Universidad de La Habana
Facultad de Matemática y Computación

Sistema digital para la gestión de asignaturas electivas en la Universidad de La Habana

Autor:
Alex Samuel Bas Beovides

Tutor:
Lic. Carlos León González

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

15 de junio de 2025

https://github.com/AlexBeovides/Electia

Resumen

El presente trabajo aborda el diseño y desarrollo de un sistema digital para la gestión de asignaturas electivas universitarias, respondiendo a una necesidad de la Dirección de Formación de Pregrado de contar con una herramienta institucional que facilite la inscripción, aprobación y administración integral del proceso académico en la Universidad de La Habana. El sistema propuesto se centra en la creación de un entorno digital que permita a profesores, administradores y estudiantes gestionar el ciclo completo de las asignaturas electivas de manera eficiente, desde la propuesta docente hasta la matrícula estudiantil y el seguimiento académico. Además, como parte del documento, se discuten los desafíos y/o limitaciones de los sistemas similares y existentes de gestión académica, destacando la importancia de una solución adaptada al contexto universitario cubano que integre las particularidades organizativas del centro universitario.

El desarrollo del sistema se basa en una arquitectura de software robusta y escalable, utilizando tecnologías como ASP.NET Core, React.js, SQL Server y Docker para garantizar su sostenibilidad y rendimiento. En este trabajo se detallan los componentes principales del sistema, así como el diseño de la base de datos relacional, la interfaz de usuario diferenciada por roles y los servicios de gestión de propuestas académicas. También se presentan los resultados de las pruebas de validación experimental que demuestran la funcionalidad y eficiencia del sistema en la automatización de procesos administrativos y la generación de documentación académica oficial.

Abstract

This work addresses the design and development of a digital system for managing university elective courses, responding to the need for an institutional tool that facilitates enrollment, approval, and comprehensive administration of the academic process at the *Universidad de La Habana* (University of Havana). The proposed system focuses on the creation of a digital environment that enables professors, administrators, and students to efficiently manage the entire lifecycle of elective courses—from course proposal to student registration and academic monitoring. The challenges and limitations of existing academic management systems are discussed, highlighting the importance of a solution adapted to the Cuban university context and integrating the organizational particularities of the *Dirección de Formación de Pregrado* (Undergraduate Education Office).

The system's development is based on a robust and scalable software architecture, employing technologies such as ASP.NET Core, React.js, SQL Server, and Docker to ensure sustainability and performance. This work details the system's main components, including the design of the relational database, a role-based user interface, and services for managing academic proposals. Test results and experimental validation are presented, demonstrating the system's functionality and efficiency in automating administrative processes and generating official academic documentation. Finally, the implications of this work for the academic university community are discussed, and future lines of development are proposed to extend the system's functionality to other educational management processes.

Índice general

Introducción **1**

1. Marco teórico-conceptual **4**
1.1. Sistemas de asignación de recursos basados en reglas 4
1.2. Ingeniería de software aplicada a la gestión de recursos 5
1.2.1. Arquitectura de Software 6
1.2.2. Diseño Orientado al Dominio 6
1.2.3. Arquitectura por Capas 10
1.3. Estado del arte 13
1.3.1. Plataformas de gestión académica para la asignación de recursos 13
1.3.2. Relevancia de los sistemas de asignación de recursos en el contexto actual 15

2. Concepción y diseño de la solución **17**
2.1. Problemática 17
2.2. Determinación de los Requerimientos 19
2.2.1. Requerimientos Funcionales 19
2.2.2. Requerimientos No Funcionales 20
2.2.3. Requerimientos Informacionales 21
2.2.4. Requerimientos de Entorno 22
2.3. Sistema de Reglas para la Gestión de Inscripciones 22
2.4. Casos de uso 23
2.5. Concepción y diseño de la base de datos 27

3. Detalles de Implementación y Experimentación **30**
3.1. Tecnologías y herramientas utilizadas para el desarrollo del sistema . 30
3.1.1. Sistema de Gestión de Base de Datos: SQL Server 31
3.1.2. Framework de Desarrollo Backend: ASP.NET Core 31
3.1.3. Mapeo Objeto-Relacional: Entity Framework Core 32
3.1.4. Tecnología de Desarrollo Frontend: React.js 32
3.1.5. Despliegue de la aplicación: Docker 33

3.2. Arquitectura del sistema implementado 33
3.2.1. Desarrollo del Frontend 34
3.2.2. Principales vistas 35
3.2.3. Desarrollo del Backend 41
3.2.4. Despliegue con Docker 46
3.3. Pruebas y validación experimental 46
3.3.1. Entorno de Pruebas 47
3.3.2. Experimentos Realizados 48
3.3.3. Análisis de Resultados 51

Conclusiones **55**

Recomendaciones **56**

Bibliografía **58**

Índice de figuras

1.1. Diagrama de una arquitectura por capas. 12

2.1. Diagrama de casos de uso del Administrador. 24
2.2. Diagrama de casos de uso del Profesor. 25
2.3. Diagrama de casos de uso del Estudiante. 26
2.4. Modelo Entidad-Relacional Extendido de la base de datos. 28

3.1. Vista de la página de Gestión de Convocatoria. 38
3.2. Vista de la página de Evaluaciones del Curso. 40
3.3. Comportamiento del sistema durante la prueba de carga concurrente. 53

Introducción

La gestión eficiente de las asignaturas electivas en entornos universitarios constituye un desafío clave para garantizar la calidad del proceso formativo, la equidad al acceso de los cursos y el cumplimiento de los objetivos estratégicos de cada institución. Actualmente, la Dirección de Formación de Pregrado (DFP), perteneciente a la Universidad de La Habana, requiere sistemas que no solo faciliten la inscripción y aprobación de asignaturas por parte del profesorado, sino que además aseguren una adecuada organización, control y seguimiento de la matrícula estudiantil en correspondencia a criterios académicos definidos. Esta necesidad se hace más crítica en contextos donde la oferta electiva es diversa y debe ser gestionada por roles específicos, como administradores, profesores y estudiantes, cada uno con distintos niveles de acceso y funciones.

A nivel internacional, los Sistemas de Gestión del Aprendizaje (LMS, por sus siglas en inglés), como Moodle y Canvas, son ampliamente utilizados para administrar cursos y contenidos educativos en línea. Estas plataformas ofrecen funcionalidades robustas en cuanto a la creación de cursos, la gestión de usuarios y el seguimiento del desempeño académico. Sin embargo, presentan limitaciones cuando se trata de adaptar sus funcionalidades a necesidades particulares de gestión nacional. Por ejemplo, muchos de estos sistemas requieren pagar suscripciones para acceder a módulos avanzados o a servicios personalizados y, no siempre permiten configurar reglas específicas para la asignación automatizada de cursos, tales como restricciones por centro docente, perfil académico o intereses personales. Estas carencias impiden su implementación efectiva en escenarios que demandan alta personalización y autonomía.

Ante las limitaciones identificadas en los sistemas actuales de inscripción y administración de los cursos, el propósito central de esta tesis es el **diseño y desarrollo de una plataforma digital que permita gestionar de forma integral el proceso de inscripción, aprobación, asignación y matrícula de asignaturas electivas en la Universidad de la Habana**. Este sistema busca dar respuesta a diversas problemáticas institucionales que dificultan la organización eficiente de estas ofertas docentes, entre las que destacan:

1. La inexistencia de una herramienta digital propia que responda a las particula-

ridades organizativas y normativas de la Dirección de Formación de Pregrado.

2. La dispersión del proceso de inscripción y validación de asignaturas electivas, la cual depende actualmente de métodos manuales.

3. La dificultad para aplicar criterios diferenciados de acceso según carrera, año académico u otros parámetros establecidos por la institución.

4. La ausencia de mecanismos automáticos para el seguimiento de la matrícula, control de cupos, generación de actas y obtención de estadísticas útiles para la toma de decisiones.

Como solución a las problemáticas recién mencionadas, se propone el desarrollo de un sistema web para la inscripción y gestión de asignaturas electivas, accesible a todos los actores del proceso educativo universitario, garantizando la transparencia y eficiencia de cada etapa. Esta solución estará basada en una arquitectura modular y una base de datos relacional, permitiendo escalabilidad, control de acceso según los roles y adaptabilidad a futuros requerimientos institucionales. Para cumplir este propósito general, se plantean los siguientes objetivos específicos:

1. Diseñar una interfaz visual donde el profesorado proponga nuevas asignaturas electivas, junto a datos requeridos por la DFP para su evaluación y aprobación.

2. Implementar un módulo de revisión y validación de asignaturas para ciertos usuarios, permitiendo definir criterios y filtros personalizables según la edición del curso a impartir.

3. Desarrollar funcionalidades de inscripción estudiantil que respeten la capacidad máxima de cada curso y permitan aplicar reglas específicas por carrera, facultad y/o año académico.

4. Automatizar la asignación de estudiantes, generar listados, actas académicas y reportes, minimizando errores producto del trabajo manual.

5. Evaluar el desempeño del sistema mediante pruebas funcionales y de usabilidad, con el fin de asegurar su efectividad y aceptación por parte de los usuarios finales.

La presente investigación se organiza en tres capítulos, cada uno enfocado en una etapa clave del proceso de desarrollo del sistema propuesto. Ellos son:

- **Capítulo 1: Marco teórico-conceptual.** Se examinan los principios teóricos relacionados con la gestión académica de asignaturas electivas en entornos universitarios, así como la caracterización de plataformas similares, sus funcionalidades, beneficios y limitaciones, con especial atención a las necesidades del contexto cubano. Además, se abordan conceptos esenciales del campo de la Ingeniería de Software que resultan útiles para el desarrollo del trabajo.
- **Capítulo 2: Concepción y diseño de la solución.** Se expone la propuesta del sistema informático, especificando los requerimientos funcionales, no funcionales, informacionales y de entorno, los roles de usuario, los principales casos de uso, el diseño de la base de datos necesaria, y un flujo general de cómo se presenta el sistema propuesto.
- **Capítulo 3: Implementación y experimentación.** Se describen los detalles del proceso de la implementación del sistema, incluyendo las tecnologías utilizadas, la arquitectura del sistema y los resultados obtenidos a partir de pruebas de funcionamiento, eficiencia y experiencia de usuario.

Finalmente, se presentan las conclusiones del trabajo, en las que se sintetizan los principales resultados obtenidos y las recomendaciones para trabajos futuros. A continuación, se incluye la bibliografía consultada, que respalda los contenidos desarrollados y las decisiones de diseño adoptadas a lo largo de la investigación.

Capítulo 1

Marco teórico-conceptual

Este capítulo tiene como objetivo presentar un conjunto de bases teóricas que permitan la construcción del sistema presentado en los capítulos posteriores. Dentro de los conceptos claves, se encuentran los relacionados con los sistemas basados en reglas, arquitecturas de software y criterios de diseño. Asimismo, se examinan sistemas existentes para la gestión de recursos, específicamente en el área de la educación dentro de la Universidad de La Habana.

1.1. Sistemas de asignación de recursos basados en reglas

Los sistemas de asignación de recursos basados en reglas constituyen un enfoque ampliamente utilizado en la gestión de procesos que requieren la distribución automatizada de recursos limitados entre un conjunto de entidades, según criterios predefinidos. Este tipo de sistema se fundamenta en la definición de reglas lógicas que determinan la elegibilidad de las entidades o recursos y priorizan su asignación en función de parámetros específicos, como características demográficas, académicas o estratégicas (Giarratano y Riley 2005).

Además, los sistemas basados en reglas pueden incorporar mecanismos de priorización para resolver conflictos cuando la demanda supera la capacidad del recurso. Las reglas de priorización establecen un orden de preferencia, antes de asignar los recursos de manera aleatoria o por otros criterios. Este enfoque no solo automatiza el proceso, reduciendo la intervención manual, sino que también garantiza consistencia y trazabilidad en las decisiones de asignación.

Además de su aplicabilidad práctica, los sistemas de asignación de recursos basados en reglas presentan diversas alternativas técnicas para su implementación, dependiendo del paradigma de programación adoptado. En el ámbito de la programación

lógica, por ejemplo, un lenguaje como Prolog[1] permite construir una base de conocimientos donde las reglas se definen directamente en los ficheros fuente, combinando hechos con reglas de inferencia (Bratko 2012). Estas reglas son evaluadas por un motor de inferencia que aplica mecanismos de retorno (backtracking) y una estrategia de búsqueda en profundidad basada en unificación para explorar todas las soluciones posibles (Clocksin y Mellish 2003).

