción con transacciones complejas que involucran múltiples tablas relacionadas. Las validaciones de duplicidad y requisitos son fundamentales para garantizar integridad de datos. El cálculo de puntaje total con agregaciones SQL demuestra la correcta integración entre lógica de negocio y persistencia, ejecutando queries complejas con SUM y WHERE sobre múltiples tablas relacionadas.

Resumen de cobertura de pruebas de integración:

Controlador	Tests de Integración
TribunalController	13
PostulanteController	9
LlamadoController	9
InscripcionController	9
Total	40

Tabla J.1: Distribución de tests de integración por controlador

Aspectos validados por las pruebas de integración:

Las 40 pruebas de integración garantizan el correcto funcionamiento del pipeline HTTP completo:

- **Serialización/Deserialización JSON:** Validación de conversión bidireccional entre DTOs y entidades con estructuras anidadas complejas
- **Middleware de ASP.NET Core:** Verificación del flujo completo a través de todos los middleware (routing, CORS, exception handling, model validation)
- **Validación de modelos:** Comprobación de atributos DataAnnotations y validaciones personalizadas antes de ejecutar lógica de negocio
- **Operaciones de base de datos:** Ejecución real de queries SQL con Entity Framework Core, incluyendo:
 - Consultas simples y complejas con múltiples joins
 - Eager loading de entidades relacionadas para evitar N+1 queries
 - Agregaciones SQL (SUM, COUNT, AVG) para cálculos de puntajes y estadísticas
 - Transacciones con commit/rollback para operaciones atómicas

- Validación de constraints de unicidad y claves foráneas
- **Códigos de estado HTTP:** Verificación de respuestas apropiadas (200 OK, 201 Created, 400 Bad Request, 404 Not Found) según contexto de operación
- **Manejo de errores end-to-end:** Comprobación de propagación correcta de excepciones desde base de datos hasta respuesta HTTP estructurada

Infraestructura de testing avanzada: El uso de CustomWebApplicationFactory con configuración personalizada permite simular un entorno de producción realista, incluyendo base de datos con datos de prueba pre-cargados mediante seeders, lo que garantiza que los tests sean repetibles y aislados entre ejecuciones.

Categorías de Tests Implementados

Tests de Validación de Reglas de Negocio

Estos tests verifican que las restricciones críticas del dominio se cumplen correctamente:

- Validación de rangos de puntajes para pruebas y méritos
- Prevención de inscripciones duplicadas al mismo llamado
- Verificación de estado del llamado antes de permitir inscripción
- Validación de formato de cédula uruguaya con algoritmo de dígito verificador
- Validación de formato de email según estándar RFC 5322
- Validación de teléfonos uruguayos (celulares y fijos con código de área)
- Verificación de edad mínima para inscripción (18 años)
- Validación de disponibilidad de departamento en llamado específico

Tests de Casos Exitosos

Estos tests validan el flujo completo de operaciones cuando los datos son correctos:

- Creación de inscripción con datos completos y persistencia transaccional
- Calificación de méritos asignando puntajes válidos dentro del rango permitido
- Calificación de pruebas con marcado automático de aprobado/desaprobado según umbral
- Generación de ordenamiento con ordenación por puntaje descendente
- Actualización completa de datos personales de postulante
- Consulta de información de llamados con todas sus relaciones (departamentos, requisitos, ítems, apoyos)
- Subida y validación de constancias documentales con verificación de tipo de archivo
- Operaciones con archivos físicos (guardar, obtener, eliminar) con manejo de directorios

Tests de Casos de Error

Estos tests verifican el manejo apropiado de situaciones excepcionales:

- Entidad no encontrada con retorno de mensaje descriptivo
- Datos inválidos o incompletos con retorno de errores de validación
- Validación de tipo de archivo con rechazo de extensiones no permitidas
- Manejo de errores de conversión Base64 con mensaje descriptivo
- Propagación de excepciones de base de datos con detalles estructurados

Tests de Integridad Transaccional

Estos tests verifican que operaciones complejas mantienen consistencia mediante transacciones:

- Creación de inscripción con persistencia atómica de múltiples entidades relacionadas (inscripción, autodefinición ley, requisitos, méritos, apoyos) mediante `BeginTransaction` y `CommitTransaction`
- Rollback automático cuando falla validación de puntajes
- Generación de ordenamiento con persistencia atómica de ordenamiento + posiciones mediante transacción única
- Validación de que operaciones fallidas no persisten cambios parciales gracias al uso de `RollbackTransactionAsync`

Tests de Mapeo y Transformación de Datos

Estos tests verifican la correcta conversión entre capas de la aplicación:

- Mapeo de entidades de dominio a DTOs de respuesta con todas las propiedades
- Cálculo de puntajes totales sumando méritos aprobados + pruebas verificadas
- Mapeo de colecciones relacionadas con conteo correcto de elementos
- Conversión de códigos de estado HTTP según resultado de operación en controladores (200 OK, 201 Created, 400 Bad Request, 404 Not Found)
- Serialización/deserialización JSON en tests de integración con estructuras anidadas complejas

Tests Parametrizados con Theory

El proyecto utiliza extensivamente tests parametrizados mediante [Theory] e [InlineData] para validar múltiples casos con código conciso:

- Validación de cédula con 6 casos diferentes (formatos válidos e inválidos)
- Validación de email con 5 casos de formatos correctos e incorrectos
- Validación de teléfono con 5 casos (celulares, fijos, con/sin prefijo)
- Validación de tamaño de archivo con 5 casos de boundary testing (1MB, 5MB, 10MB límite, 11MB, 20MB)
- Validación de tipo de archivo con 4 casos de extensiones (válidas, inválidas, case-insensitive)

Ejemplo de Test Unitario

A continuación se presenta un ejemplo representativo de test unitario de la capa de servicios, extraído del archivo `TribunalServiceTests.cs`. Este test valida el mapeo correcto de puntajes y totales al obtener inscripciones para evaluación:

```
1 [Fact] // Atributo de xUnit que marca el método como test
2 public async Task ObtenerInscripcionesParaEvaluarAsync_MapeaPuntajesYTotales()
3 {
4     // ARRANGE: Preparación de datos de prueba
5     var llamadoId = 1;
6     var inscripcion = CrearInscripcion(); // Helper que crea inscripción
7         completa
8
9     // Configurar mock de repositorio de inscripciones
10    _inscripcionesMock
```

```

10     .Setup(r => r.GetByLlamadoIdAsync(llamadoId))
11     .ReturnsAsync(new List<Inscripcion> { inscripcion });
12
13 // Configurar mock de repositorio de pruebas
14 _pruebasMock
15     .Setup(r => r.GetByLlamadoIdAsync(llamadoId))
16     .ReturnsAsync(new List<Prueba> { new Prueba { Id = 10 } });
17
18 // Configurar evaluaciones de pruebas (1 verificada aprobada, 1 no
19     ↳ verificada)
20 _evaluacionesPruebasMock
21     .Setup(r => r.GetByInscripcionIdAsync(inscripcion.Id))
22     .ReturnsAsync(new List<EvaluacionPrueba>
23     {
24         new EvaluacionPrueba {
25             Verificado = true,
26             PuntajeObtenido = 30,
27             Aprobado = true
28         },
29         new EvaluacionPrueba {
30             Verificado = false,
31             PuntajeObtenido = 20,
32             Aprobado = true
33         }
34     });
35
36 // Configurar evaluaciones de méritos (1 aprobado, 1 rechazado)
37 _evaluacionesMeritosMock
38     .Setup(r => r.GetByInscripcionIdAsync(inscripcion.Id))
39     .ReturnsAsync(new List<EvaluacionMerito>
40     {

```

```

40         new EvaluacionMerito {
41             Estado = "Aprobado",
42             PuntajeAsignado = 15
43         },
44         new EvaluacionMerito {
45             Estado = "Rechazado",
46             PuntajeAsignado = 40
47         }
48     );
49
50 // Configurar méritos del postulante
51 _meritosMock
52     .Setup(r => r.GetByInscripcionIdAsync(inscripcion.Id))
53     .ReturnsAsync(new List<MeritoPostulante> {
54         new MeritoPostulante(),
55         new MeritoPostulante()
56     });
57
58 // ACT: Ejecutar el método a probar
59 var resultado = await _sut.ObtenerInscripcionesParaEvaluarAsync(llamadoId);
60
61 // ASSERT: Verificar resultados esperados
62 Assert.True(resultado.Success);
63 var data = AssertNotNull(resultado.Data);
64 Assert.Single(data); // Solo 1 inscripción retornada
65
66 // Verificar puntajes calculados correctamente:
67 // - PuntajePruebas: Solo suma pruebas VERIFICADAS (30)
68 Assert.Equal(30, data[0].PuntajePruebas);
69
70 // - PuntajeMeritos: Solo suma méritos APROBADOS (15)

```

```

71     Assert.Equal(15, data[0].PuntajeMeritos);

72

73     // - PuntajeTotal: Suma de pruebas verificadas + méritos aprobados
74     Assert.Equal(45, data[0].PuntajeTotal);

75

76     // - PruebasTotales: Total de pruebas del llamado
77     Assert.Equal(1, data[0].PruebasTotales);

78

79     // Verificar que se llamó al repositorio exactamente una vez
80     _inscripcionesMock.Verify(
81         r => r.GetByIdAsync(llamadoId),
82         Times.Once
83     );
84 }

```

Aspectos destacados del test:

1. **Patrón AAA (Arrange-Act-Assert):** El test sigue claramente las tres secciones estándar de organización de tests unitarios.
2. **Uso de mocks:** Se configuran múltiples repositorios mockeados (_inscripcionesMock, _pruebasMock, _evaluacionesPruebasMock, _evaluacionesMeritosMock) para aislar la lógica del servicio sin dependencias de base de datos.
3. **Setup específico:** Cada mock se configura con .Setup() para retornar datos controlados cuando se llama a métodos específicos, permitiendo probar escenarios predefinidos.
4. **Verificación de lógica de negocio:** El test valida reglas críticas del dominio:
 - Solo se cuentan pruebas verificadas para el puntaje final
 - Solo se cuentan méritos con estado “probado”
 - El puntaje total se calcula sumando ambos componentes

5. **Verificación de llamadas:** Se usa `Verify()` para confirmar que el repositorio fue invocado exactamente una vez, garantizando que no hay llamadas duplicadas innecesarias.
6. **Aserciones múltiples:** El test valida varios aspectos del resultado (success flag, cantidad de elementos, valores calculados) en un solo test cohesivo.

Este test es representativo de los 17 tests del `TribunalService`, donde cada uno valida aspectos específicos del flujo de evaluación de inscripciones, cálculo de puntajes, y generación de ordenamientos finales.

Herramientas de Testing

xUnit

xUnit [69] es el framework de testing elegido por su diseño moderno, extensibilidad y comunidad activa. Ventajas principales:

- Atributos simples (`[Fact]`, `[Theory]`) para marcar tests
- Aislamiento completo entre tests (cada test corre en instancia separada de la clase)
- Soporte nativo para tests asíncronos con `async/await`
- Tests parametrizados con `[InlineData]` para validar múltiples casos con un solo método
- Colecciones de tests con `[Collection]` para compartir contexto entre tests relacionados
- Sistema de fixtures (`IClassFixture<T>`) para configuración compartida

Moq

Moq es la librería de mocking más popular en .NET. Permite crear implementaciones simuladas (mocks) de interfaces y clases abstractas sin necesidad de escribir código de stub manualmente:

```

1 // Crear mock de repositorio
2 var mockRepo = new Mock<ILlamadoRepository>();
3
4 // Configurar comportamiento: cuando se llame a GetByIdAsync(1),
5 // devolver llamado de prueba
6 mockRepo.Setup(r => r.GetByIdAsync(1))
7     .ReturnsAsync(new Llamado {
8         Id = 1,
9         Titulo = "Docente de Matemática",
10        Estado = "Abierto"
11    });
12
13 // Usar mock como dependencia del servicio
14 var service = new LlamadoService(mockRepo.Object);
15
16 // Ejecutar operación que internamente llamará al repositorio
17 var resultado = await service.ObtenerLlamadoPorIdAsync(1);
18
19 // Verificar que el método se llamó exactamente una vez
20 mockRepo.Verify(r => r.GetByIdAsync(1), Times.Once);

```

Los mocks son fundamentales para:

- Aislar la lógica de negocio de dependencias externas (base de datos, APIs, sistema de archivos)
- Controlar el comportamiento de dependencias para probar escenarios específicos
- Verificar que las interacciones entre componentes ocurren según lo esperado
- Ejecutar tests rápidamente sin operaciones costosas de I/O

WebApplicationFactory (ASP.NET Core Testing)

Para las pruebas de integración, se utiliza `WebApplicationFactory<TEntryPoint>` que permite levantar un servidor web completo en memoria:

```
1 [Collection("WebApplicationFactory Collection")]
2 public class InscripcionControllerTests
3 {
4     private readonly HttpClient _client;
5
6     public InscripcionControllerTests(CustomWebApplicationFactory factory)
7     {
8         // Factory crea servidor HTTP real en memoria
9         _client = factory.CreateClient();
10    }
11
12    [Fact]
13    public async Task ObtenerInscripcion_DebeRetornarOk()
14    {
15        // Realizar solicitud HTTP real al servidor de prueba
16        var response = await _client.GetAsync("/api/incripcion/1");
17
18        // Verificar respuesta HTTP completa
19        Assert.Equal(HttpStatusCode.OK, response.StatusCode);
20    }
21 }
```

El proyecto implementa una `CustomWebApplicationFactory` personalizada que:

- Configura una base de datos SQL Server específica para tests
- Inicializa datos de prueba mediante `DbInitializer` y `TestDataSeeder`

- Configura servicios con implementaciones reales (no mocks)
- Permite ejecutar el pipeline HTTP completo incluyendo middleware, routing, validación de modelos, y persistencia

Infraestructura de Datos de Prueba

El proyecto incluye componentes especializados para gestión de datos de prueba:

- **DbInitializer**: Crea el esquema de base de datos y datos maestros (departamentos, tipos de documentos, etc.)
- **TestDataSeeder**: Genera datos de prueba realistas (llamados, postulantes, inscripciones) para tests de integración
- **WebApplicationFactoryCollection**: Permite reutilizar la misma instancia del servidor entre múltiples clases de test, optimizando el tiempo de ejecución

Ejecución y Reporte

Los tests se ejecutaron utilizando Visual Studio 2022 Enterprise, que proporciona herramientas integradas para desarrollo y análisis de pruebas:

- **Test Explorer**: Panel integrado que permite visualizar, ejecutar y debuggear tests individuales o por grupos, mostrando resultados en tiempo real con mensajes de error detallados
- **Code Coverage**: Herramienta de análisis de cobertura de código que permite medir qué porcentaje del código se ejecuta durante los tests, visualizando las líneas cubiertas y no cubiertas directamente en el editor