Por otro lado, en sistemas empresariales más tradicionales, es común el uso de motores de reglas de negocio que implementan estructuras condicionales (como IF-THEN-ELSE) y donde las reglas pueden almacenarse en bases de datos relacionales o herramientas especializadas como IBM Operational Decision Manager (ODM) o Oracle Business Rules (Boyer y Mili 2011). Asimismo, entornos como Drools[2] optan por definir las reglas en archivos de configuración utilizando formatos como XML, JSON o YAML, lo que proporciona una mayor flexibilidad para el despliegue y mantenimiento de las reglas, especialmente en sistemas que requieren adaptabilidad continua (Fowler 2013).

La gestión de recursos basada en reglas es un enfoque que, aunque sencillo, resulta efectivo en contextos donde los criterios de asignación son claros y no se requieren algoritmos de optimización complejos. Por ejemplo, en entornos de producción industrial las llamadas reglas de despacho (dispatching rules) constituyen métodos heurísticos simples capaces de generar planes de producción adecuados de forma rápida y con bajo costo computacional (Zeiträg y Figueira 2023).

1.2. Ingeniería de software aplicada a la gestión de recursos

La ingeniería de software constituye una disciplina fundamental que abarca el conjunto de principios, métodos y herramientas orientados al desarrollo sistemático de sistemas computacionales de alta calidad. Esta disciplina se centra en la aplicación de enfoques rigurosos y metodológicos para el análisis, diseño, implementación, verificación y mantenimiento de softwares, garantizando que los sistemas resultantes sean confiables, escalables y mantenibles a lo largo de su ciclo de vida útil.

Los sistemas de asignación de recursos basados en reglas representan un área de particular relevancia, donde la aplicación de principios sólidos de ingeniería permite desarrollar algoritmos sofisticados capaces de manejar restricciones múltiples, criterios de optimización y reglas de negocio específicas del dominio. La arquitectura de software, el diseño orientado al dominio y los patrones arquitectónicos constituyen

[1] https://www.swi-prolog.org/
[2] https://www.drools.org/

pilares fundamentales que permiten estructurar sistemas complejos de manera que sean comprensibles, modificables y extensibles.

1.2.1. Arquitectura de Software

La arquitectura de software constituye el conjunto de decisiones estructurales fundamentales que definen la organización, interconexión y comportamiento de los componentes de un sistema informático, estableciendo los principios rectores para su diseño e implementación (Bass et al. 2021). Desde la perspectiva de la ingeniería de software, estas decisiones modelan cómo interactúan los distintos módulos funcionales como la gestión de usuarios, el procesamiento de datos, la validación de reglas de negocio y la generación de informes para satisfacer requisitos técnicos y operativos.

La arquitectura de software cumple propósitos estratégicos dentro del desarrollo de sistemas: permite abordar restricciones técnicas específicas, optimiza atributos de calidad como el rendimiento en escenarios de alta carga, garantiza la seguridad en el manejo de información crítica y proporciona escalabilidad para afrontar cambios en el tamaño o complejidad del sistema. Asimismo, una arquitectura bien estructurada reduce la complejidad del mantenimiento, facilita la evolución del sistema y permite la incorporación de nuevas funcionalidades sin comprometer su integridad.

Desde una visión integral, la arquitectura de software no se limita a aspectos puramente técnicos; también refleja las prioridades y estructuras organizativas del equipo de desarrollo. En este sentido, sirve como un puente conceptual entre los procesos de negocio abstractos y su implementación tecnológica, asegurando que las decisiones arquitectónicas se alineen con los objetivos funcionales y los criterios de calidad definidos para el producto.

La gestión de la complejidad inherente a los sistemas con múltiples responsabilidades requiere la aplicación de principios arquitectónicos sólidos que consideren tanto aspectos técnicos como organizacionales. En este contexto, la Ley de Conway establece que "las organizaciones que diseñan sistemas están restringidas a producir diseños que son copias de las estructuras de comunicación de estas organizaciones" (Conway 1968). Esta correspondencia entre la estructura organizativa y el diseño arquitectónico adquiere relevancia en cualquier entorno de desarrollo donde las dinámicas internas influyen directamente en las soluciones técnicas producidas.

1.2.2. Diseño Orientado al Dominio

El Diseño Orientado al Dominio (Domain-Driven Design, DDD) constituye una aproximación metodológica que sitúa el conocimiento del dominio como eje central del proceso de desarrollo de software, concebida por Eric Evans a comienzos del siglo XXI (Evans 2004). Esta filosofía busca establecer una correspondencia directa

entre los sistemas informáticos y la realidad que modelan, priorizando la comprensión profunda de los procesos funcionales propios de cada entorno de aplicación.

En contextos caracterizados por una alta complejidad lógica y una variedad significativa de reglas de negocio, DDD permite abordar dicha complejidad mediante la segmentación del dominio en áreas funcionales claramente definidas. A diferencia de enfoques centrados exclusivamente en la estructura técnica, el diseño orientado al dominio promueve una integración estrecha entre el conocimiento específico del dominio y las decisiones arquitectónicas del software (Khononov 2021).

La implementación de esta perspectiva resulta especialmente útil cuando el modelado de sistemas exige una comprensión detallada de reglas, excepciones y procesos internos. DDD reconoce a los expertos del dominio como figuras clave en la construcción de modelos conceptuales robustos, capaces de reflejar fielmente los comportamientos y restricciones del entorno operativo.

La metodología propuesta en DDD se organiza en dos grandes áreas: el diseño estratégico y el diseño táctico. El diseño estratégico se enfoca en definir cuál es el propósito del software y qué necesidades del negocio busca resolver. Por su parte, el diseño táctico aborda la forma en que se construyen los componentes del sistema para representar con precisión las reglas y procesos del dominio. Ambos enfoques se complementan para asegurar que el desarrollo del software esté alineado con los objetivos de la organización, sea mantenible y pueda adaptarse con el tiempo.

Diseño estratégico

Dentro de la propuesta metodológica del Diseño Orientado al Dominio, el diseño estratégico representa la fase encargada de definir los límites conceptuales y operativos del sistema. Esta etapa busca identificar los subdominios esenciales del problema, organizándolos en contextos delimitados que reflejen la estructura funcional y semántica del dominio tratadoEvans 2004.

Estos subdominios pueden corresponder, por ejemplo, a procesos independientes con reglas de negocio diferenciadas, niveles de autorización distintos o flujos operativos separados. Establecer límites claros entre ellos permite reducir el acoplamiento, minimizar ambigüedades conceptuales y evitar conflictos semánticos durante el crecimiento o la integración de nuevas funcionalidades.

La identificación de los subdominios se realiza en estrecha colaboración con los expertos del dominio, quienes aportan tanto conocimiento tácito como explícito sobre las operaciones relevantes. Esta interacción constante entre desarrolladores y expertos permite la construcción de un lenguaje compartido, conocido como lenguaje ubicuo, el cual unifica la terminología empleada en el código, la documentación y la comunicación técnica (Vernon 2016). Este lenguaje es fundamental para preservar la coherencia semántica y facilitar la evolución controlada del sistema.

El resultado de este proceso es un modelo de dominio que actúa como representación conceptual y operativa del problema, orientando el diseño del software y sirviendo como base para validar su alineación con los requerimientos reales. Dicho modelo de dominio está diseñado para adaptarse a medida que evolucionan las necesidades funcionales o cambian las políticas operativas del entorno en cuestión.

El diseño estratégico, por tanto, no constituye una fase rígida del desarrollo, sino un proceso iterativo y adaptativo que permite alinear continuamente la arquitectura del sistema con las prioridades funcionales del dominio. Su correcta aplicación fomenta la flexibilidad, facilita el mantenimiento y asegura la sostenibilidad técnica del producto a largo plazo.

Diseño táctico

El diseño táctico constituye la materialización práctica de los conceptos identificados durante la fase estratégica, proporcionando los mecanismos concretos para transformar el conocimiento del dominio en estructuras de código ejecutables y mantenibles. Esta dimensión operativa de DDD se fundamenta en la aplicación sistemática de patrones específicos que permiten representar fielmente la complejidad inherente a dominios de negocio ricos y cambiantes (Vernon 2013).

La transición desde el modelo conceptual hacia la implementación técnica requiere el empleo de construcciones arquitectónicas que preserven la semántica del dominio y faciliten la evolución futura del sistema. Esta transformación debe considerar tanto las reglas de negocio específicas como los flujos de interacción que articulan las operaciones centrales del sistema.

Los elementos fundamentales del diseño táctico incluyen estructuras que encapsulan los conceptos nucleares del dominio, las cuales son:

- **Entidades**:
  Constituyen los elementos fundamentales que poseen identidad propia y mantienen continuidad a lo largo de su ciclo de vida. Son componentes que, aunque sus propiedades puedan cambiar, conservan su identidad y siguen siendo reconocibles dentro del sistema.

- **Objetos de valor**:
  Representan conceptos sin identidad individual, definidos exclusivamente por sus atributos. Su importancia radica en las características que encapsulan y no en una identidad única dentro del sistema.

- **Agregados**:
  Conforman unidades cohesivas que agrupan entidades y objetos de valor relacionados, estableciendo límites de consistencia y controlando el acceso mediante

una entidad raíz. Esta estructura garantiza que las reglas de negocio se apliquen de manera consistente en todo el conjunto.

- **Servicios de dominio**:
  Encapsulan operaciones del dominio que no pertenecen naturalmente a ninguna entidad u objeto de valor. Estos servicios permiten coordinar reglas o procesos complejos que implican múltiples elementos del modelo, sin comprometer la cohesión de los objetos existentes.

- **Repositorios**:
  Proporcionan una abstracción sobre los mecanismos de persistencia, permitiendo que la lógica del dominio opere de manera independiente de los detalles técnicos relacionados con el almacenamiento de datos. Además, facilitan el acceso a las entidades del dominio como si fueran colecciones en memoria.

- **Eventos de dominio**:
  Capturan acontecimientos significativos que ocurren dentro del modelo y que pueden provocar efectos secundarios o reacciones dentro del sistema. Estos eventos permiten una comunicación clara y desacoplada entre las distintas partes del sistema.

La implementación efectiva del diseño táctico demanda adherencia a principios fundamentales que aseguren la coherencia entre el modelo conceptual y su representación técnica (Evans 2004); cuyos principios son:

- **Separación de la lógica del dominio**:
  Es un requisito esencial para lograr flexibilidad y mantenibilidad. Se basa en aislar la lógica central del negocio de las consideraciones infraestructurales como las bases de datos, las interfaces de usuario o las integraciones externas, utilizando patrones arquitectónicos adecuados.

- **Coherencia del modelo**:
  El mantenimiento de la coherencia exige que todas las construcciones técnicas respeten las reglas y restricciones del dominio. Esto implica una validación continua para garantizar que entidades, objetos de valor y agregados conserven las invariantes definidas y representen fielmente los procesos clave.

- **Preservación del lenguaje ubicuo**:
  Es esencial mantener la correspondencia semántica entre el dominio de modelado y su implementación en el código fuente. Esta práctica asegura que los términos utilizados por los expertos del dominio se reflejen directamente en las estructuras del software, facilitando una comprensión compartida entre todos los miembros del equipo.

Con el fin de incorporar la interacción del usuario sin comprometer la estructura del código, es posible recurrir a diversas arquitecturas de software, como la Arquitectura por Capas o la Arquitectura por Componentes. Estas arquitecturas proporcionan soluciones estructuradas que responden tanto a las necesidades técnicas como operativas y permiten una organización y escalabilidad sostenibles del código. En este sentido, se abordará una arquitectura ampliamente utilizada que puede combinarse eficazmente con DDD: la Arquitectura por Capas.

1.2.3. Arquitectura por Capas

La Arquitectura por Capas representa uno de los paradigmas estructurales más consolidados en el desarrollo de los sistemas informáticos contemporáneos, caracterizándose por su enfoque organizacional basado en la separación funcional de las responsabilidades. Su amplia adopción en el ámbito de las aplicaciones empresariales se fundamenta en la simplicidad conceptual, la familiaridad que genera entre los equipos de desarrollo y los costos relativamente reducidos asociados a su implementación (Richards y Ford 2020).

Esta aproximación arquitectónica resulta particularmente intuitiva debido a su correspondencia natural con la estructura operativa de muchos equipos de desarrollo, en los que las responsabilidades suelen dividirse entre especialistas en presentación, lógica de negocio y gestión de datos. En consonancia con la Ley de Conway, esta organización técnica tiende a reflejarse directamente en la arquitectura del software, facilitando la distribución de tareas y la colaboración entre roles diferenciados (Richards y Ford 2020).

Sin embargo, esta alineación puede derivar en patrones arquitectónicos problemáticos sino se gestiona adecuadamente. Entre las principales dificultades identificadas se encuentran: el acoplamiento excesivo entre niveles funcionales y la dispersión inadecuada de la lógica de negocio, los cuales pueden comprometer la cohesión del sistema y dificultar su mantenimiento a largo plazo (Richards y Ford 2020).

La estructuración fundamental de esta arquitectura se basa en la organización de componentes en estratos horizontales lógicamente diferenciados, donde cada nivel asume responsabilidades específicas dentro del ecosistema aplicativo. Aunque la cantidad de capas puede variar según la complejidad del sistema, la configuración más extendida comprende cuatro niveles principales: Presentación, Lógica de Negocio, Persistencia y Almacenamiento de Datos, cuyas responsabilidades se describen a continuación:

- **Capa de Presentación:**
  Constituye el punto de contacto directo entre el sistema y sus usuarios finales, implementando los mecanismos de interacción necesarios para acceder a

sus funcionalidades. Esta capa debe proporcionar experiencias diferenciadas según el perfil del usuario, asegurando el acceso controlado a las funcionalidades relevantes. Aunque tradicionalmente esta arquitectura está asociada a las interfaces gráficas, hoy en día abarca múltiples modalidades, tanto síncronas como asíncronas (Khononov 2021). Sus manifestaciones contemporáneas incluyen interfaces web, aplicaciones móviles, interfaces de línea de comandos, interfaces de programación de aplicaciones (API) y mecanismos de suscripción a eventos mediante sistemas de mensajería.

- **Capa de Lógica de Negocio:**
  Representa el núcleo conceptual del sistema, encargándose de la implementación y encapsulación de todas las reglas, procesos y políticas que rigen el comportamiento del dominio. Aquí residen las decisiones algorítmicas que determinan la respuesta del sistema frente a condiciones específicas, validaciones, flujos de aprobación y automatización de procesos (Khononov 2021). Una correcta implementación garantiza que las operaciones se mantengan alineadas con los requisitos del dominio y que los datos permanezcan consistentes a lo largo del ciclo de vida del sistema.

- **Capa de Persistencia:**
  Gestiona la comunicación entre la lógica de negocio y los diversos mecanismos de almacenamiento. Esta responsabilidad no se limita al manejo tradicional de las bases de datos, sino que puede incluir la integración con servicios en la nube, sistemas de mensajería, almacenamiento de archivos o servicios externos (Khononov 2021). El propósito de esta capa es abstraer los detalles de acceso a los datos y proporcionar interfaces conceptuales estables para su consumo por parte de otras capas.

- **Capa de Almacenamiento de Datos:**
  Se encarga del acceso directo a los repositorios físicos de información. Su función principal consiste en implementar consultas y modificaciones sobre las fuentes de datos. A diferencia de la capa de persistencia, no expone interfaces abstractas, sino que opera directamente sobre la infraestructura de almacenamiento (Richards y Ford 2020).

Consideraciones Arquitectónicas

Dentro del análisis arquitectónico realizado, la organización por capas surge como una opción comúnmente adoptada debido a su enfoque técnico y estructurado. Esta forma de agrupación, basada en responsabilidades tecnológicas, no en casos de uso, puede facilitar ciertos aspectos de modularidad y separación de preocupaciones.

Sin embargo, también introduce desafíos cuando se desea mantener la cohesión funcional de una misma operación de negocio, ya que esta suele estar distribuida a lo largo de varias capas, lo que puede dificultar su evolución, especialmente en enfoques orientados al dominio (Richards y Ford 2020).

Algunos modelos alternativos, como el enfoque propuesto por Khononov 2021, proponen una estructura simplificada de tres capas, consolidando las responsabilidades de persistencia y almacenamiento en una única Capa de Acceso a Datos. Esta organización puede complementarse con una Capa de Servicios adicional que expone los casos de uso del sistema, coordinando las interacciones entre niveles subyacentes.

La comunicación entre las capas sigue un patrón descendente unidireccional, donde cada nivel interactúa únicamente con la capa inmediatamente inferior, como se muestra en la Figura 1.1. Esta disciplina permite mantener un bajo acoplamiento y una clara separación de responsabilidades, lo cual es esencial para la integridad arquitectónica y la mantenibilidad del sistema.

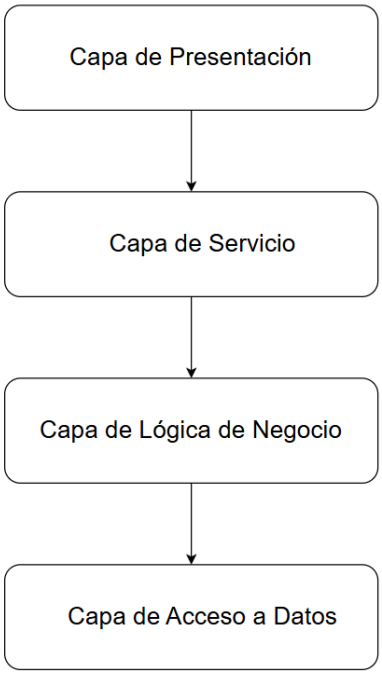

Figura 1.1: Diagrama de una arquitectura por capas. Cada rectángulo es una capa de la arquitectura y las flechas indican la comunicación entre ellas.

En contextos donde la complejidad del dominio es moderada o las reglas de negocio son simples —como sistemas de tipo CRUD[3] administrativos, aplicaciones internas

[3]Acrónimo que se refiere a las cuatro operaciones básicas que se pueden realizar en un sistema

de gestión o portales de contenido—, es común consolidar las capas de Negocio y Persistencia en una única estructura funcional. Esta simplificación resulta especialmente práctica cuando se emplean herramientas de mapeo objeto-relacional (ORM) como TypeORM, Hibernate o Entity Framework, que proporcionan una abstracción automática entre los objetos del lenguaje de programación y las estructuras de consulta de la base de datos.

1.3. Estado del arte

En esta sección se presenta una revisión de las soluciones tecnológicas existentes orientadas a la asignación de recursos. El análisis se enfoca en sistemas de gestión y asignación en entornos educativos, que han incorporado mecanismos basados en reglas para la distribución eficiente y controlada de recursos limitados, tales como asignaturas con cupos restringidos, espacios físicos y disponibilidad de los docentes.

La revisión permite identificar tanto las fortalezas como las limitaciones de dichas plataformas, especialmente en lo referido a su adaptabilidad a contextos institucionales específicos, como el universitario cubano. Asimismo, se discuten los enfoques arquitectónicos y funcionales adoptados, y se extraen elementos relevantes para el diseño de una solución tecnológica contextualizada que responda adecuadamente a las necesidades locales.

1.3.1. Plataformas de gestión académica para la asignación de recursos

A continuación, se analizan sistemas que implementan mecanismos de asignación de recursos basados en reglas de disponibilidad y criterios de elegibilidad, evaluando su aplicabilidad en diferentes escenarios institucionales.

Classe365

Classe365[4] es una plataforma integral de gestión educativa que se distingue por su enfoque en la automatización de procesos administrativos y la asignación inteligente de recursos académicos. Su arquitectura modular incluye un sistema robusto para la gestión de inscripciones que opera mediante reglas de elegibilidad configurables, permitiendo establecer criterios específicos para determinar qué estudiantes pueden acceder a determinados cursos o recursos en función de su programa académico, año de estudio, prerrequisitos cumplidos y disponibilidad de cupos.

de almacenamiento: Crear (Create), Leer (Read), Actualizar (Update), Eliminar (Delete).

[4]https://www.classe365.com/

La plataforma implementa un motor de reglas que evalúa automáticamente la disponibilidad de recursos en tiempo real, considerando factores como la capacidad máxima, las restricciones temporales y los criterios de prioridad. Su sistema de asignación permite configurar múltiples niveles de validación, desde verificación de prerrequisitos hasta aplicación de políticas institucionales específicas para la distribución equitativa de recursos limitados.

Entre sus fortalezas destacan la capacidad de manejo de listas de espera automatizadas, generación de reportes de ocupación de recursos y notificaciones automáticas a estudiantes sobre la disponibilidad. Sin embargo, su naturaleza comercial implica costos significativos de licenciamiento y dependencia del proveedor para las actualizaciones y el mantenimiento, lo que puede representar limitaciones presupuestarias. Además, no resulta adaptable a los entornos cubanos, específicamente en el manejo de los cursos electivos para los estudiantes de las discímiles carreras universitarias que hoy existen, lo cual limita su aplicabilidad en contextos con estructuras curriculares más diversas y dinámicas.

OpenEduCat

OpenEduCat[5], construido sobre el framework Odoo[6], representa una solución de código abierto especializada en la gestión integral de instituciones educativas con particular énfasis en la asignación automatizada de recursos académicos. Su arquitectura basada en módulos permite implementar sistemas de reglas complejas para la distribución de asignaturas, considerando múltiples variables como la capacidad de aulas, la disponibilidad de profesores, los horarios institucionales y los criterios de elegibilidad estudiantil.

El sistema opera mediante un motor de flujos de trabajo que evalúa la disponibilidad de recursos en función de las reglas predefinidas, permitiendo establecer criterios diferenciados según la facultad, la carrera o el nivel académico. Su módulo de gestión de cursos incluye funcionalidades avanzadas para el control automatizado de cupos, la validación de prerrequisitos y la asignación basada en las prioridades institucionales.

Las ventajas de OpenEduCat incluyen su flexibilidad para la personalización debido a su naturaleza de código abierto, la integración nativa con otros módulos de gestión administrativa y su capacidad para implementar reglas de negocio específicas sin restricciones de licenciamiento. No obstante, su implementación requiere conocimientos técnicos especializados en el framework, y la ausencia de soporte comercial directo puede representar desafíos para aquellas instituciones sin equipos técnicos

[5]https://openeducat.org/

[6]Odoo es una suite de aplicaciones empresariales de código abierto que incluye comercio electrónico, facturación, contabilidad, manufactura, almacén, gestión de proyectos e inventario, todo integrado en una plataforma modular.

especializados. Al igual que Classe365, presenta limitaciones en su adaptación a las particularidades del contexto universitario cubano, especialmente en lo referente a la gestión flexible de cursos electivos en estructuras curriculares altamente diversas.

Moodle

Moodle[7] es una plataforma de gestión del aprendizaje (LMS, por sus siglas en inglés) ampliamente utilizada a nivel global por instituciones educativas de todos los niveles. Si bien su diseño original se centró en la gestión de contenidos y actividades didácticas en entornos virtuales, su estructura modular y la extensa comunidad de desarrolladores han permitido la incorporación de funcionalidades orientadas a la gestión y asignación de recursos académicos. A través de extensiones especializadas y personalizaciones, Moodle puede gestionar la inscripción automática en cursos con cupos limitados, validar requisitos previos y establecer prioridades de acceso según criterios definidos por la institución.

Entre sus funcionalidades destacadas se encuentra la posibilidad de integrar reglas de asignación mediante módulos que permiten condicionar el acceso a cursos y recursos en función de variables académicas, temporales o administrativas. Asimismo, existen extensiones para la administración de listas de espera y notificaciones automatizadas, facilitando una experiencia más dinámica y controlada para los estudiantes. Su carácter de código abierto y alta escalabilidad lo convierten en una opción viable para su adaptación a contextos institucionales específicos, incluidos los entornos universitarios cubanos.

Sin embargo, su enfoque principal como LMS limita de forma nativa las capacidades de gestión administrativa compleja, como la planificación detallada de la disponibilidad de espacios físicos o la asignación optimizada de docentes. La implementación de funcionalidades avanzadas requiere el desarrollo de extensiones o integraciones externas, lo cual demanda recursos técnicos adicionales y puede implicar una carga de mantenimiento significativa.

1.3.2. Relevancia de los sistemas de asignación de recursos en el contexto actual

En el contexto universitario cubano, donde la gestión de asignaturas electivas debe optimizar recursos limitados mientras cumple con normativas institucionales específicas, los sistemas de asignación basados en reglas representan una aproximación más apropiada que las plataformas tradicionales de gestión del aprendizaje. A diferencia de estas plataformas que se enfocan en la entrega de contenido, los sistemas basados en

[7] https://moodle.org/

reglas priorizan la automatización de decisiones de asignación mediante la evaluación sistemática de criterios de elegibilidad y disponibilidad de recursos.