La cobertura de código se midió utilizando la herramienta integrada de Visual Studio 2022 Enterprise, alcanzando los siguientes niveles:

Servicio	Cobertura
TribunalService	89 %
ValidacionService	91 %
InscripcionService	87 %
PostulanteService	85 %
LlamadoService	82 %

Tabla J.2: Cobertura de código por servicio

Ejecución desde Línea de Comandos

Alternativamente, los tests pueden ejecutarse desde terminal mediante el CLI de .NET:

```
# Ejecutar todos los tests del proyecto
dotnet test

# Ejecutar con nivel de detalle verbose
dotnet test --verbosity detailed

# Ejecutar solo tests de un proyecto específico
dotnet test PortalDGC.Tests/PortalDGC.Tests.csproj
```

Métricas de Calidad

Tiempo de Ejecución

La suite completa de 152 tests unitarios se ejecuta en aproximadamente 3.2 segundos, lo que permite feedback rápido durante el desarrollo (fast feedback loop). Este tiempo rápido es posible porque:

- No se usa base de datos real (solo mocks)
- No hay operaciones I/O de red
- Cada test es completamente independiente
- xUnit paraleliza ejecución en múltiples threads

Tests Pendientes

Aunque la cobertura actual es satisfactoria para un MVP, existen áreas que requerirán tests adicionales antes de producción:

Tests de Integración

Tests que verifican interacción real con base de datos (sin mocks):

- Verificar que queries complejas con múltiples joins funcionan correctamente
- Validar que constraints de base de datos se respetan
- Probar transacciones con rollback real
- Verificar migraciones de esquema

Tests End-to-End

Tests que navegan la aplicación completa desde la UI:

- Completar flujo de inscripción desde inicio a fin
- Completar flujo de evaluación desde inicio a fin
- Generar ordenamiento y exportar a Excel/PDF
- Validar que notificaciones se disparan correctamente

Tests de Rendimiento

Tests que miden tiempos de respuesta bajo carga:

- Medir tiempo de generación de ordenamiento con 1000 inscripciones
- Medir tiempo de carga de dashboard con datos reales
- Probar escalabilidad con 500 usuarios concurrentes

Apéndice K: Plan de Validación de RNF Proyectoado

Introducción

Este apéndice documenta el plan de validación de requisitos no funcionales que deberá ejecutarse previo al despliegue en producción del Portal DGC. Si bien el MVP implementado cumple con los requisitos funcionales core y valida la arquitectura técnica, varios RNF requieren pruebas en entornos que simulen condiciones reales de operación.

Requisitos de Rendimiento

RNF-RT-01: Tiempo de Respuesta de Operaciones Críticas

Especificación: Todas las operaciones críticas del sistema deben completar en menos de 5 segundos bajo condiciones normales de carga.

Operaciones críticas identificadas:

- Autenticación de usuario
- Consulta de llamados disponibles
- Carga de detalle de llamado completo
- Envío de formulario de inscripción
- Generación de ordenamiento
- Exportación a Excel

- Exportación a PDF

Método de validación:

1. **Herramienta:** JMeter o Gatling para pruebas de carga automatizadas

2. **Escenario de prueba:**

- Configurar entorno de staging que replique infraestructura de producción
- Poblar base de datos con volumen realista (5 llamados activos, 1000 inscripciones por llamado, 50 usuarios tribunal)
- Simular carga de 100 usuarios concurrentes distribuidos en las operaciones críticas
- Ejecutar durante 30 minutos de prueba sostenida

3. **Métricas a recolectar:**

- Tiempo de respuesta promedio (avg)
- Percentil 95 (p95): el 95 % de peticiones completa en este tiempo o menos
- Percentil 99 (p99): el 99 % de peticiones completa en este tiempo o menos
- Tiempo máximo observado
- Tasa de errores (timeout, server error)

4. **Criterio de éxito:**

- $p95 < 5$ segundos para todas las operaciones críticas
- Tasa de errores $< 0.1 \%$

5. **Plan de mitigación si falla:**

- Optimizar queries SQL lentas (usar índices adicionales)
- Implementar caché de Redis para datos consultados frecuentemente (listado de llamados, departamentos)

- Implementar paginación en listados grandes
- Revisar N+1 queries en Entity Framework (usar Include apropiadamente)

RNF-RT-02: Tiempo de Respuesta de Búsquedas y Filtros

Especificación: Las operaciones de búsqueda y filtrado deben completar en menos de 3 segundos.

Operaciones de búsqueda identificadas:

- Filtro de inscripciones por departamento
- Filtro por estado de evaluación
- Búsqueda de postulante por cédula
- Filtro combinado (departamento + cupo + estado)

Método de validación: Similar a RNF-RT-01 pero con criterio de éxito más estricto (p95 <3 segundos).

Requisitos de Escalabilidad

RNF-SC-01: Usuarios Concurrentes

Especificación: El sistema debe soportar al menos 500 usuarios concurrentes sin degradación significativa de rendimiento (tiempo de respuesta <10 segundos).

Método de validación:

1. **Herramienta:** JMeter con rampa gradual de usuarios
2. **Escenario de prueba:**

- Simular 3 perfiles de usuario con comportamientos realistas:
 - a) **Postulante (70 % de usuarios):** Consulta llamados → Ve detalle → Completa inscripción (15 min)
 - b) **Tribunal (25 % de usuarios):** Ve dashboard → Filtra inscripciones → Evalúa méritos (20 min)
 - c) **RR. HH. (5 % de usuarios):** Genera reportes → Consulta estadísticas (10 min)
- Iniciar con 50 usuarios, incrementar 50 cada 5 minutos hasta llegar a 500
- Mantener 500 usuarios durante 20 minutos
- Monitorear métricas continuamente

3. Métricas a recolectar:

- Throughput (peticiones/segundo) sostenible
- Tiempos de respuesta (promedio, p95, p99)
- Utilización de CPU del servidor
- Utilización de memoria RAM
- Conexiones activas a base de datos
- Tasa de errores

4. Criterio de éxito:

- Sistema mantiene p95 <10 segundos con 500 usuarios concurrentes
- CPU <80 % utilización sostenida
- Memoria <85 % utilización
- Tasa de errores <0.5 %

5. Plan de mitigación si falla:

- Escalar horizontalmente: agregar más instancias de backend con load balancer
- Implementar connection pooling en base de datos

- Optimizar consultas lentas identificadas en profiling
- Implementar caché distribuido (Redis)

Requisitos de Seguridad

RNF-SG-01: Comunicación Segura (HTTPS)

Especificación: Toda comunicación cliente-servidor debe realizarse mediante HTTPS con certificado institucional .gub.uy válido.

Método de validación:

1. Verificación de certificado:

- Solicitar certificado SSL institucional a autoridad certificadora reconocida
- Configurar certificado en servidor web (IIS o Nginx)
- Verificar con herramienta SSL Labs que el certificado es válido y confiable
- Verificar que el navegador muestra “candado verde” sin advertencias

2. Verificación de redirección:

- Intentar acceder via HTTP (<http://portal-dgc.gub.uy>)
- Verificar que el servidor redirige automáticamente a HTTPS (301 Permanent Redirect)
- Verificar que no hay contenido mixto (mixed content) en páginas

3. Verificación de protocolos:

- Verificar que solo se permiten protocolos seguros (TLS 1.2+)
- Deshabilitar protocolos obsoletos (SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1)
- Configurar cipher suites fuertes (AES-256, ChaCha20)

RNF-SG-02: Control de Acceso Basado en Roles

Especificación: El sistema debe implementar control de acceso mediante roles (Postulante, Tribunal, RR. HH., Admin) con permisos específicos para cada rol.

Método de validación:

1. **Matriz de permisos:** Documentar matriz completa de permisos:

Operación	Postulante	Tribunal	RR. HH.	Admin
Ver llamados públicos	✓	✓	✓	✓
Inscribirse	✓	✗	✗	✗
Ver mis inscripciones	✓	✗	✗	✓
Evaluar méritos	✗	✓	✗	✓
Generar ordenamiento	✗	✓	✗	✓
Crear llamado	✗	✗	✓	✓
Gestionar usuarios	✗	✗	✗	✓

2. **Tests de autorización:** Para cada endpoint protegido, verificar:

- Usuario sin autenticar recibe HTTP 401 (Unauthorized)
- Usuario autenticado sin rol apropiado recibe HTTP 403 (Forbidden)
- Usuario con rol apropiado accede exitosamente (HTTP 200)

3. **Pruebas de bypass:** Intentar eludir controles:

- Modificar token JWT manualmente (debe fallar validación de firma)
- Cambiar rol en cookie del cliente (debe ignorarse, rol viene del token)
- Acceder directamente a URL de endpoint protegido sin token (debe rechazar)

RNF-SG-03: Encriptación de Datos Sensibles

Especificación: Datos sensibles (autodefinición étnico-racial, identidad de género, discapacidad, documentos adjuntos) deben encriptarse en reposo usando AES-256.

Método de validación:

1. Verificación en base de datos:

- Ejecutar query directo a SQL Server para columnas sensibles
- Verificar que el valor almacenado no es legible (está encriptado)
- Intentar desencriptar sin clave (debe fallar)

2. Verificación de archivos:

- Navegar al directorio donde se almacenan archivos adjuntos
- Verificar que archivos tienen nombres ofuscados (UUID)
- Verificar que archivos están encriptados (no se pueden abrir directamente)
- Verificar que solo se desencriptan cuando usuario autorizado los descarga

3. Gestión de claves:

- Verificar que claves de encriptación NO están en código fuente
- Verificar que claves se almacenan en Azure Key Vault o HSM
- Verificar que claves rotan periódicamente (cada 90 días)

Requisitos de Usabilidad

RNF-UB-01: Flujo de Inscripción en Máximo 6 Pasos

Especificación: El workflow de inscripción debe completarse en máximo 6 pasos lógicos.

Método de validación:

Este requisito se validó durante el desarrollo del prototipo y del MVP. El workflow implementado tiene exactamente 6 pasos:

1. Selección de Departamento
2. Autodefinition
3. Requisitos Excluyentes
4. Méritos
5. Apoyos (condicional)
6. Confirmación

Estado: VALIDADO ✓

RNF-UB-02: Accesibilidad WCAG 2.1 Nivel AA

Especificación: El sistema debe cumplir con las pautas de accesibilidad WCAG 2.1 nivel AA.

Método de validación:

1. Auditoría automática con herramientas:

- **axe DevTools:** Extensión de navegador que detecta problemas automáticamente
- **WAVE:** Herramienta online de WebAIM
- **Lighthouse:** Herramienta integrada en Chrome DevTools

Ejecutar estas herramientas en todas las páginas principales y verificar que no reportan errores críticos de nivel A o AA.

2. Revisión manual de criterios WCAG:

Criterio 1.4.3 - Contraste (Nivel AA):

- Texto normal: Relación de contraste $\geq 4.5:1$
- Texto grande: Relación de contraste $\geq 3:1$
- Herramienta: Colour Contrast Analyser

Criterio 2.1.1 - Teclado (Nivel A):

- Todas las funcionalidades deben ser operables mediante teclado
- Probar navegación con Tab (avanzar), Shift+Tab (retroceder), Enter (activar)
- Verificar que el orden de tabulación es lógico
- Verificar que el foco visual es claramente visible

Criterio 1.3.1 - Info y Relaciones (Nivel A):

- Formularios: Todos los inputs tienen `<label>` asociado
- Tablas: Usar `<th>` para encabezados, atributo scope apropiado
- Listas: Usar ``, ``, `` según corresponda
- Landmarks: Usar `<header>`, `<nav>`, `<main>`, `<footer>`

Criterio 4.1.2 - Nombre, Función, Valor (Nivel A):

- Componentes interactivos tienen atributos ARIA apropiados
- `role="button"` en divs clickeables
- `aria-label` en iconos sin texto visible
- `aria-describedby` en campos con instrucciones adicionales

3. Pruebas con tecnologías asistivas:

- **NVDA (Windows):** Lector de pantalla gratuito

- **JAWS (Windows):** Lector de pantalla comercial más usado
- **VoiceOver (macOS/iOS):** Lector de pantalla nativo de Apple

Navegar el sitio completo con cada lector de pantalla y verificar:

- Todos los textos se leen correctamente
- Los encabezados permiten navegar por secciones (H1, H2, etc.)
- Los formularios son comprensibles (labels claros)
- Los mensajes de error son anunciados
- Los cambios dinámicos se notifican (aria-live regions)

4. Pruebas con usuarios reales:

Reclutar 3-5 usuarios con diferentes tipos de discapacidad:

- Personas ciegas (dependencia total de lector de pantalla)
- Personas con baja visión (magnificador de pantalla, contraste alto)
- Personas con limitaciones motrices (solo teclado, sin mouse)
- Personas con discapacidad cognitiva (simplicidad del lenguaje)

Solicitarles completar tareas específicas:

- Consultar un llamado disponible
- Completar formulario de inscripción
- Verificar estado de postulación

Observar dificultades encontradas y priorizar correcciones.

5. Criterio de éxito:

- 0 errores críticos detectados por herramientas automáticas
- 100 % de criterios WCAG 2.1 nivel A cumplidos

- 100 % de criterios WCAG 2.1 nivel AA cumplidos
- Usuarios con discapacidad pueden completar tareas sin asistencia externa

6. Plan de mitigación si falla:

- Priorizar corrección de errores detectados por herramientas (bajo esfuerzo)
- Consultar con especialistas en accesibilidad para casos ambiguos
- Revisar documentación WCAG 2.1 para criterios específicos que fallan
- Implementar iterativamente y re-validar después de cada cambio

Requisitos de Fiabilidad

RNF-FI-01: Disponibilidad 99.5 %

Especificación: El sistema debe mantener disponibilidad de 99.5 % mensual (downtime máximo de 3.6 horas/mes).

Método de validación:

1. Monitoreo continuo: Implementar herramienta de monitoreo (Nagios, Zabbix, o Azure Monitor) que verifique disponibilidad cada minuto mediante:

- HTTP health check endpoint (/api/health)
- Ping a servidor
- Verificación de conectividad a base de datos

2. Métricas a recolectar durante 3 meses:

- Uptime total (minutos operativo / minutos totales)
- Cantidad de incidentes de indisponibilidad
- Duración promedio de incidentes

- MTBF (Mean Time Between Failures)
- MTTR (Mean Time To Repair)

3. Criterio de éxito:

- Uptime \geq 99.5 % cada mes
- Incidentes no planificados <3 por mes
- MTTR <30 minutos

4. Plan de mitigación si falla:

- Implementar arquitectura de alta disponibilidad (HA) con redundancia
- Load balancer con al menos 2 servidores backend activos
- Base de datos con replicación (master-slave o Always On)
- Monitoreo proactivo con alertas automáticas
- Plan de respuesta a incidentes documentado

RNF-FI-02: Backups Diarios

Especificación: El sistema debe realizar backups automáticos diarios de la base de datos con retención de 30 días.