Las soluciones analizadas en la Sección 1.3.1 demuestran que es posible implementar mecanismos sofisticados de asignación que consideren múltiples variables simultáneamente, desde restricciones de capacidad física hasta políticas académicas complejas. Sin embargo, tanto Classe365 como OpenEduCat presentan limitaciones para el contexto específico cubano: la primera por sus costos asociados y dependencia comercial; la segunda por los requerimientos técnicos especializados para su implementación y mantenimiento. En el caso de Moodle, si bien ofrece una base flexible y extensible gracias a su naturaleza de código abierto y amplia adopción nacional, su enfoque centrado en la gestión del aprendizaje y la necesidad de personalización avanzada para cubrir procesos administrativos complejos como la asignación optimizada de recursos, limitan su aplicabilidad directa sin un esfuerzo adicional de desarrollo e integración.

Por estas razones, resulta indispensable considerar el desarrollo de una solución tecnológica propia, diseñada específicamente para responder a las necesidades del contexto educativo cubano. Un sistema de asignación de asignaturas electivas desarrollado localmente permitiría implementar reglas de negocio específicas para el control automatizado de disponibilidad y elegibilidad, ofreciendo una interfaz adaptada a los procesos institucionales existentes y capacidad para generar reportes especializados que apoyen la toma de decisiones académicas, sin las limitaciones económicas o técnicas de las soluciones existentes.

Capítulo 2

Concepción y diseño de la solución

El desarrollo de los sistemas digitales de gestión de asignaturas electivas en la educación superior requiere una comprensión profunda de las necesidades institucionales y una traducción precisa de dichas necesidades en especificaciones técnicas funcionales. La fase de análisis y diseño constituye el puente fundamental entre la identificación de las problemáticas organizacionales y la materialización de soluciones tecnológicas viables.

Este capítulo presenta el diseño teórico-conceptual de la plataforma, aborda el análisis detallado de los requerimientos y describe los roles y perfiles de usuario del sistema. Asimismo, se establecen los casos de uso principales y la arquitectura de datos necesaria para garantizar que la solución responda de manera integral a las necesidades identificadas en el contexto de la gestión de asignaturas electivas universitarias.

2.1. Problemática

Las instituciones de educación superior enfrentan desafíos significativos en la gestión de asignaturas electivas, particularmente cuando estas deben coordinarse entre múltiples facultades, carreras y niveles académicos. En el contexto de la Universidad de la Habana, este proceso tradicionalmente se ha caracterizado por una alta dependencia de métodos manuales y sistemas fragmentados que no logran integrarse de manera efectiva las diferentes etapas del ciclo de gestión académica. Esta situación genera ineficiencias operativas, limita la transparencia del proceso y dificulta la toma de decisiones basada en datos confiables.

Esta problemática se manifiesta de manera particular en la Dirección de Formación de Pregrado, donde convergen las necesidades de múltiples actores con diferentes niveles de responsabilidad y acceso a la información. Los profesores requieren mecanismos ágiles para proponer nuevas asignaturas electivas, especificando contenidos,

requisitos y modalidades de evaluación. Simultáneamente, los administradores académicos necesitan herramientas que faciliten la revisión, validación y aprobación de estas propuestas, aplicando criterios institucionales específicos. Por su parte, los estudiantes demandan procesos claros de inscripción, equitativos y que respeten las particularidades de su plan de estudios.

Los desafíos descritos se agravan al analizar detenidamente el flujo actual del proceso de gestión de asignaturas electivas. En su forma actual, la Dirección de Formación de Pregrado debe coordinar múltiples fases administrativas con una alta carga operativa y escaso apoyo tecnológico. Los profesores interesados en impartir asignaturas electivas deben presentar una propuesta detallada que incluya información general sobre la asignatura, su fundamentación, objetivos, temario, bibliografía, modalidad de impartición, sistema de evaluación y una justificación de la modalidad seleccionada. Además, los profesores deben registrar datos personales y profesionales tanto del docente principal como de sus colaboradores, incluyendo categoría docente, grado científico y medios de contacto.

A esta información se suma la necesidad de especificar elementos logísticos como la capacidad máxima de la matrícula, los requisitos mínimos para cursar la asignatura, el lugar exacto donde se impartirá (en caso de ser presencial o híbrida), así como los ejes y sectores estratégicos a los que tributa el curso. Todo este proceso requiere la recopilación y validación manual de una cantidad considerable de datos, muchas veces mediante documentos físicos o formatos no estandarizados, lo que introduce una alta probabilidad de errores, demoras y pérdida de información. Una vez recibidas las propuestas de cursos, la Dirección debe revisarlas y aprobarlas.

Los estudiantes, por su parte, acceden a un listado de asignaturas aprobadas para realizar su inscripción, donde deben proporcionar información detallada que incluye datos personales, académicos y de contacto, además de justificar su interés en matricular la asignatura.

Los actores involucrados —profesores, estudiantes y administradores— se enfrentan a limitaciones importantes: los profesores a la complejidad del proceso de solicitud; los estudiantes a la opacidad y lentitud del procedimiento de inscripción y; los administradores a la falta de herramientas que permitan un control eficiente, seguro y centralizado. Además, la elaboración de actas, el seguimiento de las calificaciones, la producción de reportes y estadísticas y la trazabilidad general del proceso resultan fragmentados y altamente dependientes del trabajo manual, dificultando cualquier análisis posterior de calidad o eficiencia.

La fragmentación del proceso recién mencionado y la falta de automatización impiden una planificación académica ágil, dificultan la transparencia en la toma de decisiones y retrasan los procesos claves para el desarrollo curricular y la satisfacción de las necesidades formativas de los estudiantes.

A continuación se detallan los requerimientos específicos que justifican el desarrollo

de la solución matemática-computacional propuesta.

2.2. Determinación de los Requerimientos

Los requerimientos del sistema se agrupan en cuatro categorías fundamentales: funcionales, no funcionales, informacionales y de entorno. Estos fueron identificados mediante un análisis detallado de la problemática institucional expuesta en la sección anterior y las necesidades específicas de la Dirección de Formación de Pregrado en el contexto de la gestión de las asignaturas electivas.

2.2.1. Requerimientos Funcionales

Las siguientes funcionalidades han sido definidas para permitir una gestión integral del proceso de inscripción, aprobación y matrícula de asignaturas electivas:

- **Definición de roles diferenciados:**
  Se contemplan tres perfiles de usuario:
  - **Personal de la Dirección de Formación de Pregrado:** Acceso completo al sistema, con funciones de supervisión, aprobación y configuración de parámetros institucionales. Aprueba los cursos propuestos por los profesores y es quien determina cuando un curso es puesto como disponible en un semestre.
  - **Profesor:** Propone asignaturas, gestiona información académica y administra calificaciones.
  - **Estudiante:** Consulta la oferta de asignaturas, realiza su inscripción y accede a su información académica.
- **Gestión del proceso de propuesta de asignaturas:**
  La plataforma debe permitir el registro detallado de propuestas, incluyendo datos generales, docentes responsables, contenidos académicos, modalidades, cupos, requisitos y vínculos institucionales estratégicos.
- **Módulo de revisión y aprobación administrativa:**
  La plataforma debe de proveer herramientas especializadas que permitan a los administradores evaluar las propuestas según los criterios institucionales y aprobar las asignaturas de acuerdo a dichos criterios.
- **Validación de identidad con sistemas institucionales existentes:**
  El sistema debe permitir la verificación de credenciales institucionales (correo

y contraseña universitarios) a través de mecanismos de autenticación con servicios de la universidad, garantizando que los usuarios registrados sean miembros válidos de la comunidad académica.

- **Definición de reglas de inscripción:**
  El sistema debe permitir la creación y configuración de las reglas asociadas a cada asignatura, siendo una regla de inscripción el conjunto de condiciones o criterios que determinan si un estudiante puede o no inscribirse en una determinada asignatura. Estas reglas pueden incluir el año académico del estudiante y la carrera que cursa.

- **Funcionalidades de inscripción estudiantil:**
  Los estudiantes deben acceder al catálogo de asignaturas aprobadas e inscribirse según la disponibilidad de cupos y las reglas configurables por el administrador.

- **Generación automatizada de documentación académica:**
  El sistema debe generar documentos oficiales, con estructura y formato requeridos para certificación institucional, tales como las actas académicas.

2.2.2. Requerimientos No Funcionales

Para garantizar un correcto funcionamiento, la solución debe cumplir con los siguientes requerimientos no funcionales:

- **Seguridad y control de acceso:**
  Autenticación y autorización según los roles y considerar la protección de datos personales y académicos. El control de acceso se implementará a través de un servicio proporcionado por la Universidad de La Habana para la autenticación de usuarios mediante su correo electrónico institucional, garantizando así la verificación de identidad y el acceso seguro al sistema.

- **Usabilidad y experiencia de usuario:**
  Interfaces intuitivas, accesibles y adecuadas para los usuarios con distintos niveles de competencia tecnológica. La interfaz debe respetar los colores oficiales de la Universidad de La Habana, manteniendo la identidad visual institucional.

- **Administración y mantenimiento:**
  Herramientas para configurar parámetros, gestionar contenidos y usuarios, y actualizar el sistema con mínima interrupción.

- **Accesibilidad multiplataforma:**
  Aplicación web compatible con diversos dispositivos y navegadores sin necesidad de instalación adicional.

2.2.3. Requerimientos Informacionales

El sistema debe presentar información clara, oportuna y contextualizada, organizada en formatos adecuados para apoyar los procesos de gestión, propuesta, matrícula y evaluación de asignaturas electivas. A continuación, se describen los tipos de información que se visualizarán para cada perfil de usuario:

a) Dirección de Formación de Pregrado (Administrador)

- Visualización de listados tabulares con filtros por estado de propuesta (pendiente, aprobada, rechazada), que muestran:
  - Títulos de asignaturas propuestas.
  - Estado actual de cada propuesta.
- Paneles de detalle para cada docente, con secciones que incluyen información laboral, académica y datos de contacto.
- Gráficos estadísticos y tablas dinámicas con distribución de matrícula por asignatura y por facultad.
- Reportes descargables con calificaciones en formato estructurado para análisis cualitativo.

b) Profesor

- Fichas informativas de sus propuestas de asignatura, presentadas en formato estructurado con campos como: contenidos temáticos, modalidad, bibliografía, criterios de evaluación y cupo máximo.
- Listado interactivo de estudiantes inscritos en sus cursos, con columnas para nombre, contacto, facultad, carrera y año académico.

c) Estudiante

- Catálogo navegable de asignaturas disponibles para matrícula, con visualizaciones por tarjeta o lista que incluyen:
  - Nombre del curso y del profesor responsable.
  - Modalidad (virtual, presencial, híbrida).
  - Unidad académica que ofrece la electiva.
  - Descripción general del curso.

  - Ubicación física o plataforma de impartición.

- Indicador visible del estado de su inscripción en cada asignatura (confirmada, pendiente, cerrada por cupo).

Estos elementos informacionales están diseñados para facilitar la toma de decisiones, la transparencia del proceso y la autonomía de cada usuario dentro del sistema.

2.2.4. Requerimientos de Entorno

Para una implementación exitosa y operación estable, se establecen los siguientes requerimientos del entorno:

- **Arquitectura multiplataforma:**
  Compatibilidad con distintos sistemas operativos y entornos de servidor.

- **Alojamiento en infraestructura nacional:**
  Uso de servidores nacionales para garantizar autonomía tecnológica y cumplimiento de normativas de protección de datos.

- **Escalabilidad y rendimiento:**
  Capacidad de manejar crecimiento en usuarios y datos sin afectar el rendimiento, incluso en periodos de alta demanda.

2.3. Sistema de Reglas para la Gestión de Inscripciones

El sistema implementa un mecanismo de reglas que permite controlar y optimizar el proceso de inscripción de estudiantes. Este sistema de reglas constituye el núcleo para garantizar una asignación equitativa y ordenada de las plazas disponibles, considerando los criterios académicos establecidos por la institución.

Una regla en el contexto del sistema se define como un conjunto de condiciones o criterios que determinan el comportamiento del proceso de inscripción. Estas reglas operan sobre los atributos fundamentales de los estudiantes, específicamente la carrera que cursan y el año académico en el que se encuentran, permitiendo establecer restricciones y prioridades de acceso a las asignaturas electivas.

Reglas de Filtrado

Las reglas de filtrado constituyen el primer nivel de control en el proceso de inscripción. Su función principal es determinar qué estudiantes tienen derecho a visualizar y

solicitar inscripción en una determinada convocatoria de curso electivo. Estas reglas establecen los criterios de elegibilidad básicos que debe cumplir un estudiante para acceder a la oferta académica.

Cuando se define una regla de filtrado para una convocatoria específica, el sistema evalúa automáticamente si cada estudiante cumple con los criterios establecidos. Solo aquellos estudiantes que satisfagan al menos una de las reglas de filtrado configuradas podrán visualizar el curso en el catálogo y proceder con su solicitud de inscripción. Este mecanismo garantiza que las asignaturas electivas se dirijan exclusivamente a la población estudiantil objetivo, respetando los prerrequisitos académicos y curriculares establecidos.

Reglas de Prioridad

Las reglas de prioridad operan en el segundo nivel del proceso de inscripción y entran en acción durante la fase de selección definitiva de estudiantes. Su propósito es establecer un orden jerárquico para la asignación de plazas cuando el número de solicitudes excede la capacidad de matrícula del curso.

Estas reglas se aplican secuencialmente según el orden de prioridad establecido por el administrador del sistema. Cada regla de prioridad selecciona estudiantes que cumplan con criterios específicos hasta agotar la capacidad disponible o hasta que no existan más estudiantes elegibles bajo esa regla. Este proceso continúa con las reglas subsiguientes hasta completar la matrícula o agotar todas las solicitudes válidas.

El sistema de reglas de prioridad permite implementar políticas institucionales específicas, como dar preferencia a estudiantes de determinadas carreras o años académicos, garantizando así una distribución estratégica y equitativa de las oportunidades educativas.

La combinación de ambos tipos de reglas proporciona un mecanismo robusto y flexible para la gestión automatizada de inscripciones, reduciendo la intervención manual y minimizando la posibilidad de errores en el proceso de asignación de plazas.

2.4. Casos de uso

La definición de los casos de uso constituye un enfoque metodológico fundamental para representar las interacciones entre el sistema y sus diferentes actores, estableciendo un marco conceptual que facilita la comprensión de las funcionalidades requeridas y los flujos de trabajo necesarios para su implementación. A través del análisis de los casos de uso se logra identificar de manera precisa las responsabilidades y capacidades de cada rol dentro del sistema, proporcionando una base sólida para el diseño de la arquitectura funcional.

En el contexto del sistema de gestión de asignaturas electivas, se han identificado tres actores principales que interactúan con la plataforma, cada uno con responsabilidades específicas y niveles de acceso diferenciados que reflejan su función dentro del proceso académico institucional.

El sistema contempla tres perfiles de usuario claramente diferenciados, que coinciden con los usuarios descritos en la Sección 2.2.1, a continuación se muestran las interacciones que puede hacer cada uno en el sistema:

Interacciones del Administrador

El administrador del sistema ejerce funciones de supervisión y control que garantizan el cumplimiento de los estándares académicos institucionales. Además de sus funciones administrativas, el administrador tiene la capacidad de desempeñar todas las acciones disponibles para el perfil de profesor. Sus interacciones principales incluyen la evaluación y aprobación de propuestas de cursos o asignaturas presentadas por los profesores, aplicando criterios específicos establecidos por la institución.

Estas interacciones se representan en la Figura 2.1, que muestra el diagrama de casos de uso correspondiente al perfil de administrador.

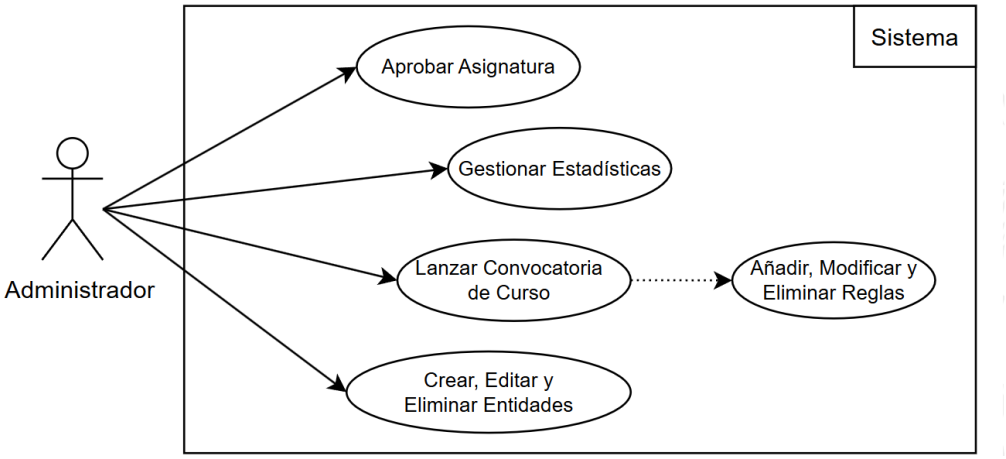

Figura 2.1: Diagrama de casos de uso del Administrador. El caso de uso "Crear, Editar y Eliminar Entidades" agrupa acciones sobre múltiples entidades del sistema, como asignaturas, facultades, centros, carreras y usuarios. Esta agrupación se realizó para mantener la claridad visual del diagrama.

La gestión de entidades constituye una función central del administrador, quien puede crear, editar y eliminar elementos fundamentales del sistema como facultades, centros, carreras y usuarios. Esta capacidad de configuración permite adaptar el sistema a las necesidades específicas de la institución y mantenerlo actualizado ante cambios organizacionales.

El establecimiento y la modificación de las reglas de inscripción, que fueron detalladas en la Sección 2.3, representa otra responsabilidad crítica del administrador, . Establecer correctamente estas reglas es fundamental para asegurar una planificación académica ordenada, coherente con los programas de estudio y equitativa para todos los estudiantes.

La generación de estadísticas y reportes proporciona al administrador herramientas analíticas para evaluar el desempeño del sistema, identificar tendencias de matrícula y tomar decisiones informadas sobre la oferta académica. Estos análisis incluyen datos de participación por facultad, carrera y año académico, así como indicadores de rendimiento.

Interacciones del Profesor

Los profesores constituyen los proveedores de contenido académico dentro del sistema, con responsabilidades que abarcan desde la concepción inicial de las asignaturas hasta la evaluación final de los estudiantes inscritos en el curso habilitado.

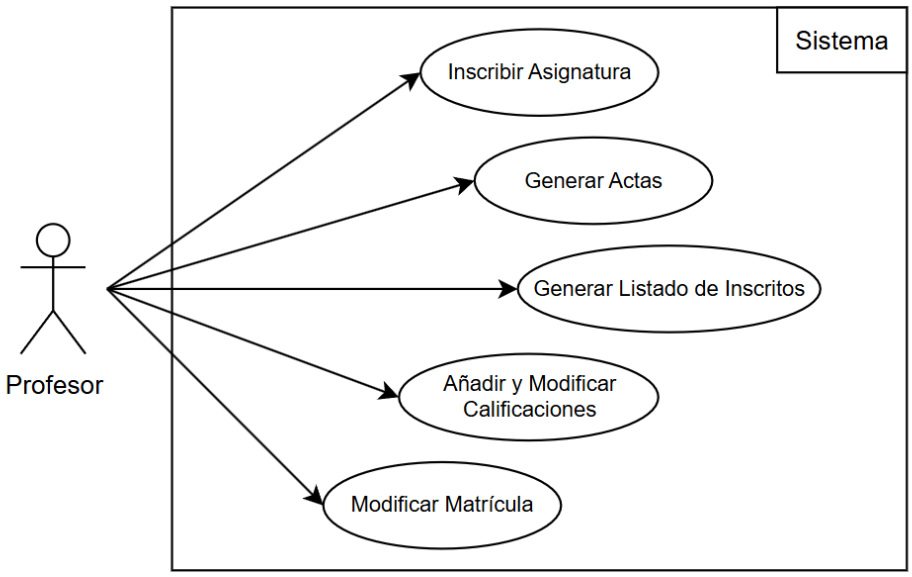

Figura 2.2: Diagrama de casos de uso del Profesor. Los óvalos simbolizan las distintas funcionalidades del sistema, mientras que las flechas muestran la interacción directa del actor con cada caso de uso.

La inscripción de asignaturas representa el punto de entrada del profesor al sistema, requiriendo la presentación de información detallada sobre el curso propuesto. Este proceso incluye la especificación de datos generales, información del docente responsable y colaboradores, contenidos académicos, metodología de evaluación y re-

quisitos para los estudiantes inscritos en el curso habilitado. Estas interacciones se representan en la Figura 2.2.

Una vez aprobada una asignatura, el profesor adquiere capacidades de gestión académica que incluyen la administración de la matrícula estudiantil. Esta función permite realizar ajustes en la lista de participantes según criterios académicos específicos y particulares, respetando siempre los límites de capacidad establecidos y las reglas institucionales vigentes. Además, el profesor tiene la facultad de asignar y modificar las calificaciones finales de los estudiantes inscritos.

Además, el profesor puede acceder al sistema para generar listados actualizados de estudiantes inscritos en sus asignaturas, así como el acta académica correspondiente a la evaluación final del curso. A través de una interfaz sencilla, el docente selecciona el curso y el sistema genera automáticamente los documentos con el formato institucional requerido, incluyendo espacios destinados a firmas y sellos según la normativa universitaria. Esta funcionalidad facilita al profesor la gestión administrativa y documental del proceso educativo.

Interacciones del Estudiante

Los estudiantes representan el eje central del sistema, con acceso a funcionalidades específicamente diseñadas para facilitar su participación en el proceso de selección e inscripción de asignaturas electivas. Estas interacciones se detallan en la Figura 2.3, que presenta el diagrama de casos de uso correspondiente al perfil de estudiante.

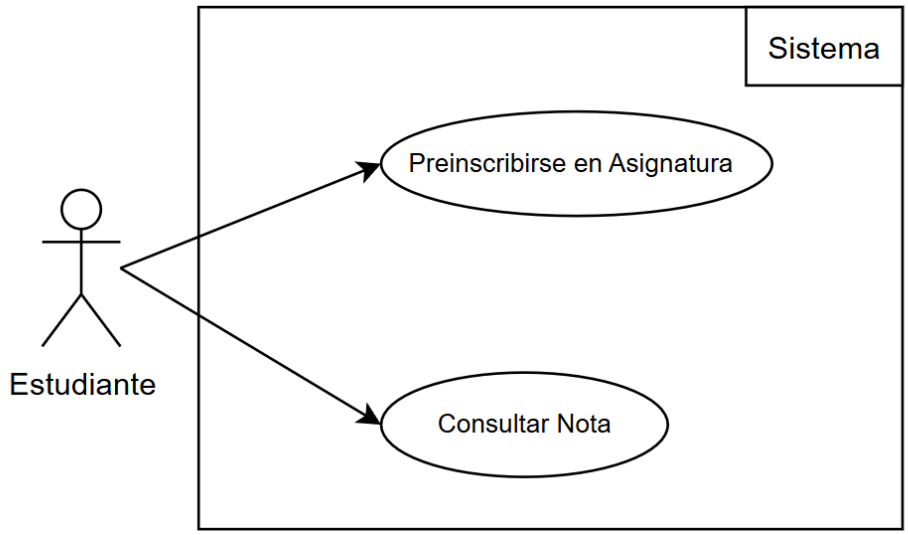

Figura 2.3: Diagrama de casos de uso del Estudiante. Los óvalos simbolizan las distintas funcionalidades del sistema, mientras que las flechas muestran la interacción directa del actor con cada caso de uso.

La consulta de las asignaturas disponibles proporciona a los estudiantes acceso personalizado al conjunto de cursos electivos que han sido aprobados y habilitados para su perfil académico específico (como carrera, año u otras condiciones), según las reglas definidas por la administración. Esta funcionalidad presenta información detallada sobre cada asignatura, incluyendo objetivos, contenidos, modalidad de impartición, requisitos y disponibilidad de cupos.

El proceso de inscripción en asignaturas permite a los estudiantes formalizar su participación en cursos de su interés, sujeto a la disponibilidad de cupos y el cumplimiento de los requisitos establecidos. El sistema implementa validaciones automáticas que verifican la elegibilidad del estudiante según las reglas configuradas por el administrador.

Además de las funcionalidades relacionadas con la selección e inscripción, el sistema también ofrece herramientas para el seguimiento académico. La consulta de calificaciones proporciona acceso a la información académica personal, permitiendo a los estudiantes monitorear su progreso y desempeño en las asignaturas en las que participan. Esta funcionalidad mantiene un registro histórico completo de todas las evaluaciones recibidas.