Método de validación:

1. Configuración de backups:

- Configurar SQL Server Maintenance Plan o Azure SQL automated backups
- Programar backup completo diario a las 2:00 AM
- Programar backups diferenciales cada 6 horas
- Programar backups de log de transacciones cada hora

2. Verificación de ejecución:

- Verificar que job de backup se ejecuta exitosamente cada día
- Revisar logs de SQL Server para confirmar que no hay errores
- Verificar que archivos de backup se generan correctamente
- Verificar que archivos mayores a 30 días se eliminan automáticamente

3. Prueba de restauración:

- Crear base de datos de prueba
- Restaurar backup más reciente
- Verificar integridad de datos (DBCC CHECKDB)
- Ejecutar queries de validación en datos críticos
- Documentar tiempo de restauración (debe ser <2 horas)

4. Criterio de éxito:

- Backups se ejecutan exitosamente 100 % de los días
- Restauración de backup funciona sin pérdida de datos
- RTO (Recovery Time Objective) <2 horas
- RPO (Recovery Point Objective) <24 horas

Requisitos de Portabilidad

RNF-PT-01: Compatibilidad con Navegadores

Especificación: El sistema debe funcionar correctamente en las últimas dos versiones de Chrome, Firefox, Edge y Safari.

Método de validación:

1. Navegadores a probar:

- Google Chrome (últimas 2 versiones)
- Mozilla Firefox (últimas 2 versiones)
- Microsoft Edge (últimas 2 versiones)
- Safari (últimas 2 versiones) - requiere macOS

2. Pruebas funcionales en cada navegador:

- Completar inscripción a llamado
- Evaluar postulante (calificar méritos y pruebas)
- Generar y exportar ordenamiento
- Verificar que formularios validan correctamente
- Verificar que modales se abren/cierran correctamente
- Verificar que filtros funcionan

3. Pruebas visuales:

- Verificar que layout no se rompe (no hay overlaps, scrolls extraños)
- Verificar que colores se muestran correctamente
- Verificar que iconos se renderizan
- Verificar que fuentes se cargan

4. Herramientas de testing:

- BrowserStack o Sauce Labs para testing en múltiples navegadores sin necesidad de máquinas virtuales
- Selenium WebDriver para automatizar pruebas cross-browser

5. Criterio de éxito:

- Todas las funcionalidades operan correctamente en todos los navegadores

- Layout es consistente (permitiendo diferencias menores de rendering)
- No se requieren workarounds específicos por navegador

Requisitos de Interoperabilidad

RNF-IN-01: API REST con JSON

Especificación: El sistema debe exponer API REST que consuma/producza JSON para permitir integración con sistemas externos.

Método de validación:

1. Verificación de especificación OpenAPI:

- Generar documentación OpenAPI 3.0 (Swagger) de la API
- Verificar que todos los endpoints están documentados
- Verificar que todos los modelos (DTOs) están documentados
- Verificar que códigos de respuesta están documentados

2. Pruebas de integración:

- Desarrollar cliente de prueba en lenguaje diferente (Python, JavaScript, Java)
- Verificar que cliente puede autenticarse
- Verificar que cliente puede consumir endpoints principales
- Verificar que JSON devuelto es parseable sin errores

3. Validación de contrato:

- Usar herramienta Postman para validar cada endpoint
- Verificar que estructura JSON es consistente

- Verificar que campos obligatorios están presentes
- Verificar que tipos de datos son correctos (string, number, boolean, etc.)
- Verificar que formatos son estándar (ISO 8601 para fechas, etc.)

4. Criterio de éxito:

- Documentación OpenAPI completa y precisa
- Cliente externo puede integrar exitosamente consumiendo la API
- JSON es válido según RFC 7159
- API sigue principios REST (recursos, verbos HTTP apropiados, HATEOAS)

Requisitos de Mantenibilidad

RNF-MT-01: Cobertura de Tests >= 85 %

Especificación: El código debe mantener cobertura de tests unitarios >= 85 % en servicios críticos.

Método de validación:

Este requisito se validó durante el desarrollo. La cobertura actual es:

- TribunalService: 89 %
- InscripcionService: 87 %
- ValidacionService: 91 %
- PostulanteService: 85 %
- LlamadoService: 82 %
- Promedio: 87 %

Estado: VALIDADO ✓(superó el objetivo de 85 %)

RNF-MT-02: Documentación Completa

Especificación: El sistema debe incluir documentación técnica completa (ESRE, casos de uso, diagramas arquitectónicos, comentarios inline).

Método de validación:

Este requisito se cumplió durante el desarrollo. La documentación incluye:

- ESRE completo con 21 RF y 16 RNF
- 20 casos de uso documentados con plantilla Cockburn
- Diagramas de arquitectura, flujos, modelo de dominio
- XML comments en 100 % de clases públicas de C#
- JSDoc en 80 % de servicios TypeScript
- README con instrucciones de setup y ejecución

Estado: VALIDADO ✓

Cronograma de Validación Pre-Producción

La siguiente tabla propone un cronograma estimado para ejecutar todas las validaciones previo al despliegue:

Conclusión

Este plan de validación proporciona una hoja de ruta clara para completar las validaciones de requisitos no funcionales que no pudieron ejecutarse completamente durante el desarrollo del

Actividad de Validación	Duración	Recurso Requerido
Pruebas de rendimiento (RT-01, RT-02)	1 semana	Especialista QA + Infra
Pruebas de escalabilidad (SC-01)	1 semana	Especialista QA + Infra
Configuración HTTPS (SG-01)	3 días	Administrador Infra
Implementación RBAC completo (SG-02)	2 semanas	Desarrollador Backend
Implementación encriptación (SG-03)	1 semana	Desarrollador Backend
Auditoría accesibilidad automatizada (UB-02)	2 días	Especialista UX
Pruebas con lectores de pantalla (UB-02)	1 semana	Especialista Accesibilidad
Pruebas con usuarios con discapacidad (UB-02)	1 semana	Facilitador + Usuarios
Configuración monitoreo (FI-01)	3 días	Administrador Infra
Configuración backups (FI-02)	2 días	Administrador BD
Prueba de restauración (FI-02)	1 día	Administrador BD
Testing cross-browser (PT-01)	3 días	Tester QA
Total estimado	8-10 semanas	

Tabla K.1: Cronograma estimado de validación pre-producción

MVP. La ejecución sistemática de estas pruebas es crítica para garantizar que el sistema cumple con estándares de calidad, seguridad, y usabilidad requeridos para un despliegue en producción en el ámbito gubernamental.

Cada validación incluye método detallado, herramientas recomendadas, criterios de éxito claros, y planes de mitigación si las pruebas detectan problemas. Esta estructura permite a futuros equipos de trabajo ejecutar las validaciones de manera ordenada y documentar resultados de forma trazable.

El cronograma estimado de 8-10 semanas representa el esfuerzo adicional requerido para transformar el MVP en un sistema production-ready con un equipo de mas integrantes. Este tiempo se distribuye entre trabajo de desarrollo (implementación de seguridad), configuración de infraestructura (HTTPS, backups, monitoreo), y testing exhaustivo (rendimiento, accesibilidad,

cross-browser).

Apéndice L: Trabajo Pendiente Priorizado

Introducción

Este apéndice documenta las funcionalidades, mejoras y tareas que quedaron pendientes en el MVP y que deben completarse antes de un despliegue en producción. Las tareas están priorizadas en tres niveles: CRÍTICO (bloqueante para producción), ALTA (necesario antes del lanzamiento), y MEDIA (deseable pero no bloqueante).

Funcionalidades CRÍTICAS

Autenticación y Autorización

RF-01: Autenticación JWT con Refresh Tokens

Estado actual: El MVP implementa autenticación simulada que valida usuario/contraseña pero no genera tokens JWT reales.

Trabajo requerido:

1. Implementar generación de Access Token (JWT) con expiración corta (15 minutos)
2. Implementar generación de Refresh Token con expiración larga (7 días)
3. Crear endpoint POST /api/auth/refresh para renovar tokens
4. Almacenar Refresh Tokens en base de datos con fecha de expiración
5. Implementar revocación de tokens (logout, cambio de contraseña)
6. Configurar middleware de validación de JWT en backend

7. Implementar almacenamiento seguro de tokens en frontend (httpOnly cookies o sessionStorage)
8. Implementar renovación automática de token cuando expira

Estimación: 2 semanas (2 desarrollador backend + 1 frontend)

Dependencias: Ninguna

RF-18: Sistema RBAC de Roles y Permisos

Estado actual: Los roles están hardcoded en el código. No hay gestión dinámica de permisos.

Trabajo requerido:

1. Crear modelo de datos para Roles y Permisos en base de datos
2. Definir matriz de permisos completa (ver Apéndice K)
3. Implementar middleware de autorización basado en permisos
4. Crear tabla de asociación Usuario-Rol (un usuario puede tener múltiples roles)
5. Crear tabla de asociación Rol-Permiso (un rol tiene múltiples permisos)
6. Implementar atributos de autorización en controllers: [Authorize(Permission = ".EVALUAR_MERITOS")]
7. Implementar UI condicional en frontend basada en permisos del usuario actual
8. Crear panel administrativo para asignar roles a usuarios

Estimación: 2 semanas (2 desarrollador backend + 1 frontend)

Dependencias: RF-01 (requiere autenticación JWT funcionando)

Seguridad de Infraestructura

RNF-SG-01: HTTPS con Certificado Institucional

Estado actual: El MVP corre sobre HTTP sin encriptación.

Trabajo requerido:

1. Solicitar certificado SSL institucional .gub.uy a autoridad certificadora
2. Configurar certificado en servidor web (IIS, Nginx, o Azure App Service)
3. Configurar redirección automática de HTTP a HTTPS (301 redirect)
4. Habilitar HSTS (HTTP Strict Transport Security) headers
5. Deshabilitar protocolos obsoletos (TLS 1.0, TLS 1.1)
6. Configurar cipher suites fuertes
7. Verificar con SSL Labs que configuración obtiene calificación A

Estimación: 1 semana (administrador de infraestructura)

Dependencias: Aprobación institucional para solicitud de certificado

RNF-SG-03: Encriptación de Datos Sensibles en Reposo

Estado actual: Datos sensibles se almacenan en texto plano en base de datos.

Trabajo requerido:

1. Implementar servicio de encriptación usando AES-256
2. Integrar con Azure Key Vault para almacenamiento seguro de claves

3. Encriptar columnas sensibles en base de datos:
 - Autodeficion.EsAfrodescendiente
 - Autodeficion.EsTrans
 - Autodeficion.TieneDiscapacidad
 - ApoyoSolicitado.Descripcion
4. Implementar rotación automática de claves cada 90 días
5. Encriptar archivos adjuntos en file system
6. Modificar queries para desencriptar transparentemente al leer

Estimación: 1.5 semanas (2 desarrolladores backend)

Dependencias: Configuración de Azure Key Vault en ambiente productivo

Almacenamiento de Archivos

RF-06: Almacenamiento Escalable de Archivos

Estado actual: Archivos se almacenan en file system local del servidor, no escalable ni respaldado.

Trabajo requerido:

1. Configurar Azure Blob Storage o AWS S3
2. Implementar ArchivoService para subir archivos a storage cloud
3. Implementar generación de URLs firmadas (SAS tokens) con expiración
4. Modificar frontend para usar URLs firmadas en descarga
5. Migrar archivos existentes de file system a cloud storage

6. Implementar política de ciclo de vida (eliminar archivos huérfanos después de X días)

Estimación: 1 semana (1 desarrollador backend + 1 devops)

Dependencias: Cuenta de Azure Storage o AWS configurada

Infraestructura de Alta Disponibilidad

RNF-FI-01: Monitoreo de Disponibilidad

Estado actual: No hay monitoreo de disponibilidad ni alertas.

Trabajo requerido:

1. Implementar endpoint /api/health que verifique:

- Aplicación responde (HTTP 200)
- Conectividad a base de datos
- Conectividad a storage de archivos
- Espacio en disco disponible

2. Configurar Azure Monitor, AWS CloudWatch, o Nagios

3. Configurar health checks cada minuto

4. Configurar alertas cuando disponibilidad <100 % durante 5 minutos

5. Configurar notificaciones por email/SMS a equipo técnico

6. Crear dashboard de monitoreo en tiempo real

Estimación: 1 semana (1 administrador de infraestructura)

Dependencias: Ambiente productivo configurado

RNF-FI-02: Backups Automatizados

Estado actual: No hay backups automatizados configurados.

Trabajo requerido:

1. Configurar SQL Server Maintenance Plan o Azure SQL automated backups
2. Programar backup completo diario (2:00 AM)
3. Programar backups diferenciales cada 6 horas
4. Programar backups de log de transacciones cada hora
5. Configurar retención de 30 días
6. Configurar almacenamiento de backups en ubicación geográfica separada
7. Documentar procedimiento de restauración
8. Realizar prueba de restauración mensual

Estimación: 1 semana (1 administrador de base de datos)

Dependencias: Ambiente productivo configurado

Funcionalidades de ALTA Prioridad

Sistema de Notificaciones

RF-16: Notificaciones por Email y WhatsApp

Estado actual: No hay sistema de notificaciones implementado.

Trabajo requerido:

1. Integrar con servicio SMTP para envío de emails (SendGrid, AWS SES, o servidor institucional)
2. Integrar con WhatsApp Business API
3. Diseñar templates de notificaciones:
 - Confirmación de inscripción
 - Cambio de estado de inscripción
 - Resultados publicados
 - Recordatorio de pruebas próximas
4. Implementar cola de mensajes (RabbitMQ, Azure Service Bus) para envío asíncrono
5. Implementar servicio worker que procese cola
6. Implementar reintentos automáticos si envío falla
7. Crear tabla de auditoría de notificaciones enviadas
8. Implementar preferencias de usuario (habilitar/deshabilitar notificaciones)

Estimación: 2 semanas (2 desarrolladores backend + 1 frontend)

Dependencias: Configuración de cuentas de SendGrid/WhatsApp Business

Módulo de Administración RR. HH.

RF-21: Creación y Gestión de Llamados

Estado actual: Llamados se insertan manualmente mediante scripts SQL.

Trabajo requerido:

1. Crear interfaz de creación de llamado con workflow:

- Paso 1: Información general (título, descripción, fechas, plazas)
 - Paso 2: Requisitos excluyentes (agregar/eliminar requisitos)
 - Paso 3: Méritos valorables (definir categorías, puntajes máximos)
 - Paso 4: Pruebas (definir tipo, puntaje máximo, umbral aprobación)
 - Paso 5: Cuotas (configurar porcentajes afro, trans, discapacidad)
 - Paso 6: Departamentos disponibles
 - Paso 7: Cronograma de etapas
 - Paso 8: Revisión y publicación
2. Implementar validaciones de datos ingresados
 3. Implementar flujo de aprobación (borrador → revisión → publicado)
 4. Implementar edición de llamado (solo si no tiene inscripciones)
 5. Implementar cierre manual de llamado
 6. Implementar duplicación de llamado existente como plantilla

Estimación: 3 semanas (2 desarrolladores full-stack)

Dependencias: RF-18 (requiere permisos RR. HH. implementados)

Auditoría y Trazabilidad

RF-19: Panel de Auditoría de Acciones

Estado actual: No hay logging estructurado de acciones de usuarios.