2.5. Concepción y diseño de la base de datos

Con el fin de estructurar y gestionar eficientemente la información del sistema, se ha diseñado una base de datos relacional que contempla los principales conjuntos de entidades involucrados. Este diseño se basa en los requerimientos derivados de la problemática expuesta en la Sección 2.1, donde se identificaron las necesidades del sistema y los procesos clave a resolver. A continuación, se describen los principales conjuntos de entidades involucrados junto a sus atributos e iterrelaciones clave.

Definicion de Entidades:

- **Curso**: ID (PK[1]), Título, ID Centro (FK[2] → Centro), Modalidad, Capacidad de matrícula, ID Profesor principal (FK → Profesor), Justificación del curso, Objetivo general, Objetivos específicos, Temario, Bibliografía básica, Bibliografía complementaria, Sistema de evaluación, Justificación de modalidad, Requisitos básicos, Lugar de encuentro, Ejes estratégicos, Sectores estratégicos, Ruta de carta de autorización, URL de imagen, Fecha de creación, Fecha de actualización, Estatus de aprobación.

[1]PK (Primary Key): Llave primaria del conjunto de entidades.

[2]FK (Foreign Key): Llave foránea presente en un conjunto de entidades y referencia a otro conjunto.

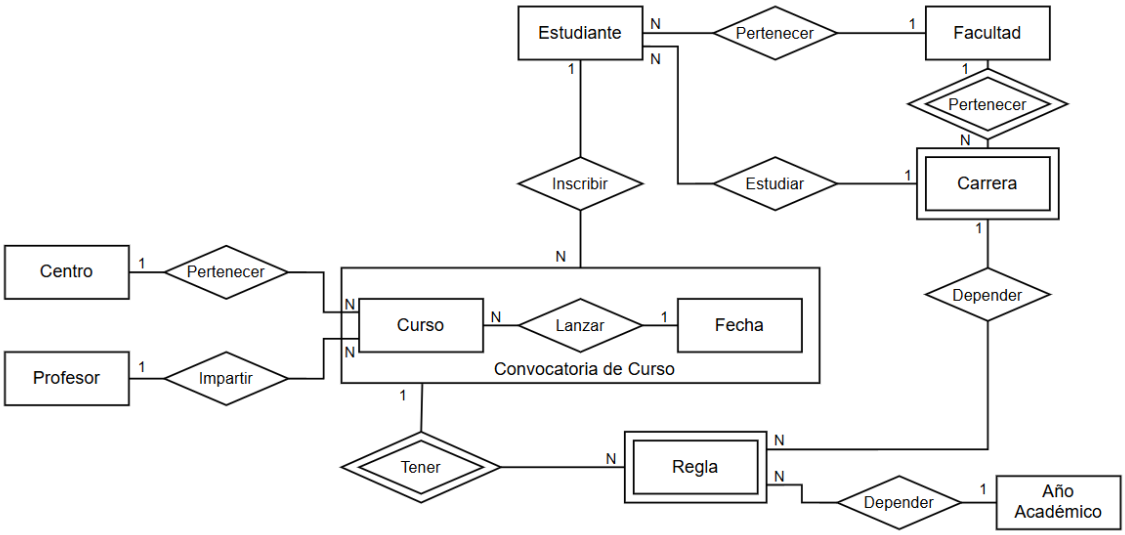

Figura 2.4: Modelo Entidad-Relacional Extendido de la base de datos. Los rectángulos son las entidades, los rombos las relaciones, y los que tienen doble contorno son las entidades débiles del modelo.

- **Convocatoria de Curso**: ID (PK), ID Curso (FK → Curso), Fecha de inicio, Fecha de fin.
- **Aplicación a Curso**: ID (PK), ID Estudiante (FK → Estudiante), ID Curso Aprobado (FK → Curso Aprobado), Carta de motivación, Año académico, Estatus, Fecha de aplicación.
- **Calificación de Curso**: ID (PK), ID Curso Aprobado (FK → Curso Aprobado), ID Estudiante (FK → Estudiante), Calificación, Comentario.
- **Profesor**: ID Usuario (PK), Nombre completo, Email principal, Categoría docente, Grado académico, Teléfono fijo, Número de teléfono, Email secundario, Estatus de aprobación.
- **Estudiante**: ID Usuario (PK), Nombre completo, Número de identificación, Email principal, Nombre de usuario Evea, Número de teléfono, ID Facultad (FK → Facultad), ID Carrera (FK → Carrera), Email secundario.
- **Centro**: ID (PK), Nombre.
- **Facultad**: ID (PK), Nombre.
- **Carrera**: ID (PK), Nombre.

Para facilitar la comprensión de la estructura de la base de datos, el diagrama de la Figura 2.4 ilustra el Modelo Entidad-Relacionalidad-Extendido (MERX) del problema, que refleja los conjuntos de entidades principales y sus interrelaciones clave.

Este modelo proporciona una visión integral de la estructura lógica del sistema y constituye la base para la implementación física de la base de datos. La claridad de las relaciones entre entidades facilita tanto la normalización como la integridad referencial, aspectos fundamentales para garantizar la consistencia de los datos y la eficiencia en su manipulación. En el capítulo posterior se abordará la implementación concreta del modelo, así como los mecanismos que asegurarán su correcto funcionamiento dentro del sistema propuesto.

Capítulo 3

Detalles de Implementación y Experimentación

Toda solución matemática-computacional, una vez concebida y diseñada, requiere una implementación técnica rigurosa para transformarse en una herramienta funcional y accesible para los usuarios finales. En este capítulo se abordan los aspectos fundamentales del proceso de desarrollo de la plataforma de gestión de las asignaturas electivas, desde la selección de tecnologías hasta la validación experimental del sistema implementado. Se detallan las decisiones técnicas adoptadas, justificando la elección de cada componente tecnológico en función de los requerimientos identificados y las particularidades del contexto académico universitario.

El proceso de implementación se estructura en múltiples dimensiones que abarcan desde la arquitectura de datos hasta la experiencia de usuario final. Se presenta un análisis exhaustivo de las herramientas tecnológicas seleccionadas, considerando criterios de escalabilidad, mantenibilidad, seguridad y compatibilidad con el ecosistema tecnológico institucional. Asimismo, se documentan las pruebas experimentales realizadas para validar tanto la funcionalidad técnica como la efectividad operacional de la solución propuesta.

3.1. Tecnologías y herramientas utilizadas para el desarrollo del sistema

La arquitectura tecnológica del sistema incorpora herramientas de código abierto, buscando optimizar la sostenibilidad y facilitar la autonomía tecnológica de la institución, aunque también se ha integrado software propietario como SQL Server por su confiabilidad y soporte en diferentes entornos. La selección de cada componente tecnológico se fundamenta en un análisis comparativo que evalúa criterios de ren-

dimiento, escalabilidad, seguridad, facilidad de mantenimiento y compatibilidad con estándares de desarrollo empresarial.

El enfoque metodológico adoptado para la selección tecnológica considera la necesidad de facilitar la colaboración entre desarrolladores y la incorporación de nuevos miembros al equipo de desarrollo en futuras iteraciones del proyecto. Esta consideración resulta fundamental para garantizar la continuidad del desarrollo y el mantenimiento a largo plazo de la plataforma académica.

3.1.1. Sistema de Gestión de Base de Datos: SQL Server

Para la gestión de datos del sistema se ha seleccionado Microsoft SQL Server[1] como sistema de gestión de base de datos relacional. Esta elección se fundamenta en la robustez y confiabilidad que caracteriza a esta plataforma de base de datos empresarial, así como en su amplia adopción en entornos académicos y corporativos que garantiza la disponibilidad de recursos técnicos especializados.

SQL Server proporciona funcionalidades avanzadas de seguridad, copia de respaldo y recuperación que resultan esenciales para la protección de datos académicos sensibles. Su arquitectura permite implementar mecanismos de control de acceso granular y auditoría detallada, aspectos fundamentales para el cumplimiento de regulaciones de protección de datos en el contexto educativo. Adicionalmente, las capacidades de escalabilidad horizontal y vertical de la plataforma aseguran el soporte adecuado para el crecimiento proyectado de usuarios y volumen de información académica.

3.1.2. Framework de Desarrollo Backend: ASP.NET Core

Para la implementación de la lógica de negocio y los servicios del sistema se ha adoptado ASP.NET Core[2] como framework de desarrollo backend. Esta tecnología de Microsoft representa una plataforma de desarrollo moderna, multiplataforma y de alto rendimiento que facilita la creación de aplicaciones web empresariales escalables y seguras.

La elección de ASP.NET Core se justifica por su arquitectura modular que permite el desarrollo de servicios web RESTful[3] con capacidades avanzadas de autenticación, autorización y gestión de sesiones. El framework proporciona herramientas integradas para la implementación de patrones de diseño empresariales como la inyección de dependencias, el control de versiones de API y la documentación automática de

[1] https://www.microsoft.com/sql-server

[2] https://dotnet.microsoft.com/apps/aspnet

[3] Se refiere a una Interfaz de Programación de Aplicaciones (API) que sigue los principios de REST (Representational State Transfer).

servicios. Estas características facilitan el mantenimiento del código y la colaboración entre desarrolladores en equipos distribuidos.

3.1.3. Mapeo Objeto-Relacional: Entity Framework Core

Para la abstracción de la capa de acceso a datos se utiliza Entity Framework Core[4], el mapeador objeto-relacional oficial de Microsoft para .NET. Esta herramienta facilita la interacción con la base de datos mediante el uso de objetos .NET, eliminando la necesidad de escribir código SQL específico para la mayoría de operaciones de acceso a datos.

Entity Framework Core proporciona funcionalidades avanzadas como migraciones automáticas de los esquemas de bases de datos, carga diferida de entidades relacionadas y optimización automática de consultas. Estas características reducen significativamente el tiempo de desarrollo y minimizan la posibilidad de errores en la implementación de operaciones de persistencia de datos. Además, su integración nativa con ASP.NET Core facilita la implementación de patrones arquitectónicos como Repository y Unit of Work.

3.1.4. Tecnología de Desarrollo Frontend: React.js

Para la implementación de la interfaz de usuario se ha seleccionado React.js[5], una biblioteca de JavaScript desarrollada por Meta que facilita la construcción de interfaces de usuario interactivas y adaptables. React.js se ha consolidado como una de las tecnologías frontend más utilizadas en el desarrollo de aplicaciones web modernas, con un ecosistema maduro y una comunidad activa de desarrolladores.

La arquitectura basada en componentes de React permite la creación de interfaces modulares y reutilizables que facilitan el mantenimiento y la extensión de funcionalidades. Su modelo de gestión de estado y el virtual DOM[6] proporcionan un rendimiento óptimo para aplicaciones con alta interactividad, aspecto fundamental para la experiencia de usuario en procesos académicos que requieren navegación fluida entre múltiples formularios y vistas de datos.

[4]https://learn.microsoft.com/ef/core/

[5]https://react.dev/

[6]El DOM (Document Object Model) es una estructura jerárquica que representa los elementos de una página web como objetos, permitiendo que los lenguajes de programación, como JavaScript, accedan y manipulen el contenido, la estructura y el estilo de la página.

3.1.5. Despliegue de la aplicación: Docker

Para el empaquetado y despliegue de la aplicación se ha adoptado Docker[7] como plataforma de contenedorización. Esta tecnología permite encapsular la aplicación y todas sus dependencias en contenedores ligeros y portátiles que garantizan la consistencia del entorno de ejecución independientemente de la infraestructura de despliegue.

La utilización de Docker facilita la implementación de prácticas de integración continua y despliegue continuo, aspectos fundamentales para el mantenimiento eficiente de aplicaciones en producción. Los contenedores Docker proporcionan aislamiento de procesos y recursos que mejora la seguridad y estabilidad del sistema, mientras que su portabilidad garantiza la flexibilidad de migración entre diferentes entornos de infraestructura sin dependencias externas.

La estrategia de contenedorización adoptada contempla la separación de la aplicación en múltiples contenedores especializados para cada componente del sistema, incluyendo el frontend React, el backend ASP.NET Core y la base de datos SQL Server. Esta arquitectura de microservicios facilita el escalado independiente de cada componente según las demandas específicas de carga y proporciona mayor flexibilidad para actualizaciones y mantenimiento del sistema.

3.2. Arquitectura del sistema implementado

El sistema de gestión de asignaturas electivas está desarrollado en una arquitectura cliente-servidor que facilita la separación clara entre la lógica de presentación y la lógica de negocio. Esta decisión arquitectónica resulta fundamental para un sistema académico que debe atender simultáneamente las necesidades de múltiples tipos de usuarios —profesores, estudiantes y administradores—, cada uno con diferentes niveles de acceso y funcionalidades específicas dentro del proceso de gestión de asignaturas electivas.