Trabajo requerido:

1. Implementar middleware de logging que capture:

- Usuario que realiza acción
 - Timestamp de acción
 - Endpoint invocado
 - Parámetros de petición
 - Resultado (éxito/error)
 - IP de origen
2. Crear tabla AuditLog en base de datos
 3. Implementar política de retención (guardar logs por 5 años)
 4. Crear interfaz de consulta de auditoría con filtros:
 - Por usuario
 - Por fecha
 - Por tipo de acción
 - Por entidad afectada
 5. Implementar exportación de logs a CSV
 6. Configurar alertas de acciones sospechosas (múltiples fallos de login, acceso fuera de horario)

Estimación: 1.5 semanas (2 desarrollador backend + 1 frontend)

Dependencias: RF-18 (requiere permisos Admin implementados)

Accesibilidad

RNF-UB-02: Auditoría Completa de Accesibilidad

Estado actual: Se implementaron bases técnicas de accesibilidad pero no se validó con usuarios reales.

Trabajo requerido:

1. Ejecutar auditoría automática .
2. Corregir todos los errores críticos detectados
3. Realizar pruebas con lectores de pantalla.
4. Contratar facilitador de accesibilidad para reclutar usuarios con discapacidad
5. Realizar sesiones de testing con 3-5 usuarios:
 - Personas ciegas
 - Personas con baja visión
 - Personas con limitaciones motrices
6. Documentar problemas encontrados
7. Priorizar y corregir problemas bloqueantes
8. Re-validar después de correcciones
9. Obtener certificación de cumplimiento WCAG 2.1 AA (opcional pero recomendado)

Estimación: 2 semanas (2 desarrolladores frontend + 1 especialista accesibilidad + usuarios)

Dependencias: Presupuesto para contratar facilitador y compensar a usuarios

Funcionalidades de MEDIA Prioridad

Reportes y Estadísticas

RF-20: Reportes Estadísticos Avanzados

Estado actual: Solo existen métricas básicas en dashboard del tribunal.

Trabajo requerido:

1. Diseñar reportes analíticos:
 - Tasas de aprobación por departamento
 - Distribución de puntajes
 - Efectividad de aplicación de cuotas (cuántas posiciones reservadas se llenaron)
 - Comparación entre llamados
2. Implementar generación de gráficos
3. Implementar filtros de fecha (último mes, último año, período personalizado)
4. Implementar exportación de reportes a PDF y Excel [29]

Estimación: 2 semanas (2 desarrolladores full-stack)

Dependencias: RF-16 (para envío automático de reportes por email)

Gestión de Usuarios

Estado actual: No hay interfaz para dar de alta/baja usuarios.

Trabajo requerido:

1. Crear panel de gestión de usuarios:
 - Lista de usuarios con filtros
 - Crear nuevo usuario
 - Editar usuario existente
 - Desactivar/activar usuario
 - Asignar roles

- Resetear contraseña
2. Implementar validación de datos (email único, cédula única)
 3. Implementar envío de email con credenciales a nuevo usuario
 4. Implementar expiración de contraseñas cada 90 días
 5. Implementar política de contraseña fuerte (mínimo 8 caracteres, mayúsculas, números, símbolos)

Estimación: 1 semana (2 desarrolladores full-stack)

Dependencias: RF-18 (requiere permisos Admin), RF-16 (para envío de email)

Recuperación de Contraseña

Estado actual: No hay flujo de recuperación de contraseña.

Trabajo requerido:

1. Crear pantalla “olvidé mi contraseña”
2. Implementar generación de token de recuperación (válido por 1 hora)
3. Enviar email con link de recuperación
4. Crear pantalla de reseteo de contraseña
5. Validar token antes de permitir cambio
6. Invalidar token después de uso
7. Implementar límite de intentos (máximo 3 por día)

Estimación: 1 semana (2 desarrolladores full-stack)

Dependencias: RF-16 (requiere envío de email)

Configuración de Parámetros del Sistema

Estado actual: Parámetros están hardcoded en código.

Trabajo requerido:

1. Crear tabla de configuración en base de datos
2. Mover parámetros configurables a tabla:
 - Umbral de aprobación de pruebas (actualmente 30 %)
 - Porcentaje de cuotas (actualmente 8 % afro, 1 % trans, 4 % discapacidad)
 - Plazo de inscripción por defecto
 - Tamaño máximo de archivos adjuntos
3. Crear panel de configuración para Admin
4. Implementar caché de configuración en memoria (refrescar cada 5 minutos)
5. Implementar auditoría de cambios de configuración

Estimación: 1 semana (2 desarrolladores full-stack)

Dependencias: RF-18 (requiere permisos Admin)

Gestión de Catálogos

Estado actual: Catálogos (departamentos, tipos de pruebas) están en base de datos pero sin interfaz de gestión.

Trabajo requerido:

1. Crear CRUD de departamentos

2. Crear CRUD de tipos de pruebas
3. Crear CRUD de apoyos disponibles
4. Implementar habilitación/deshabilitación sin eliminar registros
5. Implementar ordenamiento de ítems en catálogos

Estimación: 1 semana (1 desarrollador full-stack)

Dependencias: RF-18 (requiere permisos Admin)

Pruebas Adicionales Requeridas

Pruebas de Carga

Trabajo requerido: Ver Apéndice K sección de RNF-RT-01 y RNF-SC-01.

Estimación: 2 semanas (1 especialista QA + 1 devops)

Pruebas de Seguridad

Trabajo requerido:

1. Ejecutar scan con OWASP ZAP para detectar vulnerabilidades web comunes:
 - SQL Injection
 - Cross-Site Scripting (XSS)
 - Cross-Site Request Forgery (CSRF)
 - Insecure Direct Object References
 - Security Misconfiguration
2. Revisar reporte y corregir vulnerabilidades detectadas

3. Ejecutar scan con Nessus o similar para vulnerabilidades de infraestructura
4. Realizar pruebas de penetración manual (pen testing)
5. Documentar hallazgos y plan de remediación

Estimación: 2 semanas (1 especialista en seguridad)

Dependencias: Ambiente productivo o staging configurado

Pruebas End-to-End Automatizadas

Trabajo requerido:

1. Configurar framework de testing E2E (Selenium [116])
2. Implementar tests de flujos críticos:
 - Flujo completo de inscripción
 - Flujo completo de evaluación
 - Generación de ordenamiento y exportación
3. Configurar ejecución automática en CI/CD
4. Configurar captura de screenshots/videos en caso de fallo

Estimación: 2 semanas (1 QA automation engineer)

Dependencias: Ninguna (puede hacerse en paralelo)

Resumen de Estimaciones

Nota: Las estimaciones asumen equipo de 2-3 personas trabajando en paralelo. El tiempo de calendario real dependerá de disponibilidad de recursos y priorización.

Categoría	Tareas	Estimación Total
Funcionalidades CRÍTICAS	6 tareas	9 semanas
Funcionalidades ALTA prioridad	4 tareas	8 semanas
Funcionalidades MEDIA prioridad	5 tareas	7 semanas
Pruebas adicionales	3 áreas	6 semanas
TOTAL	18 tareas	30 semanas

Tabla L.1: Resumen de trabajo pendiente

Roadmap Sugerido

Fase 1: Seguridad y Estabilidad (8-10 semanas)

Prioridad CRÍTICA. Sin estas funcionalidades, el sistema no puede desplegarse en producción.

1. Autenticación JWT (RF-01)
2. Sistema RBAC (RF-18)
3. HTTPS (RNF-SG-01)
4. Encriptación datos sensibles (RNF-SG-03)
5. Almacenamiento archivos cloud (RF-06)
6. Monitoreo disponibilidad (RNF-FI-01)
7. Backups automatizados (RNF-FI-02)

Fase 2: Funcionalidades Esenciales (6-8 semanas)

Prioridad ALTA. Funcionalidades muy valoradas por usuarios pero no bloqueantes técnicamente.

1. Sistema de notificaciones (RF-16)
2. Módulo RR. HH. creación llamados (RF-21)
3. Panel de auditoría (RF-19)
4. Auditoría de accesibilidad (RNF-UB-02)

Fase 3: Mejoras y Optimizaciones (4-6 semanas)

Prioridad MEDIA. Mejoras que aportan valor pero pueden posponerse si hay restricciones de tiempo/presupuesto.

1. Reportes estadísticos avanzados (RF-20)
2. Gestión de usuarios
3. Recuperación de contraseña
4. Configuración de parámetros
5. Gestión de catálogos

Fase 4: Testing y Validación (4-6 semanas)

Pruebas exhaustivas antes del lanzamiento. Puede ejecutarse en paralelo con Fase 3.

1. Pruebas de carga
2. Pruebas de seguridad
3. Pruebas E2E automatizadas

Timeline total estimado: 22-30 semanas (6 - 8 meses)

Conclusión

Este documento proporciona visión completa y realista del trabajo que resta para transformar el MVP en un sistema production-ready. Si bien la lista es extensa, es importante destacar que el MVP ya validó los conceptos fundamentales y la arquitectura técnica es sólida. El trabajo pendiente es mayormente de consolidación, seguridad, y pulido, no de diseño o replanteamiento estructural.

La priorización clara permite a los stakeholders tomar decisiones informadas sobre qué desarrollar primero en función de restricciones de tiempo y presupuesto. Un despliegue mínimo viable en producción requeriría completar al menos la Fase 1 (seguridad y estabilidad) antes de permitir acceso público al sistema.

Apéndice M: Plan de pruebas

Introducción

Este apéndice documenta el plan de pruebas funcionales elaborado por un especialista en Quality Assurance para validar el MVP del Portal DGC. El plan contiene 14 casos de prueba diseñados para verificar que los flujos implementados funcionan según lo especificado y cumplen con los requisitos funcionales del sistema.

A diferencia del plan de validación de RNF (Apéndice K) que se enfoca en aspectos no funcionales y se ejecutará previo al despliegue en producción, este plan se centra en la validación funcional del MVP actual y está diseñado para ejecutarse en el corto plazo.

Estructura de Casos de Prueba

Cada caso de prueba sigue la siguiente estructura:

- **Screen:** Pantalla o módulo donde se ejecuta la prueba
- **Description:** Descripción breve de la funcionalidad a validar
- **Steps to Reproduce:** Pasos detallados para ejecutar la prueba
- **Expected Result:** Resultado esperado si el sistema funciona correctamente
- **Actual Result:** Campo para documentar el resultado real durante ejecución (pendiente)
- **Status:** Estado de la prueba (Pass/Fail) (pendiente)

Casos de Prueba

CP-01: Login

Descripción: Ingreso exitoso de un usuario registrado.

Pasos de Reproducción:

1. Abrir la página de login.
2. Ingresar email y contraseña válidos.
3. Pulsar 'Ingresar'.

Resultado Esperado: Login completado y redirección al panel; sesión iniciada correctamente.

CP-02: Registro

Descripción: Registro de postulante y confirmación por correo.

Pasos de Reproducción:

1. Ir a formulario de registro.
2. Completar Nombre, CI, correo, contraseña y teléfono.
3. Enviar registro.
4. Abrir correo de confirmación y activar cuenta.

Resultado Esperado: Cuenta creada; correo de confirmación recibido; cuenta activada al clicar el link.

CP-03: Listado de llamados

Descripción: Visualizar listados de llamados publicados.

Pasos de Reproducción:

1. Entrar a 'Llamados'.
2. Aplicar un filtro simple (ej. departamento).
3. Revisar resultados.

Resultado Esperado: Muestra llamados vigentes con título, fechas y link a detalle.

CP-04: Detalle Llamado

Descripción: Ver detalle de un llamado y descargar bases en PDF.

Pasos de Reproducción:

1. Desde listado, abrir un llamado.
2. Revisar requisitos y cuotas.
3. Pulsar 'Descargar bases'.

Resultado Esperado: Descripción completa visible; descarga del PDF realizada y abre correctamente.

CP-05: Crear y publicar llamado (Backoffice)

Descripción: RRHH crea un llamado, lo guarda y publica.

Pasos de Reproducción:

1. Acceder a backoffice con rol RRHH.
2. Nuevo llamado: completar título, fechas, cargar PDF y cupos.
3. Guardar borrador y luego publicar.

Resultado Esperado: Llamado guardado como borrador y posteriormente publicado; visible para postulantes.

CP-06: Postulación - Guardar borrador y Enviar

Descripción: Postulante inicia postulación, guarda y finalmente envía.

Pasos de Reproducción:

1. Elegir llamado → 'Postularse'.
2. Completar datos personales y experiencia.
3. Guardar como borrador.
4. Abrir borrador y 'Enviar postulación'.

Resultado Esperado: Postulación registrada; código único generado; estado 'enviado'.

CP-07: Carga de documentos PDF

Descripción: Adjuntar CV y documentos en formato PDF válidos.

Pasos de Reproducción:

1. En postulación, ir a 'Documentos'.
2. Cargar CV.pdf (<10MB).
3. Ver previsualización y reemplazar por otro PDF válido.

Resultado Esperado: Solo .pdf aceptados; archivos almacenados y previsualización disponible.

CP-08: Confirmación postulación (correo/panel)

Descripción: Confirmación de recepción de postulación vía correo y panel.

Pasos de Reproducción:

1. Enviar postulación.
2. Revisar correo de pruebas y panel de usuario.

Resultado Esperado: Correo con identificador enviado; identificador visible en panel; coherencia entre ambos.

CP-09: Declaración cupos especiales

Descripción: Postulante declara pertenecer a un cupo y adjunta soporte.

Pasos de Reproducción:

1. En formulario, marcar pertenencia a cupo.
2. Adjuntar documento soporte en PDF.
3. Enviar postulación.

Resultado Esperado: Declaración grabada con documento adjunto; clasificación correcta en el sistema.

CP-10: Notificaciones automáticas

Descripción: Envío de notificaciones por correo y WhatsApp si autorizado.

Pasos de Reproducción:

1. Trigger: enviar postulación o publicar convocatoria.
2. Revisar logs de envío y buzón/número de pruebas.

Resultado Esperado: Registro de envío y entrega del mensaje (o log con entrega en staging).

CP-11: Preselección automática

Descripción: Ejecución de proceso de preselección y exportación.

Pasos de Reproducción:

1. Ejecutar preselección desde backoffice una vez cerrado el llamado.
2. Revisar lista de elegibles.
3. Exportar CSV.

Resultado Esperado: Lista de elegibles generada y CSV descargable.

CP-12: Ordenamiento con semilla

Descripción: Sorteo digital con semilla y publicación de resultados.

Pasos de Reproducción:

1. Ejecutar sorteo ofreciendo semilla desde backoffice.
2. Publicar resultado y exportar acta/CSV.

Resultado Esperado: Listas ordenadas por universo; semilla registrada; CSV y acta descargables.

CP-13: Gestión usuarios backoffice

Descripción: Admin crea un usuario RRHH/Tribunal y asigna permisos.

Pasos de Reproducción:

1. Admin crear nuevo usuario con email y rol.
2. Asignar permisos.
3. Nuevo usuario inicia sesión y valida acciones de rol.

Resultado Esperado: Usuario creado con permisos; acciones ejecutables; registro auditabile.

CP-14: Publicación de resultados

Descripción: Tribunal publica resultados y postulante consulta por identificador.

Pasos de Reproducción:

1. Tribunal marca resultados como publicados.
2. Postulante consulta por número identificador en portal.

Resultado Esperado: Resultados anonimizados visibles; identificador muestra estado y puntaje sin datos personales.

Ejecución del Plan

Este plan de pruebas está pendiente de ejecución formal. La ejecución se realizará como actividad posterior a la defensa del proyecto, documentando para cada caso:

- **Resultado Real:** Comportamiento observado del sistema

- **Status:** Pass (si coincide con esperado) o Fail (si hay desviación)
- **Defectos:** Descripción de problemas encontrados si el caso falla
- **Evidencia:** Screenshots o logs que documenten la ejecución

Los resultados de la ejecución permitirán identificar defectos funcionales y priorizar correcciones antes de cualquier presentación del sistema a usuarios reales de la DGC.

Conclusión

Este plan de pruebas proporciona una validación sistemática de las funcionalidades implementadas en el MVP del Portal DGC. La cobertura de 15 escenarios abarca los flujos principales de los tres roles del sistema (postulante, tribunal, RRHH) y valida tanto operaciones básicas como lógica compleja del dominio (cuotas, ordenamiento, notificaciones).

La ejecución de este plan complementa la validación técnica realizada mediante tests unitarios e integración (155 tests, 87 % de cobertura), proporcionando validación end-to-end desde la perspectiva del usuario final.

Apéndice N: ESRE

Introducción

Identificación

El proceso de inscripción a los llamados públicos de la DGC [1], se realiza actualmente de forma presencial o mediante el envío de formularios y documentos por correo electrónico.

Este procedimiento genera demoras, duplicación de tareas administrativas y dificultades para garantizar la trazabilidad, la transparencia y el cumplimiento de las normativas vigentes sobre protección de datos personales y acciones afirmativas [4].

Con el objetivo de modernizar y optimizar este proceso, se propone el desarrollo de un portal web de postulaciones bajo el dominio institucional .gub.uy [5]. Este sistema permitirá que los aspirantes se registren, completen su postulación, adjunten documentos en formato PDF y reciban confirmación automática, asegurando el anonimato durante las etapas del concurso y el cumplimiento de las cuotas establecidas por ley.

El portal también contará con un backoffice para el área de Recursos Humanos y el Tribunal Evaluador, que facilitará la gestión de postulaciones, la preselección aleatoria auditada, la carga de puntajes y la generación automática de listas de prelación. Todo esto bajo un marco normativo que contempla las leyes 18.331 [3] (Protección de Datos Personales), 19.122 [7] (Acciones Afirmativas Afrodescendientes), 18.651 [9] (Protección integral a Personas con Discapacidad), 19.684 [8] (Ley integral para personas Trans) y el Decreto 406/022 [27] (Accesibilidad Digital).

El presente trabajo consiste en especificar, diseñar y desarrollar un prototipo funcional del sistema propuesto, aplicando prácticas de Ingeniería de Software, metodologías ágiles y principios de arquitectura en capas. A través de esta herramienta se busca mejorar la eficiencia del

workflow de concursos, garantizar la transparencia en los procesos de selección y fortalecer la confianza institucional en la gestión de recursos humanos del organismo.

Propósito del ESRE

El propósito de este documento es describir de forma completa, precisa y verificable los requerimientos funcionales y no funcionales del sistema, con el fin de servir como base para su diseño, desarrollo, implementación y validación.

El ESRE [N] está destinado a:

- El equipo de desarrollo, para comprender los requerimientos del sistema.
- El área de Recursos Humanos de la DGC [1], como cliente y responsable de validación.
- El tribunal evaluador, como usuario del backoffice.
- Los postulantes, como usuarios finales del portal público.

Alcance del producto

El sistema “Portal DGC” permitirá digitalizar completamente el proceso de postulación a concursos, abarcando desde la publicación del llamado hasta la generación de listas finales.

Incluye:

- Registro y autenticación de postulantes.
- Carga y validación de formularios y documentos PDF.
- Generación de códigos únicos de postulación y confirmaciones por correo o WhatsApp.
- Preselección aleatoria auditable según universos definidos (afrodescendiente, trans, discapacidad, general).
- Gestión de etapas: pruebas, méritos, entrevistas y listas de prelación.
- Backoffice para RRHH y Tribunal con control de roles.
- Exportación de datos para auditorías y reportes.

Excluye:

- Corrección automática de pruebas o evaluaciones.
- Integraciones con sistemas de nómina o legajos de funcionarios.

Generalidades del ESRE

Este documento está organizado en tres partes principales:

1. **Introducción:** Contexto, propósito, alcance y referencias del sistema.
2. **Descripción General:** Perspectiva del producto, funciones principales, características de usuarios, restricciones y supuestos.
3. **Especificación de Requerimientos:** Detalle completo de RF, RNF y restricciones de diseño.

Descripción general

Perspectiva del Producto

El Portal DGC surge como respuesta a la necesidad de digitalizar y optimizar el proceso de inscripción y gestión de postulaciones a concursos públicos dentro del organismo. Actualmente, el procedimiento de postulación se realiza mediante formularios físicos o correos electrónicos, lo que genera sobrecarga administrativa, riesgo de errores, falta de trazabilidad y dificultades en el cumplimiento de las normativas vigentes sobre protección de datos personales y acciones afirmativas [4].

El sistema propuesto constituye una plataforma web institucional bajo el dominio .gub.uy [5], compuesta por dos módulos principales:

- **Portal del Aspirante:** interfaz pública para el registro, postulación, carga de documentos y consulta de estado.

- **Backoffice Institucional:** interfaz restringida para Recursos Humanos y el Tribunal Evaluador, donde se gestionan los llamados, sorteos, evaluaciones y generación de listas finales.

Objetivos del sistema:

- Digitalizar el proceso de inscripción y selección de aspirantes.
- Reducir tiempos administrativos y eliminar el uso de formularios físicos.
- Garantizar la transparencia y trazabilidad en todas las etapas del concurso.
- Cumplir con las leyes de protección de datos personales (Ley N°18.331 [3]), accesibilidad digital (Decreto 406/022 [27]) y cupos laborales (Leyes N°19.122 [7], N°18.651 [9] y N°19.684 [8]).
- Facilitar la participación equitativa de todos los ciudadanos en los llamados públicos.
- Mejorar la experiencia de uso para postulantes y funcionarios.
- Proveer información confiable y auditable para la toma de decisiones.

Funciones del Producto

El sistema ofrecerá las siguientes funcionalidades principales, agrupadas por tipo de usuario:

Para postulantes (Portal público):

- Registro y autenticación de usuario: creación de cuenta mediante correo electrónico y contraseña.
- Gestión de perfil personal: carga y edición de datos personales, contacto y consentimiento legal.
- Visualización de llamados activos: listado de llamados públicos disponibles con sus requisitos.
- Postulación en línea: formulario digital de inscripción con validaciones automáticas.

- Carga de documentos PDF: subida de CV, certificados y constancias de cupos especiales.
- Confirmación de inscripción: generación de código único de postulación y envío de acuse por correo o WhatsApp.
- Consulta de estado: seguimiento de las etapas del proceso (inscrito, en evaluación, resultados).
- Visualización de resultados: acceso público a listas de prelación anonimizadas.

Para Recursos Humanos (Backoffice institucional):

- Gestión de llamados: creación, edición y publicación de llamados, cupos y requisitos.
- Validación documental: revisión de formularios y archivos subidos por los postulantes.
- Preselección automática: filtrado de postulantes que cumplen requisitos excluyentes.
- Ordenamiento aleatorio informático: sorteo auditable por universos (afro, trans, discapacidad, general).
- Control de etapas: configuración de pruebas, entrevistas y evaluación de méritos.
- Gestión de comunicaciones: envío de correos automáticos y notificaciones.
- Generación de reportes: exportación de datos, listas y estadísticas.

Para el Tribunal Evaluador:

- Acceso a postulaciones preseleccionadas.
- Validación documental: revisión de formularios y archivos subidos por los postulantes.
- Carga de resultados de pruebas de conocimiento, méritos y entrevistas.
- Cálculo automático de puntajes.
- Generación de listas de prelación finales.
- Emisión de informes y actas del proceso.

Para el Administrador del sistema:

- Gestión de usuarios y roles (RR. HH., Tribunal, Auditor).
- Mantenimiento general del sistema y base de datos.
- Supervisión de logs y auditoría.
- Configuración de parámetros y respaldos automáticos.

Para auditores institucionales (acceso eventual):

- Consulta de logs y trazabilidad.
- Verificación de cumplimiento normativo (Ley 18.331 [3], Unidad Reguladora y de Control de Datos Personales (URCDP [117])).

Características de los usuarios

El sistema contempla distintos tipos de usuarios con responsabilidades, permisos y niveles de acceso claramente definidos:

Postulante

Ciudadano que desea inscribirse a llamados públicos. Accede al portal público para registrarse, completar formularios, adjuntar documentos y consultar resultados. No requiere conocimientos técnicos avanzados.

Funcionario de RR. HH.

Personal administrativo responsable de gestionar llamados, validar documentación, configurar etapas y generar reportes. Requiere conocimientos de procesos administrativos y uso básico de herramientas ofimáticas.

Miembro del Tribunal Evaluador

Evaluador responsable de calificar pruebas, méritos y entrevistas. Accede al backoffice

para registrar puntajes y generar listas finales. Requiere conocimiento del proceso de evaluación y criterios establecidos en las bases.

Administrador del sistema

Personal técnico responsable de la gestión de usuarios, configuración del sistema, respaldos y mantenimiento. Requiere conocimientos técnicos avanzados en administración de sistemas.

Auditor Institucional / URCDP [117]

Fiscalizador con acceso eventual para verificar cumplimiento normativo, revisar logs y trazabilidad. Acceso de solo lectura con permisos específicos.

El sistema debe ofrecer interfaces diferenciadas y adaptadas al perfil de cada actor, priorizando la simplicidad de uso para el postulante y la trazabilidad para los usuarios internos. Todos los roles estarán autenticados mediante credenciales seguras y las acciones quedarán registradas en el módulo de auditoría para garantizar el no repudio.

Restricciones Generales

- El desarrollo se realizará bajo el stack tecnológico .NET 8 (backend) y Angular 19 [16](frontend).
- La base de datos se implementará con SQL Server [20].
- Todo dato personal deberá almacenarse cifrado y cumplir la normativa de protección de datos.
- El hosting deberá ser provisto por Antel Data Center u otra infraestructura estatal certificada.
- Las interfaces públicas deberán cumplir WCAG 2.1 nivel AA [28].

Supuestos y dependencias

- El dominio .gub.uy [5] es autorizado por AGESIC [10].
- Los formularios y criterios de selección se basan en las bases oficiales del llamado Fiscal III [11].

- Los envíos de notificaciones dependerán de servicios institucionales de correo y mensajería.
- El acceso al backoffice estará restringido a usuarios autenticados con credenciales otorgadas por DGC.

Especificación de requerimientos

Requerimientos Funcionales

RF-01: Autenticación de usuario

Descripción: Permite al usuario ingresar al sistema con usuario y contraseña.

Prioridad: Alta

Precondición: Usuario no autenticado.

Narrativa (User Story): Como usuario, quiero iniciar sesión para usar el sistema.

Entrada: Usuario, contraseña.

Salida esperada: Token de sesión válido / pantalla de inicio.

Criterios de Aceptación:

- La operación de login se completa en ≤ 3 s en condiciones normales.
- Luego de autenticarse, un recurso protegido responde 200 sin re-login.
- Ante credenciales incorrectas, se muestra un mensaje genérico único y el formulario queda habilitado para reintentar sin recargar.

RF-02: Registro de postulante

Descripción: Permite al aspirante crear una cuenta personal para participar en llamados.

Prioridad: Alta

Precondición: No existe cuenta registrada con el mismo correo o CI.

Narrativa (User Story): Como aspirante, quiero registrarme para poder postularse a llamados.

Entrada: Nombre, CI, correo electrónico, contraseña, teléfono.

Salida esperada: Alta exitosa y envío de correo de confirmación.

Criterios de Aceptación:

- Validación de unicidad de correo y CI.
- Confirmación vía link temporal con expiración de 24 h.
- Datos almacenados cifrados y marcados con fecha de registro.

RF-03: Visualización de llamados disponibles

Descripción: Muestra al usuario la lista de llamados activos y sus detalles.

Prioridad: Alta

Precondición: Usuario autenticado o acceso público permitido.

Narrativa (User Story): Como aspirante, quiero ver los llamados disponibles para decidir a cuál postularme.

Entrada: Ninguna / filtros (departamento, cargo).

Salida esperada: Listado de llamados vigentes con descripción, requisitos y fechas.

Criterios de Aceptación:

- La página carga en ≤ 2 s para 100 llamados activos.
- Se indican fechas de apertura y cierre, y enlace a detalle.

RF-04: Detalle de llamado y requisitos

Descripción: Permite acceder a la información completa del llamado seleccionado.

Prioridad: Alta

Precondición: Llamado activo.

Narrativa (User Story): Como aspirante, quiero leer los requisitos y bases antes de inscribirme.

Entrada: ID del llamado.

Salida esperada: Descripción completa (bases PDF, requisitos, cuotas, plazas).

Criterios de Aceptación:

- Incluye enlace de descarga a bases oficiales en PDF.
- Requisitos y cuotas (8 % afro, 1 % trans, 4 % discapacidad) visibles y correctos.

RF-05: Postulación en línea

Descripción: Permite completar el formulario digital de inscripción y enviarlo electrónicamente.

Prioridad: Alta

Precondición: Usuario autenticado; llamado vigente.

Narrativa (User Story): Como aspirante, quiero postularme digitalmente sin enviar correos.

Entrada: Datos personales, formación, experiencia, declaraciones juradas.

Salida esperada: Registro de postulación + código de confirmación único.

Criterios de Aceptación:

- Validaciones obligatorias según bases (edad ≥ 18 , voto, CI vigente).
- Genera código alfanumérico único.
- El formulario se guarda en borrador hasta el envío final.

RF-06: Carga de documentos PDF

Descripción: Permite subir documentos respaldatorios (CV, certificados, constancias).

Prioridad: Alta

Precondición: Postulación en curso.

Narrativa (User Story): Como aspirante, quiero adjuntar mis documentos requeridos en PDF.