La adopción de esta arquitectura distribuida proporciona beneficios significativos para el contexto universitario. Además, la escalabilidad independiente de cada componente es crucial para un sistema que debe manejar picos de demanda durante períodos específicos del ciclo académico, como las fechas de inscripción o la publicación de nuevas convocatorias de cursos.

La persistencia de datos se gestiona mediante SQL Server, que almacena de forma centralizada toda la información relacionada con cursos, profesores, estudiantes, aplicaciones y calificaciones. La comunicación entre el Backend y la base de datos se establece a través de Entity Framework Core como ORM, lo que facilita el mapeo

[7] https://www.docker.com/

objeto-relacional y proporciona una capa de abstracción que simplifica las operaciones de acceso a datos. Esta configuración garantiza que el acceso a la información sea controlado exclusivamente a través del servidor, manteniendo la integridad y seguridad de los datos académicos.

La infraestructura completa del sistema se orquesta mediante Docker y Docker Compose, tecnologías de contenedorización que encapsulan cada componente de la aplicación junto con sus dependencias específicas. Esta aproximación resulta especialmente valiosa en el entorno universitario, donde la portabilidad y la facilidad de despliegue son aspectos críticos para la implementación en diferentes servidores y la gestión de actualizaciones. Los contenedores permiten crear entornos de desarrollo y producción consistentes, reduciendo significativamente los problemas relacionados con diferencias de configuración entre sistemas.

Una vez presentada la estructura general de la aplicación, a continuación se detallan los componentes que conforman cada parte y cómo están construidos.

3.2.1. Desarrollo del Frontend

El frontend de la aplicación, que constituye la interfaz directa de interacción para los usuarios, fue desarrollado como una Single Page Application (SPA) utilizando la biblioteca JavaScript React[8] en conjunto con TypeScript. Esta combinación fue seleccionada debido a la robustez y escalabilidad que ofrece React en la construcción de interfaces de usuario dinámicas y reutilizables, así como a las ventajas de TypeScript en cuanto a mantenimiento y fiabilidad, gracias a su sistema de tipado estático.

La arquitectura del frontend se basa en un diseño modular y basado en componentes, una característica intrínseca de React. La interfaz se descompuso en una jerarquía de componentes reutilizables, como tarjetas de cursos, formularios de inscripción, tablas de datos y paneles de navegación. Esta estructura no solo facilita el mantenimiento y la escalabilidad del sistema, sino que también promueve la reutilización del código y una experiencia de desarrollo más ordenada. La organización del proyecto se distribuyó en carpetas específicas para componentes, vistas principales (páginas), estilos, recursos estáticos (assets) y tipos de datos.

Para el desarrollo y empaquetado de la aplicación se utilizó Vite[9], una herramienta moderna que proporciona un entorno de desarrollo optimizado, con recarga en caliente y compilación eficiente para producción.

La gestión del estado global, como la información de autenticación y los permisos de acceso, se implementó mediante la API Context de React y hooks personalizados, lo que permitió mantener la coherencia del estado durante toda la navegación. El estado local de los componentes se manejó internamente en cada uno de ellos según

[8]https://react.dev/
[9]https://vitejs.dev/

su funcionalidad. Para la navegación entre las diferentes secciones de la aplicación se utilizó React Router, que facilitó la definición de rutas declarativas y una experiencia fluida sin recargas de página.

La comunicación con el backend se llevó a cabo mediante una API RESTful, consumiendo los distintos puntos de acceso a través de solicitudes HTTP (GET, POST, PUT, DELETE), gestionadas con la biblioteca Axios, la cual simplificó el envío de peticiones y el manejo de respuestas, incluyendo la gestión de errores y transformación de datos. En las vistas destinadas a la administración, donde se requiere manipulación avanzada de grandes volúmenes de datos, se integró la biblioteca AG-Grid[10] , que proporciona funcionalidades como filtrado, ordenación y paginación de tablas de manera eficiente.

En cuanto al diseño visual, se empleó SCSS[11] como preprocesador de estilos, lo que permitió organizar las hojas de estilo de forma más clara mediante variables, anidamiento y mixins. La interfaz fue diseñada siguiendo principios de usabilidad, accesibilidad y diseño responsivo, con el objetivo de ofrecer una experiencia intuitiva y eficiente tanto en dispositivos de escritorio como móviles.

Además, se incorporaron validaciones en los formularios para garantizar la integridad de los datos introducidos por los usuarios, acompañadas de mensajes claros de retroalimentación ante errores o acciones exitosas. Es válido mencionar la confección de un diagrama de cómo se comunica cada componente dentro de la arquitectura del sistema, el cual se mostrará en un epígrafe posterior. Por su parte, el sistema implementa diferentes vistas y flujos de trabajo según el rol del usuario, lo cual se detallará en el siguiente epígrafe.

3.2.2. Principales vistas

A continuación, se describen las principales vistas del sistema según el rol del usuario. Cada rol (Administrador, Profesor y Estudiante) tiene acceso a un conjunto específico de funcionalidades diseñadas para facilitar sus tareas dentro del ciclo de vida de los cursos electivos.

Explicación de las vistas desde el Administrador del sistema

El usuario Administrador tiene acceso completo a todas las funcionalidades del sistema, incluyendo la aprobación de cursos, administración de convocatorias y supervisión general de las actividades académicas. El administrador cuenta con herramientas especializadas para mantener la integridad y coherencia de la información del sistema, así como para supervisar el proceso de matrícula en cursos electivos.

[10]https://www.ag-grid.com/

[11]https://sass-lang.com/guide

Las vistas a las que el administrador tiene acceso son:

- **Catálogo de Cursos:**
  Presenta una vista comprehensiva de todos los cursos registrados en el sistema, organizados mediante tarjetas visuales que muestran información esencial como título, centro académico y modalidad. Incluye un sistema de filtrado por estado de aprobación (Todos, Aprobados, Pendientes) que permite al administrador identificar rápidamente cursos que requieren revisión. Proporciona acceso directo a la información detallada de cada curso para procesos de aprobación.

- **Información del Curso:**
  Ofrece una vista detallada y estructurada de cada curso individual, presentando información completa incluyendo datos generales, profesorado, detalles académicos, bibliografía e información estratégica. La vista destaca el estado de aprobación del curso y proporciona funcionalidad directa para aprobar cursos pendientes, actualizando automáticamente el estado en el sistema.

- **Convocatorias de Cursos:**
  Muestra un catálogo visual de todas las convocatorias activas organizadas mediante tarjetas que incluyen título del curso, centro, fechas de inicio y fin. La vista proporciona acceso directo a herramientas de lanzamiento de nuevas convocatorias y permite la navegación hacia la gestión detallada de cada convocatoria específica.

- **Lanzar Nueva Convocatoria:**
  Proporciona una interfaz especializada para crear nuevas convocatorias de cursos electivos. Presenta un formulario estructurado que permite seleccionar cursos aprobados, establecer fechas de inicio y fin con validaciones automáticas, y crear instancias de cursos con confirmaciones de éxito.

- **Gestión de Convocatoria:**
  Ofrece una interfaz comprehensiva para administrar una convocatoria existente, permitiendo la edición de fechas mediante formularios validados y la gestión de reglas de acceso. La vista se organiza en secciones especializadas para reglas de filtrado y reglas de prioridad, cada una con capacidades para agregar, eliminar y modificar criterios basados en carreras y años académicos, manteniendo el orden de prioridad para procesos de matrícula.

- **Matrícula del Curso:**
  Presenta la gestión completa de estudiantes matriculados en convocatorias específicas mediante una tabla detallada que muestra información personal y académica de cada estudiante. Incluye funcionalidades para inscribir nuevos estudiantes a través de un selector dropdown, eliminar estudiantes existentes y

proporciona enlaces de navegación hacia información relacionada del curso y calificaciones.

- **Evaluaciones del Curso:**
  Muestra un registro comprehensivo de las calificaciones de estudiantes para convocatorias específicas, presentando información detallada en formato tabular incluyendo datos personales, información académica y resultados de evaluación. La vista incluye capacidades de exportación a Excel con formato personalizado y proporciona navegación integrada hacia otras secciones relacionadas de la convocatoria.

- **Facultades:**
  Muestra un listado de las facultades registradas en el sistema presentado en una tabla interactiva con funcionalidades de filtrado y paginación. Permite visualizar el nombre de cada facultad y proporciona herramientas para agregar nuevas facultades mediante un formulario integrado. La vista incluye capacidades de edición y eliminación masiva de registros seleccionados.

- **Carreras:**
  Presenta un catálogo completo de las carreras académicas disponibles en el sistema. La interfaz utiliza una tabla con capacidades de ordenamiento y filtrado que muestra el nombre de cada carrera. Incluye funcionalidades para crear nuevas carreras a través de un formulario, editar información existente directamente en la tabla y eliminar múltiples registros simultáneamente.

- **Centros:**
  Gestiona los centros académicos del sistema mediante una tabla que muestra el nombre de cada centro. La vista proporciona herramientas completas de administración incluyendo la adición de nuevos centros, edición información existente y eliminación selectiva de registros, manteniendo la consistencia de datos a través de validaciones automáticas.

- **Profesores:**
  Administra el catálogo completo de profesores del sistema presentando información detallada en una tabla con múltiples columnas que incluyen nombre completo, correo primario, categoría docente, grado académico, teléfonos y correo secundario. La vista permite agregar nuevos profesores mediante un formulario comprehensivo, editar información existente y gestionar la eliminación de registros con confirmaciones de seguridad.

Como ejemplo, en la Figura 3.1 se puede observar una parte de la vista de gestión de convocatoria desde un Usuario Administrador.

Figura 3.1: Vista de la página de Gestión de Convocatoria. Se muestra el nombre del curso, enlaces a secciones relacionadas y un formulario para modificar las fechas de la convocatoria. Debajo de esta vista se encuentra la sección para gestionar las reglas de inscripción del curso.

Explicación de las vistas desde el Profesor

El usuario Profesor tiene acceso a un conjunto específico de vistas diseñadas para la gestión completa de cursos electivos, desde su creación hasta la evaluación final de los estudiantes. Estas vistas permiten al profesor controlar todo el ciclo de vida académico de sus asignaturas, incluyendo la propuesta de nuevos cursos, la modificación de información existente y la administración de las instancias activas de sus cursos.

Las vistas a las que el profesor tiene acceso son:

- **Panel de Gestión de Cursos:**
  Esta vista constituye el punto de entrada principal para todas las funcionalidades del profesor. Presenta un diseño tipo panel visual que muestra estadísticas relevantes como el número de cursos creados, cursos activos y estudiantes inscritos. La interfaz proporciona tres opciones principales de navegación: crear nuevos cursos, modificar información de cursos existentes y gestionar cursos activos. Cada opción se presenta como una tarjeta interactiva con iconografía descriptiva y enlaces directos a las funcionalidades correspondientes.

- **Crear Curso Nuevo:**
  Proporciona un formulario integral dividido en secciones temáticas para la creación de nuevos cursos electivos. La vista incluye campos para información básica (título, capacidad de matrícula, centro, modalidad, profesores colaboradores,

lugar de encuentro), contenido académico (fundamentación, objetivos generales y específicos, temario), bibliografía y evaluación (referencias básicas y complementarias, sistema de evaluación, justificación de modalidad, requisitos básicos), clasificación estratégica (ejes y sectores estratégicos) y recursos adicionales (enlace a imagen del curso y carta de autorización en PDF). Todos los campos críticos están marcados como obligatorios y el sistema valida la información antes del envío.

- **Mis Cursos:**
  Muestra un catálogo visual de todos los cursos propuestos por el profesor. Para cada curso se presenta una tarjeta que incluye el título, centro académico, modalidad e imagen representativa. Esta vista sirve como portal de acceso a la información detallada y funciones de edición de cada curso individual.

- **Información del Curso:**
  Presenta una vista detallada y estructurada de toda la información académica de un curso específico. La información se organiza en secciones claramente diferenciadas: información general (capacidad, lugar de encuentro), profesorado colaborador, detalles del curso (justificación, objetivos, temario, requisitos, evaluación), bibliografía (básica y complementaria), información estratégica (ejes y sectores) e imagen del curso. Para cursos no aprobados, incluye un botón de edición que permite modificar la información.

- **Editar Información del Curso:**
  Proporciona un formulario pre-poblado con la información existente del curso, permitiendo la modificación de todos los campos editables. La estructura es idéntica al formulario de creación, pero adaptada para la actualización de datos existentes. El sistema valida las modificaciones y confirma la actualización exitosa.