Entrada: Archivos PDF (≤ 10 MB cada uno).

Salida esperada: Archivos almacenados y asociados a la postulación.

Criterios de Aceptación:

- Solo formatos .pdf aceptados.
- Verificación de tamaño y virus.
- Previsualización y opción de reemplazo antes del envío final.

RF-07: Confirmación de postulación

Descripción: Genera constancia con número identificador y la envía al correo del aspirante.

Prioridad: Alta

Precondición: Formulario completado y enviado.

Narrativa (User Story): Como aspirante, quiero recibir una constancia que valide mi inscripción.

Entrada: ID de postulación.

Salida esperada: Correo de confirmación con número identificador.

Criterios de Aceptación:

- Envío automático dentro de 1 min.
- Número correlativo único (garantiza anonimato).
- Mensaje visible también en el panel del usuario.

RF-08: Gestión de cupos especiales

Descripción: Permite declarar pertenencia a colectivos (afrodescendiente, trans, discapacidad) y validar constancias.

Prioridad: Alta

Precondición: Llamado con cupos definidos.

Narrativa (User Story): Como aspirante, quiero poder registrar mi situación en cupos especiales.

Entrada: Declaración y documentación soporte.

Salida esperada: Postulación clasificada en el universo correspondiente.

Criterios de Aceptación:

- Campo opcional con consentimiento explícito.
- Verificación documental manual o automática.
- Clasificación en 8 %, 1 % o 4 % según corresponda.

RF-09: Preselección automática

Descripción: Filtra postulaciones que cumplen los requisitos excluyentes.

Prioridad: Media

Precondición: Llamado cerrado.

Narrativa (User Story): Como tribunal, quiero que el sistema identifique a los postulantes elegibles.

Entrada: Conjunto de postulaciones.

Salida esperada: Lista de elegibles / no elegibles.

Criterios de Aceptación:

- Evaluación de criterios excluyentes declarados en bases.
- Genera log con motivo de exclusión.
- Resultado exportable (CSV).

RF-10: Ordenamiento aleatorio informático

Descripción: Realiza sorteo digital auditado por universos (afro, trans, discapacidad, Cañelones, Montevideo).

Prioridad: Alta

Precondición: Lista de elegibles cargada.

Narrativa (User Story): Como tribunal, quiero generar el orden aleatorio para definir quienes avanzan.

Entrada: Conjunto de postulantes elegibles.

Salida esperada: Listas ordenadas por universo.

Criterios de Aceptación:

- Uso de generador de números pseudoaleatorios criptográficos (CSPRNG) reproducible (semilla y hash registrados).
- Publicación de resultado dentro del día hábil siguiente.
- Resultado exportable (CSV).
- Exportación con acta digital firmada.

RF-11: Gestión de pruebas de conocimiento

Descripción: Administra convocatorias, asistencia y resultados de la prueba teórica y práctica.

Prioridad: Alta

Precondición: Ordenamiento aleatorio finalizado.

Narrativa (User Story): Como tribunal, quiero cargar y calificar las pruebas de los postulantes.

Entrada: Resultados individuales.

Salida esperada: Puntajes registrados y calculados.

Criterios de Aceptación:

- Puntaje 0-60 validado; mínimo 30 para aprobar.
- Recalcula totales automáticamente.
- Bitácora de fecha y responsable de carga.

RF-12: Carga de méritos y antecedentes

Descripción: Permite valorar formación y experiencia según tabla establecida.

Prioridad: Alta

Precondición: Postulante aprobó requisitos excluyentes.

Narrativa (User Story): Como tribunal, quiero asignar puntajes de méritos basados en documentación.

Entrada: Datos de cursos y experiencia.

Salida esperada: Puntaje 0-25.

Criterios de Aceptación:

- Se aplica tabla de ponderaciones oficial.
- Genera informe de valoración por postulante.

RF-13: Registro de entrevistas personales

Descripción: Registra puntuajes de entrevistas y observaciones del tribunal.

Prioridad: Media

Precondición: Etapa de entrevista abierta.

Narrativa (User Story): Como tribunal, quiero ingresar la calificación de la entrevista.

Entrada: ID postulante, puntaje (0-15).

Salida esperada: Puntaje almacenado y sumado al total.

Criterios de Aceptación:

- Solo miembros del tribunal pueden editar.
- Guarda nombre del evaluador y hora.

RF-14: Generación de listas de prelación

Descripción: Crea listas ordenadas por puntaje total y por universos especiales.

Prioridad: Alta

Precondición: Todas las etapas calificadas.

Narrativa (User Story): Como RR. HH., quiero obtener la lista final de prelación.

Entrada: Puntajes acumulados.

Salida esperada: Listas descendentes por departamento.

Criterios de Aceptación:

- En caso de empate, prioriza mayor puntaje en la prueba.
- Si persiste, aplica sorteo automatizado.
- Marca vigencia = 18 meses.

RF-15: Publicación de resultados

Descripción: Publica resultados de cada etapa y listas finales en el portal.

Prioridad: Alta

Precondición: Validación de tribunal.

Narrativa (User Story): Como aspirante, quiero poder consultar mi estado y resultados.

Entrada: Número identificador de postulación.

Salida esperada: Estado actual y puntajes públicos anonimizados.

Criterios de Aceptación:

- Solo muestra número identificador (sin datos personales).
- Publicación visible ≤ 24 h después de aprobación del tribunal.

RF-16: Comunicación y notificaciones

Descripción: Envía avisos automáticos por correo y WhatsApp (si está autorizado).

Prioridad: Media

Precondición: Evento relevante (confirmación, convocatoria, resultado).

Narrativa (User Story): Como aspirante, quiero recibir notificaciones de avances en el proceso.

Entrada: Tipo de evento y contacto.

Salida esperada: Mensaje entregado o log de intento fallido.

Criterios de Aceptación:

- Consentimiento previo requerido para WhatsApp.
- Registro de envío con fecha/hora.
- Reintentos automáticos en 15 min.

RF-17: Gestión de designaciones

Descripción: Registra aceptación o desistimiento del puesto ofrecido.

Prioridad: Media

Precondición: Postulante seleccionado.

Narrativa (User Story): Como seleccionado, quiero aceptar o rechazar la designación.

Entrada: Opción de respuesta y observaciones.

Salida esperada: Estado actualizado y confirmación al correo.

Criterios de Aceptación:

- Plazo de respuesta \leq 5 días hábiles.
- Si vence el plazo, se marca como desistido y pasa al siguiente.

RF-18: Gestión de usuarios del backoffice

Descripción: Permite crear, modificar y eliminar cuentas de RR. HH. y tribunal.

Prioridad: Alta

Precondición: Usuario administrador autenticado.

Narrativa (User Story): Como administrador, quiero gestionar los usuarios internos del sistema.

Entrada: Datos de cuenta y roles (RR. HH., tribunal, admin).

Salida esperada: Usuario habilitado con permisos específicos.

Criterios de Aceptación:

- Control de roles granulares.
- Auditoría de altas/bajas.

RF-19: Reportes y exportaciones

Descripción: Genera reportes estadísticos y exporta datos a CSV/PDF.

Prioridad: Media

Precondición: Llamado con datos procesados.

Narrativa (User Story): Como RR. HH., quiero obtener reportes para auditoría y seguimiento.

Entrada: Filtros (departamento, etapa, cupo).

Salida esperada: Archivo descargable o vista tabular.

Criterios de Aceptación:

- Exportación disponible en ≤ 5 s para 10,000 registros.
- Columnas validadas y totales correctos.

RF-20: Auditoría y trazabilidad

Descripción: Registra toda acción relevante en el sistema (altas, sorteos, calificaciones, publicaciones).

Prioridad: Alta

Precondición: Sistema en operación.

Narrativa (User Story): Como auditor, quiero poder revisar el historial completo de acciones.

Entrada: Evento del sistema.

Salida esperada: Registro inmutable con fecha, usuario y descripción.

Criterios de Aceptación:

- Inmutabilidad garantizada (no editable).
- Filtro por usuario, fecha, tipo de acción.
- Exportación firmada digitalmente.

RF-21: Gestión de llamados (RR. HH.)

Descripción: Permite crear, editar, publicar y cerrar llamados públicos con todos sus parámetros.

Prioridad: Alta

Precondición: Usuario RR. HH. autenticado.

Narrativa (User Story): Como RR. HH., quiero crear y configurar un nuevo llamado público.

Entrada: Título, descripción, bases PDF, departamentos, plazas, cupos, fechas, requisitos.

Salida esperada: Llamado creado y publicado.

Criterios de Aceptación:

- Validación de fechas coherentes (apertura <cierre).
- Carga de archivo PDF con bases oficiales.
- Configuración de cupos por ley (8 % afro, 1 % trans, 4 % discapacidad).
- Estado inicial: borrador (no visible para postulantes hasta publicación).

Requerimientos no Funcionales

Rendimiento

RNF-RT-01: Tiempo de respuesta

Atributo: Rendimiento

Descripción: Las operaciones críticas (login, envío de formulario de postulación y generación de listas) deberán completarse en <5 segundos bajo carga normal.

Método de validación: Pruebas de uso mediante DevTools de Google Chrome y simulación de usuarios concurrentes con herramientas como JMeter.

Métrica esperada: Tiempo total de respuesta <5 s en 95 % de las solicitudes. Tiempo máximo <10 s en picos de carga (200 usuarios concurrentes).

RNF-RT-02: Respuesta de consultas

Atributo: Rendimiento

Descripción: Las búsquedas y filtros en el backoffice (por departamento, cupo o etapa) deberán completarse en menos de 3 segundos.

Método de validación: Pruebas funcionales midiendo tiempos de ejecución en consola del navegador y consultas a la base de datos con índice aplicado.

Métrica esperada: 95 % de las consultas devuelven resultados en <3 s.

Usabilidad

RNF-UB-01: Flujo de postulación

Atributo: Usabilidad

Descripción: La interfaz deberá permitir completar la postulación en un máximo de 6 pasos (inicio de sesión, elección de llamado, formulario, carga de documentos, confirmación), informando el estado del sistema al usuario en todo momento.

Método de validación: Pruebas con usuarios reales (muestra ≥ 10) observando flujos principales y mensajes de retroalimentación.

Métrica esperada: Flujo completo en ≤ 6 pasos principales. 100 % de los pasos muestran confirmación visual o textual.

RNF-UB-02: Accesibilidad web

Atributo: Usabilidad

Descripción: El portal deberá cumplir con las pautas de accesibilidad WCAG 2.1 nivel AA [28], conforme al Decreto 406/022 [27] de AGESIC [10].

Método de validación: Validación automática con axe DevTools y revisión manual de contraste y navegación por teclado.

Métrica esperada: 0 errores críticos de accesibilidad. Contraste mínimo 4.5:1 en texto. Navegación completa sin ratón.

Portabilidad

RNF-PT-01: Compatibilidad de navegadores

Atributo: Portabilidad

Descripción: El sistema deberá ser compatible con las dos últimas versiones estables de Chrome, Edge y Firefox.

Método de validación: Pruebas funcionales cruzadas en navegadores con BrowserStack y validación visual de componentes.

Métrica esperada: 100 % de las funcionalidades principales operativas en los navegadores indicados.

Escalabilidad

RNF-SC-01: Carga concurrente

Atributo: Escalabilidad

Descripción: El sistema deberá soportar simultáneamente al menos 500 usuarios concurrentes sin degradación significativa.

Método de validación: Pruebas de carga y estrés con JMeter o k6, midiendo consumo de CPU, RAM y tiempos promedio.

Métrica esperada: CPU <80 % de uso promedio. Respuesta promedio <5 s bajo carga de 500 usuarios.

Seguridad

RNF-SG-01: Cifrado de comunicaciones

Atributo: Seguridad

Descripción: Toda comunicación y almacenamiento de datos sensibles deberá realizarse mediante HTTPS/TLS 1.3 [118] [119], con control de acceso basado en roles y cifrado AES-256 [120] en reposo.

Método de validación: Pruebas de seguridad y revisión de cabeceras HTTP [21] (OWASP ZAP).

Métrica esperada: 100 % de los endpoints críticos protegidos. 0 vulnerabilidades críticas o altas.

RNF-SG-02: Control de acceso por roles

Atributo: Seguridad

Descripción: Implementar gestión de roles y permisos para usuarios internos (RR, HH., Tribunal, Administrador) que restrinja acciones según privilegios.

Método de validación: Pruebas funcionales simulando distintos roles y verificando acceso o denegación.

Métrica esperada: 100 % de endpoints internos protegidos por rol. 0 accesos no autorizados detectados.

RNF-SG-03: Protección de datos personales

Atributo: Seguridad

Descripción: Los datos personales deberán cumplir con la Ley 18.331 [3] y guías de la URCDP [117] (minimización, finalidad, confidencialidad).

Método de validación: Revisión legal y checklist de privacidad por diseño.

Métrica esperada: 100 % de los campos con base legal identificada. 0 alertas de incumplimiento en auditoría.

Fiabilidad

RNF-FI-01: Disponibilidad del sistema

Atributo: Fiabilidad / Disponibilidad

Descripción: El sistema deberá tener una disponibilidad mínima del 99,5 % mensual, asegurando continuidad del servicio.

Método de validación: Monitoreo con UptimeRobot o Prometheus y logs de servidor.

Métrica esperada: $\leq 3,6$ h de inactividad mensual.

RNF-FI-02: Respaldos automáticos

Atributo: Fiabilidad / Recuperabilidad

Descripción: Los respaldos completos de base de datos deberán ejecutarse diariamente y permitir restauración en menos de 1 hora.

Método de validación: Pruebas de backup/restore programadas y revisión de logs de restauración.

Métrica esperada: Backups automáticos cada 24 h. Restauración exitosa <60 min.

Mantenibilidad

RNF-MT-01: Calidad del código

Atributo: Mantenibilidad

Descripción: El código deberá seguir principios SOLID [114] y arquitectura en tres capas (WebAPI, BusinessLogic, DataAccess) con cobertura de test $\geq 85\%$.

Método de validación: Revisión de código estática (SonarQube) y reportes de cobertura de pruebas unitarias.

Métrica esperada: 0 errores críticos de estilo. $\geq 85\%$ de líneas cubiertas por tests.

RNF-MT-02: Documentación técnica

Atributo: Documentación

Descripción: Cada módulo deberá contar con documentación técnica y manual de usuario actualizado.

Método de validación: Revisión de entregables y verificación de consistencia con versión desplegada.

Métrica esperada: 100 % de funcionalidades documentadas. 0 desvíos detectados en revisión.

Compatibilidad

RNF-IN-01: API REST [18] estándar

Atributo: Interoperabilidad

Descripción: El sistema deberá permitir exportar y consumir datos mediante API REST JSON [18] con formato abierto para integraciones con Uruguay Concursa u otros sistemas.

Método de validación: Pruebas de consumo con cliente HTTP [21] y validación de esquema JSON [41].

Métrica esperada: 100 % de endpoints responden con JSON [41] válido conforme OpenAPI.

Legalidad

RNF-LG-01: Dominio gubernamental

Atributo: Legal / Normativo

Descripción: El sistema deberá operar bajo dominio gub.uy [5], cumpliendo lineamientos de AGESIC [10] para gobierno digital y accesibilidad.

Método de validación: Verificación del dominio institucional y revisión del cumplimiento de lineamientos técnicos AGESIC [10].

Métrica esperada: Dominio institucional verificado. Cumplimiento total de checklist AGESIC [10].

RNF-LG-02: Transparencia de sorteos

Atributo: Transparencia

Descripción: Los sorteos y procesos aleatorios deberán registrar semilla y hash para auditoría pública.

Método de validación: Prueba de regeneración de ordenamiento a partir de semilla y verificación del hash publicado.

Métrica esperada: 100 % de sorteos reproducibles. Hash coincidente en auditoría.

Restricciones de Diseño

El sistema Portal DGC está sujeto a las siguientes restricciones de diseño que deben ser consideradas durante todo el ciclo de desarrollo:

Tecnológicas:

- Backend: .NET 8 + ASP.NET Core Web API [23]
- Frontend: Angular 19 [16] con TypeScript
- Base de Datos: SQL Server [20] (compatible con versiones 2019 o superior)
- ORM: Entity Framework Core 8 [24] con patrón Repository + Unit of Work [38]

- Autenticación: JSON Web Token (JWT) [41] con refresh tokens
- Testing: xUnit [69] + Moq para backend.

Arquitectónicas:

- Arquitectura en 3 capas: Presentación (Angular [16]), Lógica de Negocio (.NET Services), Acceso a Datos (Repositories)
- API RESTful [18] siguiendo convenciones HTTP estándar [21]
- Separación clara de responsabilidades entre capas
- Principios SOLID [114] en capa de negocio
- Inyección de dependencias en todos los servicios [39]

Infraestructura:

- Hosting: Antel Data Center u otra infraestructura estatal certificada
- Dominio: Subdominio de gub.uy [5]
- Certificado SSL: Emitido por autoridad certificadora reconocida por AGESIC [10]
- Segmentación de red: VLANs separadas para DMZ, aplicación, datos y administración
- Firewall perimetral con control de tráfico entre VLANs

Normativas:

- Ley 18.331 [3]: Todos los datos personales deben cumplir principios de minimización, finalidad y confidencialidad
- Decreto 406/022 [27]: Cumplimiento de WCAG 2.1 nivel AA [28] en todas las interfaces públicas
- Leyes 19.122 [7], 18.651 [9] y 19.684 [8]: Aplicación automática de cuotas de acción afirmativa

- Lineamientos AGESIC [10]: Seguimiento de guías técnicas para desarrollo de sistemas gubernamentales

Seguridad:

- Cifrado en tránsito: Transport Layer Security (TLS [119]) 1.3 obligatorio para todas las comunicaciones
- Cifrado en reposo: Estándar de Cifrado Avanzado (AES-256 [120]) para datos sensibles en base de datos
- Control de acceso: Role-Based Access Control (RBAC [115]) con roles predefinidos
- Auditoría: Registro inmutable de todas las acciones críticas con timestamp y usuario
- Validación: Doble validación (cliente y servidor) para todos los datos de entrada

Almacenamiento:

- Archivos PDF: Almacenamiento en Azure Blob Storage o similar con acceso controlado
- Base de Datos: Backups diarios automáticos con retención de 30 días
- Logs: Retención de logs de auditoría por mínimo 5 años

Interfaz y Experiencia:

- Diseño responsivo: Compatible con dispositivos móviles, tablets y desktop
- Navegadores: Soporte para Chrome, Edge y Firefox (últimas 2 versiones)
- Accesibilidad: Navegación completa por teclado, compatible con lectores de pantalla
- Idioma: Español (Uruguay)

Integraciones:

- Correo electrónico: Integración con servidor protocolo simple de transferencia de correo (SMTP [121]) institucional
- WhatsApp: Integración con API de mensajería (solo con consentimiento del usuario)
- Exportaciones: Formato CSV y PDF para reportes y listas

Estas restricciones son de cumplimiento obligatorio y no pueden ser modificadas sin autorización explícita de la Dirección General de Casinos y validación de AGESIC [10] cuando corresponda. Cualquier desviación debe ser documentada y justificada técnicamente.

Apéndice N: Casos de Uso

MÓDULO GESTIÓN DE USUARIOS

UC-01 Autenticación

Actor: Postulante / RR. HH. / Tribunal / Administrador

Curso normal

Actor	Sistema
1 - Accede a “Iniciar sesión”	2 - Despliega formulario de login.
3 - Ingresa usuario y contraseña y presiona “Ingresar”.	4 - Valida formato; encripta/hashea y consulta identidad/rol. 5 - Genera token, inicia sesión y registra log de acceso. 6 - Redirige al panel según rol (Postulante/RR. HH./Tribunal/Admin).

Curso Alternativo

- A1) Credenciales inválidas → Sistema muestra mensaje genérico “Usuario o contraseña incorrectos”, no indica si el usuario existe; permite reintentar.
- A2) Usuario bloqueado por intentos fallidos → Muestra “Su cuenta ha sido bloqueada temporalmente. Intente nuevamente en 15 minutos o solicite restablecimiento de contraseña”.
- A3) Sesión expirada → Redirige a login mostrando “Su sesión ha expirado, por favor ingrese nuevamente”.

Restricciones/Observaciones: Política de contraseña (mínimo 8 caracteres, mayúsculas, minúsculas, números); time-out de sesión (30 min inactividad); bloqueo por 3 intentos fallidos; auditoría de accesos; HTTPS obligatorio.

Requerimientos asociados: RF-01, RF-18, RNF-SG-01, RNF-SG-02, RNF-RT-01

UC-02 Registro de Postulante

Actor: Postulante

Curso normal

Actor	Sistema
1 - Accede a “Crear cuenta”.	2 - Despliega formulario de registro.
3 - Ingresa datos personales: CI, nombre completo, correo, contraseña, teléfono.	4 - Valida formato de campos (CI, email válido, teléfono). 5 - Verifica unicidad de email y CI.
6 - Acepta términos y condiciones y política de privacidad.	7 - Registra usuario con estado “Pendiente verificación”. 8 - Genera enlace de verificación con token único (expiración 24 h). 9 - Envía correo de confirmación.
10 - Accede al enlace recibido por correo.	11 - Valida token y activa cuenta. 12 - Muestra mensaje “Cuenta activada exitosamente” y redirige a login.

Curso Alternativo

- A1) CI o email ya registrados → Muestra “Ya existe una cuenta con estos datos. Si olvidó su contraseña, use ‘Recuperar contraseña’”.
- A2) Formato de email inválido → Indica “Ingrese un correo electrónico válido”.

- A3)** Contraseña no cumple política → Detalla requisitos: “La contraseña debe tener al menos 8 caracteres, incluir mayúsculas, minúsculas y números”.
- A4)** Enlace de verificación expirado → Muestra “El enlace ha expirado. Solicite un nuevo enlace de verificación”.
- A5)** No acepta términos → No permite continuar hasta aceptar.

Restricciones/Observaciones: Enlace de verificación con expiración de 24 h; política de contraseña obligatoria; datos almacenados con cifrado AES-256 [120]; cumplimiento Ley 18.331 [3] (finalidad, consentimiento).

Requerimientos asociados: RF-02; RNF-SG-01, RNF-SG-03, RNF-UB-01

UC-18 Gestión de Usuarios y Roles

Actor: Administrador

Curso normal

Actor	Sistema
1 - Selecciona “Administrar usuarios”.	2 - Muestra lista de usuarios existentes con filtros (rol, estado, búsqueda).
3 - Selecciona “Aregar nuevo” o edita usuario existente.	4 - Despliega formulario de registro interno (nombre, correo, rol, estado).
5 - Ingresa/modifica nombre, correo, selecciona rol (RRHH/Tribunal/Auditor/Admin) y estado (Activo/Inactivo).	6 - Valida datos obligatorios y formato. 7 - Verifica unicidad de correo.

8 - Presiona “Guardar”.	9 - Crea/actualiza usuario y genera contraseña temporal si es nuevo. 10 - Envía correo con credenciales temporales (nuevo usuario). 11 - Registra operación en auditoría (quién, qué, cuándo). 12 - Actualiza lista y muestra confirmación.
-------------------------	--

Curso Alternativo

- A1)** Campo obligatorio vacío → Alerta “Complete todos los campos requeridos”.
- A2)** Correo duplicado → Muestra “Ya existe un usuario con este correo electrónico”.
- A3)** Intento de eliminar propio usuario Admin → Muestra “No puede eliminar su propia cuenta de administrador”.
- A4)** Cambio de rol requiere confirmación → Sigue la confirmación “¿Está seguro de cambiar el rol de este usuario?”.

Restricciones/Observaciones: Solo usuarios con rol Administrador pueden acceder; principio de mínimo privilegio; registro de auditoría inmutable; contraseñas temporales con obligación de cambio en primer login.

Requerimientos asociados: RF-13, RF-20; RNF-SG-02, RNF-MA-01

MÓDULO POSTULANTE

UC-03 Visualizar Llamados

Actor: Postulante

Curso normal

Actor	Sistema
1 - Accede a “Llamados disponibles”.	2 - Lista llamados activos ordenados por fecha de cierre más próxima. 3 - Muestra filtros: departamento, cargo, fecha.
4 - Aplica filtros opcionales (departamento, cargo).	5 - Actualiza listado según filtros.
6 - Selecciona un llamado para ver detalle.	7 - Carga detalles del llamado.

Curso Alternativo

- A1) No hay llamados activos → Muestra “No hay llamados abiertos en este momento. Vuelva a consultar próximamente”.
- A2) Filtros no devuelven resultados → Muestra “No se encontraron llamados con los criterios seleccionados”.

Restricciones/Observaciones: Acceso público (sin autenticación); carga de página en ≤ 2 s para 100 llamados; fechas de apertura/cierre claramente visibles.

Requerimientos asociados: RF-03; RNF-RT-01, RNF-UB-01

UC-04 Detalle de Llamado

Actor: Postulante

Curso normal

Actor	Sistema
1 - Selecciona un llamado desde el listado (UC-03).	2 - Despliega información completa del llamado: <ul style="list-style-type: none">• Cargo y departamento• Cantidad de plazas• Requisitos excluyentes• Requisitos valorables (méritos)• Cuotas (8 % afro, 1 % trans, 4 % discapacidad)• Cronograma de etapas• Enlace a bases oficiales (PDF) 3 - Habilita botón “Postularme” si está autenticado y llamado vigente.

Curso Alternativo

- A1) Usuario no autenticado → Botón “Postularme” muestra “Debe iniciar sesión para postularse”.
- A2) Llamado cerrado → Muestra “Este llamado ya cerró su período de inscripción” y deshabilita postulación.
- A3) Bases PDF no disponibles → Muestra mensaje y contacto de RR. HH..

Restricciones/Observaciones: Información clara y completa; descarga de bases PDF disponible; cuotas legales visibles; acceso público.

Requerimientos asociados: RF-04; RNF-UB-01, RNF-AC-01

UC-05 Postulación a Llamado

Actor: Postulante

Curso normal

Actor	Sistema
1 - Ingresa al módulo “Llamados disponibles”.	2 - Lista los llamados abiertos con filtros por área y fecha.
3 - Selecciona un llamado.	4 - Muestra los detalles y requisitos del llamado.
5 - Presiona “Postularme”.	6 - Verifica autenticación y que no existe postulación previa. 7 - Despliega formulario workflow multipaso: <ul style="list-style-type: none">• Paso 1: Datos personales• Paso 2: Requisitos excluyentes• Paso 3: Méritos y antecedentes• Paso 4: Autodefinición Ley• Paso 5: Apoyos necesarios• Paso 6: Confirmación
8 - Completa cada paso del formulario, guardando borradores automáticamente.	9 - Valida campos obligatorios en cada paso. 10 - Permite navegación entre pasos. 11 - Guarda borrador automáticamente cada 30 segundos.

12 - Adjunta documentos requeridos (PDFs).	13 - Valida formato (solo PDF), tamaño (máx 10 MB) y antivirus/malware. 14 - Almacena archivos con hash único.
15 - Revisa resumen de postulación en paso final.	16 - Muestra resumen completo para verificación final.
17 - Presiona “Enviar postulación”.	18 - Valida completitud de requisitos excluyentes. 19 - Cambia estado a “Enviada”. 20 - Genera código único de postulación. 21 - Registra fecha/hora de envío. 22 - Dispara UC-06 (Confirmación).

Curso Alternativo

- A1)** Postulante ya inscrito en este llamado → Muestra “Ya se encuentra postulado a este llamado” y no permite duplicar.
- A2)** Campo obligatorio vacío → Alerta “Complete todos los campos requeridos” y no permite avanzar.
- A3)** Error de carga de archivo → Mantiene borrador y permite reintento sin perder datos.
- A4)** Archivo excede tamaño máximo → Muestra “El archivo excede el tamaño máximo permitido (10 MB)”.
- A5)** Archivo con virus/malware → Rechaza archivo y muestra “Archivo rechazado por razones de seguridad”.
- A6)** Llamado cerrado mientras completa → Muestra “El período de inscripción ha finalizado” y no permite enviar.

- A7)** Sesión expirada → Guarda borrador automático y redirige a login; al volver carga borrador guardado.

Restricciones/Observaciones: Cumplimiento Ley 18.331 [3] (consentimiento, finalidad); WCAG AA [28] en formularios; anonimato público por código; un postulante solo puede postularse una vez por llamado; borrador editable hasta cierre del llamado; validación dual (cliente + servidor).

Requerimientos asociados: RF-03, RF-04, RF-05, RF-06, RF-07; RNF-UB-01, RNF-RT-01, RNF-SG-03, RNF-AC-01

UC-06 Confirmación de Postulación

Actor: Sistema / Postulante

Curso normal

Actor	Sistema
-------	---------

	<p>1 - Recibe evento “Postulación enviada” (disparado por UC-05).</p> <p>2 - Genera código único alfanumérico (8 caracteres).</p> <p>3 - Crea constancia digital (PDF) con código, fecha/hora, llamado, estado.</p> <p>4 - Envía notificación por email con constancia adjunta.</p> <p>5 - Si autorizado: envía notificación WhatsApp con código.</p> <p>6 - Registra estado de entrega de notificaciones.</p> <p>7 - Muestra constancia en “Mis postulaciones”.</p>
8 - Accede a “Mis postulaciones” en su panel.	<p>9 - Muestra listado con código, llamado, fecha y estado.</p> <p>10 - Permite descargar constancia PDF.</p>

Curso Alternativo

- A1)** Error en envío de email → Reintenta 3 veces; registra fallo en log; muestra constancia en panel igualmente.
- A2)** Usuario no autorizó WhatsApp → Solo envía email.
- A3)** Email rebota → Registra error; permite al usuario actualizar correo desde su perfil.

Restricciones/Observaciones: Código es referencia pública (anónimo); mensajería WhatsApp sujeta a consentimiento explícito; reintentos automáticos ante fallos de entrega.

Requerimientos asociados: RF-07, RF-16, RF-20; RNF-RT-01, RNF-SG-03, RNF-FI-01

UC-07 Consultar Estado de Postulación

Actor: Postulante

Curso normal

Actor	Sistema
1 - Ingresa a “Mis postulaciones” en su panel.	2 - Lista todas las postulaciones con: código, llamado, fecha, estado, próxima acción.
3 - Selecciona una postulación para ver detalle.	4 - Muestra timeline de estados, fechas de etapas, próximos pasos, documentación y constancia.

Curso Alternativo

A1) Usuario sin postulaciones → Muestra “No tiene postulaciones activas”.

A2) Postulación en estado “Rechazada” → Muestra motivo de exclusión si aplicable.

Restricciones/Observaciones: Solo información del propio usuario; estados sincronizados en tiempo real; notificaciones proactivas al cambiar estado.

Requerimientos asociados: RF-07; RNF-UB-01, RNF-SG-03, RNF-FI-01

UC-17 Aceptar/Desistir Designación

Actor: Postulante seleccionado

Curso normal

Actor	Sistema
1 - Recibe notificación de designación.	2 - Envío notificación informando designación y plazo (5 días hábiles).
3 - Ingresa al portal en “Mis postulaciones”.	4 - Muestra notificación destacada. 5 - Presenta opciones: Aceptar/Desistir y contador de tiempo.
6 - Presiona “Aceptar” o “Desistir”.	7 - Sigue proceso.
8 - Confirma su decisión.	9 - Registra respuesta con fecha/hora. 10 - Cambia estado según decisión. 11 - Notifica a RR. HH.. 12 - Si desiste: activa siguiente en lista. 13 - Envía confirmación.

Curso Alternativo

A1) No responde en plazo → Sistema marca como “No respondió” y avanza al siguiente.

A2) Intenta cambiar respuesta → No permite modificación.

Restricciones/Observaciones: Plazo 5 días hábiles; respuesta irreversible; registro de auditoría.

Requerimientos asociados: RF-17, RF-16; RNF-FI-01, RNF-MA-01

MÓDULO RR. HH.

UC-08 Crear/Configurar Llamado

Actor: RR. HH.

Curso normal

Actor	Sistema
1 - Accede a “Gestión de llamados”.	2 - Muestra dashboard (activos, cerrados, borradores).
3 - Selecciona “Crear nuevo llamado”.	4 - Despliega formulario con secciones.
5 - Completa información básica.	6 - Valida campos obligatorios.
7 - Define departamentos y plazas.	8 - Valida plazas > 0. 9 - Calcula cuotas automáticamente (8 %, 1 %, 4 %).
10 - Define cronograma.	11 - Valida coherencia de fechas.
12 - Agrega requisitos.	13 - Registra cada requisito con tipo y ponderación.
14 - Adjunta bases (PDF).	15 - Valida formato y almacena con versionado.
16 - Revisa y presiona “Guardar” o “Publicar”.	17 - Crea llamado (Borrador/Publicado). 18 - Registra auditoría. 19 - Si publicado: notifica suscriptores.

Curso Alternativo

A1) Campo vacío → Alerta completar antes de publicar.

A2) Fechas incoherentes → Valida orden cronológico.

A3) Sin bases → Permite borrador, no publicación.

A4) Duplicar llamado → Ofrece reutilizar configuración.

Restricciones/Observaciones: Validación obligatoria, versionado; auditoría.

Requerimientos asociados: RF-08; RNF-MA-01, RNF-FI-01, RNF-UB-01

UC-09 Preselección por Requisitos

Actor: RR. HH.

Curso normal

Actor	Sistema
1 - Selecciona llamado cerrado.	2 - Verifica cierre y postulaciones.
3 - Confirma inicio.	4 - Evalúa requisitos excluyentes. 5 - Clasifica elegibles/no elegibles. 6 - Registra motivos de exclusión. 7 - Genera reporte completo. 8 - Actualiza estados.
9 - Revisa reporte.	10 - Presenta opciones de descarga y confirmación.
11 - Confirma preselección.	12 - Marca finalizado. 13 - Habilita sorteo. 14 - Notifica elegibles.

Curso Alternativo

A1) Llamado abierto → No permite ejecutar.

A2) Sin postulaciones → Indica no hay datos.

A3) Todos excluidos → Genera alerta para revisión.

Restricciones/Observaciones: Solo ejecutable con llamado cerrado; trazabilidad completa; exportable.

Requerimientos asociados: RF-09; RNF-FI-01, RNF-MA-01

UC-15 Publicar Resultados

Actor: RR. HH.

Curso normal

Actor	Sistema
1 - Accede a “Gestión de resultados”.	2 - Lista documentos pendientes.
3 - Selecciona documento.	4 - Muestra vista previa. 5 - Valida anonimización automática. 6 - Muestra resultado validación.
7 - Si OK, presiona “Publicar”.	8 - Publica en portal público. 9 - Registra auditoría. 10 - Notifica postulantes. 11 - Actualiza estados.

Curso Alternativo

A1) Datos sensibles detectados → Bloquea publicación.

A2) Documento no generado → Indica completar etapa previa.

A3) Error notificaciones → Publica y reintenta.

Restricciones/Observaciones: Solo códigos (anonimato); validación automática; cumplimiento Ley 18.331 [3].

Requerimientos asociados: RF-15; RNF-SG-03, RNF-FI-01, RNF-MA-01

MÓDULO TRIBUNAL

UC-10 Ordenamiento Aleatorio por Universos

Actor: RR. HH. / Tribunal

Curso normal

Actor	Sistema
1 - Accede a “Sorteo aleatorio”.	2 - Verifica preselección completada. 3 - Lista elegibles por universos.
4 - Selecciona universo.	5 - Muestra cantidad y plazas.
6 - Presiona “Ejecutar sorteo”.	7 - Genera semilla CSPRNG [122]. 8 - Registra semilla y timestamp. 9 - Ejecuta Fisher-Yates shuffle. 10 - Asigna posiciones. 11 - Calcula hash SHA-256. 12 - Genera acta digital firmada. 13 - Firma digitalmente.
14 - Revisa acta.	15 - Presenta opciones descarga.
16 - Confirma sorteo.	17 - Marca finalizado. 18 - Actualiza estados. 19 - Habilita publicación.

Curso Alternativo

- A1) Preselección incompleta → No permite ejecutar.
- A2) Sin elegibles → Indica universo vacío.
- A3) Re-ejecución → Requiere justificación y aprobación.

Restricciones/Observaciones: CSPRNG [122]; semilla y hash registrados; acta firmada; reproducibilidad; trazabilidad.

Requerimientos asociados: RF-10; RNF-SG-01, RNF-FI-01, RNF-MA-01, RNF-CM-01

UC-11 Gestionar Prueba de Conocimiento

Actor: Tribunal

Curso normal

Actor	Sistema
1 - Accede a “Pruebas”.	2 - Muestra postulantes ordenados.
3 - Configura convocatoria.	4 - Valida datos. 5 - Genera lista convocados. 6 - Envía notificaciones.
7 - Registra asistencia.	8 - Actualiza estado Presente/Ausente. 9 - Marca ausentes eliminados.
10 - Carga puntajes.	11 - Valida rango 0-60. 12 - Verifica umbral ≥ 30 . 13 - Clasifica Aprobado/Reprobado. 14 - Actualiza totales. 15 - Registra bitácora.
16 - Confirma cierre.	17 - Genera reporte estadístico. 18 - Notifica resultados. 19 - Habilita méritos.

Curso Alternativo

- A1) Puntaje fuera de rango → Rechaza valor.
- A2) Modificación posterior → Requiere justificación.

A3) Todos reprueban → Genera alerta.

Restricciones/Observaciones: Rango 0-60; umbral 30; bitácora; recalcula automático; notificaciones.

Requerimientos asociados: RF-11; RNF-FI-01, RNF-MA-01, RNF-SG-02

UC-12 Valorar Méritos y Antecedentes

Actor: Tribunal

Curso normal

Actor	Sistema
1 - Accede a “Méritos”.	2 - Lista postulantes aprobados.
3 - Selecciona postulante.	4 - Muestra información y documentos. 5 - Presenta tabla valoración.
6 - Revisa documentación.	7 - Permite visualizar PDFs.
8 - Asigna puntajes según tabla.	9 - Valida rango 0-25. 10 - Calcula total méritos.
11 - Agrega observaciones.	12 - Almacena observaciones.
13 - Guarda valoración.	14 - Actualiza total (Prueba + Méritos). 15 - Registra bitácora. 16 - Genera informe individual.
17 - Repite para cada postulante.	18 - Actualiza ranking.

Curso Alternativo

A1) Puntaje fuera de rango → Rechaza.

A2) Falta documentación → Permite puntaje 0 con justificación.

A3) Modificación → Requiere justificación y auditoría.

Restricciones/Observaciones: Requiere documentos; tabla paramétrica; bitácora; informe individual.

Requerimientos asociados: RF-12; RNF-FI-01, RNF-MA-01, RNF-SG-02

UC-13 Registrar Entrevista

Actor: Tribunal

Curso normal

Actor	Sistema
1 - Accede a “Entrevistas”.	2 - Lista convocados.
3 - Selecciona postulante.	4 - Abre ficha con datos y puntajes previos.
5 - Realiza entrevista.	
6 - Registra puntaje 0-15.	7 - Valida rango.
8 - Agrega comentarios.	9 - Almacena observaciones.
10 - Guarda evaluación.	11 - Actualiza total final. 12 - Registra bitácora. 13 - Bloquea si etapa cierra. 14 - Actualiza ranking.

Curso Alternativo

- A1) Fuera de rango → Rechaza.
- A2) Etapa cerrada → No permite edición.
- A3) Modificación justificada → Requiere aprobación.

Restricciones/Observaciones: Control permisos; sellos tiempo; rango 0-15; bloqueo automático.

Requerimientos asociados: RF-13; RNF-SG-02, RNF-FI-01, RNF-MA-01

UC-14 Generar Listas de Prelación

Actor: RR. HH. / Tribunal

Curso normal

Actor	Sistema
1 - Accede a “Generar listas finales”.	2 - Verifica etapas completas.
3 - Presiona “Generar”.	4 - Calcula totales (máx 100). 5 - Ordena descendente. 6 - Aplica desempate. 7 - Separa por departamento/universo. 8 - Aplica cuotas legales. 9 - Marca vigencia 18 meses. 10 - Genera documento.
11 - Revisa listas.	12 - Presenta opciones.
13 - Confirma.	14 - Versiona documento. 15 - Mantiene histórico. 16 - Actualiza estados. 17 - Habilita publicación.

Curso Alternativo

- A1)** Etapas incompletas → Indica faltante.
- A2)** Empate → Fuerza desempate por sorteo.
- A3)** Cuotas no alcanzables → Ajusta y documenta.
- A4)** Regeneración → Justificación y versiona.

Restricciones/Observaciones: Vigencia 18 meses; versionado; cuotas automáticas; desempeño; trazabilidad.

Requerimientos asociados: RF-14; RNF-FI-01, RNF-MA-01, RNF-CM-01

MÓDULO SISTEMA Y TRANSVERSALES

UC-16 Notificar Eventos

Actor: Sistema (configurado por RR. HH./Tribunal)

Curso normal

Actor	Sistema
	<ol style="list-style-type: none">1 - Detecta evento disparador.2 - Identifica destinatarios.3 - Carga plantilla mensaje.4 - Personaliza con datos.5 - Verifica canales autorizados.6 - Envía notificación.7 - Registra estado entrega.8 - Si fallo: reintenta hasta 3 veces.9 - Actualiza estado.10 - Permite reenvío manual si necesario.

Curso Alternativo

A1) Email rebota → Registra y solicita actualización.

A2) WhatsApp no autorizado → Solo email.

A3) Servicio caído → Encola y reintenta.

A4) Usuario desuscribe → Mantiene solo críticas.

Restricciones/Observaciones: WhatsApp con consentimiento; reintentos máx 3; registro estado; plantillas parametrizables.

Requerimientos asociados: RF-16; RNF-FI-01, RNF-SG-03

UC-19 Auditoría y Trazabilidad

Actor: Auditor / Administrador

Curso normal

Actor	Sistema
1 - Accede a “Auditoría”.	2 - Muestra interfaz con filtros.
3 - Aplica filtros.	4 - Consulta logs. 5 - Presenta resultados paginados.
6 - Selecciona registro.	7 - Muestra detalles completos.
8 - Exporta evidencias.	9 - Genera archivo CSV/PDF. 10 - Incluye hash verificación. 11 - Registra exportación.

Curso Alternativo

- A1) Sin permisos → Deniega acceso.
- A2) Sin resultados → Indica criterios sin datos.
- A3) Exportación grande → Genera en background.

Restricciones/Observaciones: Registros inmutables; solo lectura auditor; timestamps; hash integridad.

Requerimientos asociados: RF-19; RNF-SG-02, RNF-CM-01, RNF-MA-01

UC-20 Reportes y Exportaciones

Actor: RR. HH. / Tribunal

Curso normal

Actor	Sistema
1 - Accede a “Reportes”.	2 - Muestra catálogo predefinidos.
3 - Selecciona tipo.	4 - Presenta filtros específicos.
5 - Aplica filtros y genera.	6 - Procesa datos. 7 - Aplica anonimización. 8 - Genera visualizaciones. 9 - Implementa paginación si grande.
10 - Visualiza en pantalla.	11 - Muestra con gráficos interactivos.
12 - Descarga formato deseado.	13 - Genera archivo. 14 - Incluye metadatos.
15 - Guarda como plantilla (opcional).	16 - Almacena definición. 17 - Permite reutilizar.

Curso Alternativo

- A1) Sin datos → Indica no disponible.
- A2) Reporte muy grande → Genera en background.
- A3) Timeout → Sugiere reducir alcance.

Restricciones/Observaciones: Anonimización cuando corresponda; paginación; metadatos; plantillas; WCAG [28].

Requerimientos asociados: RF-20; RNF-RT-01, RNF-SG-03, RNF-UB-01

*

RESUMEN

*

Total de Casos de Uso: 20

Por Módulo:

- **Módulo Autenticación y Gestión:** 3 casos de uso (UC-01, UC-02, UC-18)
- **Módulo Postulante:** 6 casos de uso (UC-03, UC-04, UC-05, UC-06, UC-07, UC-17)
- **Módulo RR. HH.:** 3 casos de uso (UC-08, UC-09, UC-15)
- **Módulo Tribunal:** 5 casos de uso (UC-10, UC-11, UC-12, UC-13, UC-14)
- **Módulo Sistema:** 3 casos de uso (UC-16, UC-19, UC-20)