- **Mis Cursos Activos:**
  Muestra un catálogo de las instancias de cursos que el profesor está impartiendo actualmente. Para cada curso activo se presenta información básica junto con las fechas de inicio y finalización del período académico. Esta vista sirve como punto de acceso a las funcionalidades de gestión académica de cada instancia específica.

- **Matrícula del Curso:**
  Proporciona una tabla interactiva con información completa de todos los estudiantes matriculados en una instancia específica del curso. La tabla incluye datos como nombre del estudiante, facultad, carrera, año académico, correos electrónicos y número telefónico. La vista permite eliminar estudiantes de la

matrícula mediante selección múltiple y también incluye un formulario para agregar manualmente estudiantes que hayan aplicado al curso pero no fueron incluidos automáticamente.

- **Evaluaciones del Curso:**
  Presenta una tabla editable para la gestión de calificaciones de los estudiantes matriculados. Muestra información académica básica de cada estudiante junto con campos editables para las calificaciones finales (Prueba Final, Extraordinario y Mundial) y comentarios adicionales. La vista incluye funcionalidad para generar directamente las actas oficiales, completadas automáticamente con los datos registrados en la tabla y conforme al formato institucional establecido.

Como ejemplo, en la Figura 3.2 se puede observar una parte de la vista de Evaluaciones del Curso desde un Usuario Profesor.

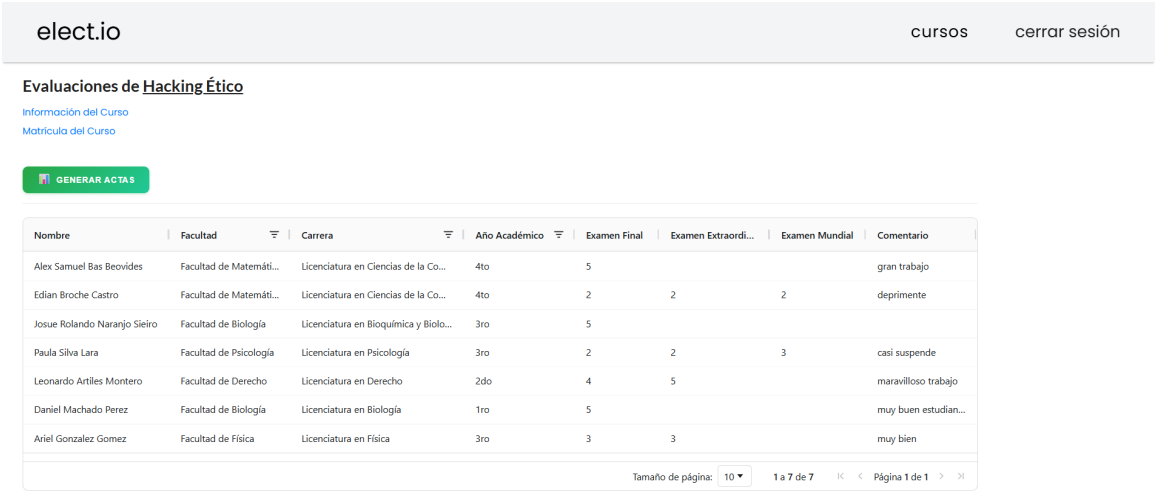

Figura 3.2: Vista de la página de Evaluaciones del Curso. En la parte superior se muestran enlaces a secciones relacionadas con el curso. Debajo se encuentra una tabla con las calificaciones de los estudiantes y sus datos de inscripción.

Explicación de las vistas desde el Estudiante

El rol Estudiante constituye el usuario final principal del sistema, quien puede explorar, solicitar inscripción y gestionar sus cursos electivos. Las vistas disponibles para este rol son:

- **Catálogo de Cursos:**
  Presenta un listado completo de los cursos electivos disponibles para inscripción,

mostrando información esencial como el título del curso, el centro académico que lo imparte, la modalidad (presencial, virtual o híbrida), las fechas de inicio y finalización, y una imagen representativa. La vista implementa un sistema de filtrado automático que solo muestra los cursos para los cuales el estudiante cumple con los requisitos de acceso establecidos en las reglas de elegibilidad, basándose en su carrera y año académico.

- **Información Detallada del Curso:**
  Proporciona una vista comprehensiva de un curso específico, incluyendo información general como fechas, capacidad de matrícula y lugar de encuentro, detalles del profesorado (profesor principal y colaboradores), contenido académico (justificación, objetivos generales y específicos, temario), sistema de evaluación, requisitos básicos, bibliografía (básica y complementaria) e información estratégica sobre ejes y sectores. Esta vista incluye un formulario de preinscripción que permite al estudiante solicitar su inscripción proporcionando una carta de motivación y seleccionando su año académico.

- **Mis Cursos Electivos:**
  Muestra el listado personalizado de los cursos electivos en los que el estudiante se encuentra actualmente matriculado. Para cada curso se presenta información básica como título, centro académico, modalidad e imagen, facilitando la navegación hacia los detalles específicos de cada curso matriculado.

- **Información del Curso Matriculado:**
  Similar a la vista de información detallada del curso, pero adaptada para cursos en los que el estudiante ya está inscrito. Incluye toda la información académica del curso y proporciona un enlace directo para acceder a las calificaciones de los estudiantes inscritos en dicho curso.

- **Evaluaciones del Curso:**
  Presenta una tabla interactiva con las calificaciones de todos los estudiantes matriculados en un curso específico, mostrando información como nombre del estudiante, facultad, carrera, año académico, nota final y comentarios de evaluación. Esta vista utiliza AG-Grid para proporcionar funcionalidades avanzadas de filtrado, ordenación y paginación, y está restringida para mostrar únicamente las calificaciones de cursos en los que el estudiante consultante está matriculado.

3.2.3. Desarrollo del Backend

La implementación del Backend del sistema de gestión de asignaturas electivas se basó en ASP.NET Core, utilizando el lenguaje de programación C#[12]. Esta decisión

[12]https://dotnet.microsoft.com/languages/csharp

tecnológica responde a la necesidad de desarrollar una solución robusta y escalable que pueda gestionar eficientemente las múltiples entidades del dominio académico, incluyendo cursos, profesores, estudiantes, convocatorias, aplicaciones y calificaciones, así como sus complejas interrelaciones dentro del contexto universitario.

La arquitectura del sistema se fundamenta en los principios del Domain-Driven Design (DDD), mencionado y explicado en la Sección 1.2, adoptando específicamente la variante de Arquitectura por Capas (N-Layers) como estrategia de organización del código. Esta elección arquitectónica resulta particularmente apropiada para el dominio académico que se aborda, donde las reglas de negocio, aunque específicas y críticas, mantienen una complejidad manejable que no justifica arquitecturas más sofisticadas.

El Backend expone sus funcionalidades a través de una API RESTful que proporciona puntos de acceso HTTP para realizar operaciones CRUD sobre el conjunto de entidades del sistema. Los diferentes tipos de solicitudes HTTP (GET, POST, PUT, DELETE) permiten a los clientes —ya sean aplicaciones web o móviles— interactuar de manera uniforme con los recursos del servidor.

La estructura del proyecto se organiza en cuatro capas principales: Presentación, Aplicación, Dominio e Infraestructura. Esta separación por capas garantiza una clara separación de responsabilidades, facilita el mantenimiento del código y permite una mayor flexibilidad para futuras modificaciones o extensiones del sistema. Cada capa encapsula aspectos específicos de la lógica del sistema, desde la exposición de puntos de acceso hasta la persistencia de datos, pasando por la implementación de las reglas de negocio académicas específicas de la gestión de asignaturas electivas.

Capa de Presentación

La capa de presentación del sistema constituye el punto de entrada para todas las interacciones externas, implementando controladores que exponen la funcionalidad del sistema a través de puntos de acceso HTTP. Esta capa se estructura siguiendo una organización basada en entidades del dominio académico, donde cada controlador se encarga específicamente de gestionar las operaciones relacionadas con una entidad particular del sistema de asignaturas electivas, como cursos, profesores, estudiantes, convocatorias y aplicaciones.

Los controladores implementan principalmente las operaciones CRUD estándar, proporcionando puntos de acceso que siguen las convenciones RESTful. El patrón típico incluye una ruta GET para recuperar colecciones de recursos con soporte para paginación y filtrado mediante parámetros de consulta, una ruta GET con identificador para obtener recursos específicos, una ruta POST para la creación de nuevos recursos, una ruta PUT para actualizaciones, y una ruta DELETE para eliminaciones. Esta estructura uniforme facilita la comprensión y uso de la API por parte de

los diferentes clientes del sistema. Para el manejo de autenticación y autorización, el sistema utiliza tokens JWT (JSON Web Tokens)[13], que permiten validar la identidad del usuario y sus privilegios en cada solicitud de manera segura y eficiente.

Es importante destacar que no todos los controladores siguen exactamente el mismo patrón de puntos de acceso, ya que las particularidades del dominio académico requieren adaptaciones específicas. Algunos controladores, como el de autenticación, incluyen puntos de acceso especializados para el registro diferenciado de profesores y estudiantes, cada uno con sus propios requisitos de información y validación. Del mismo modo, controladores relacionados con entidades de solo lectura o con comportamientos específicos del proceso académico pueden implementar un subconjunto de operaciones CRUD o incluir rutas adicionales para funcionalidades particulares del dominio universitario.

Cabe resaltar que la capa de presentación contiene exclusivamente controladores, los cuales no implementan lógica de negocio directamente. En su lugar, delegan dicha lógica a los servicios definidos en la capa de aplicación. Esta separación de responsabilidades promueve una arquitectura limpia y facilita el mantenimiento, la reutilización de código y la realización de pruebas unitarias e integradas.

Capa de Aplicación

La capa de aplicación constituye el núcleo funcional del sistema, encargándose de implementar la lógica de negocio y coordinar los casos de uso específicos de la aplicación. Esta capa se estructura en dos componentes fundamentales que trabajan de manera complementaria para procesar las solicitudes del usuario y generar las respuestas apropiadas.

Los Data Transfer Objects (DTOs) representan el primer componente de esta arquitectura, funcionando como estructuras de datos especializadas que definen el formato y la estructura de la información intercambiada entre las diferentes capas del sistema. Estos objetos en C# encapsulan los datos que conforman el cuerpo de las peticiones HTTP o los parámetros de consulta requeridos por los puntos de acceso de la API. Los DTOs incorporan atributos de validación que aseguran la integridad y consistencia de los datos recibidos, implementando reglas de negocio básicas como validación de formatos, rangos de valores y campos obligatorios antes de que la información sea procesada por las capas superiores.

El segundo componente lo constituyen los Servicios de Aplicación, clases especializadas que implementan la lógica de los casos de uso del sistema. Cada servicio se organiza típicamente en torno a una entidad o dominio específico de la aplicación, conteniendo métodos que representan operaciones de negocio concretas. Estos servicios actúan como intermediarios entre los controladores de la capa de presentación

[13]https://jwt.io/

y las funcionalidades de las capas inferiores, orquestando las operaciones necesarias para completar cada caso de uso.

La implementación de los servicios sigue el patrón de separación de responsabilidades, donde cada método corresponde a un caso de uso específico que será invocado desde los controladores al recibir peticiones en los puntos de acceso correspondientes. Esta relación directa entre rutas y casos de uso facilita el mantenimiento del código .

Es importante destacar que los servicios de aplicación no interactúan directamente con la base de datos, sino que utilizan el patrón Repository implementado en la capa de infraestructura. Esta abstracción permite que los servicios se concentren exclusivamente en la lógica de negocio, delegando las operaciones de persistencia a los repositorios correspondientes, lo que resulta en un diseño más flexible y testeable que facilita el mantenimiento y la evolución del sistema.

Capa de Infraestructura

La capa de infraestructura constituye el nivel más bajo de la arquitectura, proporcionando las implementaciones concretas de los servicios y funcionalidades que requieren interacción con sistemas externos. Esta capa encapsula todas las dependencias técnicas y de infraestructura, manteniendo aislada la lógica de negocio de los detalles de implementación específicos. A continuación, se detallan los principales componentes de esta capa: los repositorios, el contexto de datos, las migraciones y el inicializador de datos, cada uno con un papel clave en la gestión de la persistencia y configuración técnica del sistema.

Los **Repositorios** representan la implementación concreta del patrón Repository definido en las interfaces del dominio. Utilizando Entity Framework Core como ORM (Object-Relational Mapping), estos repositorios proporcionan las operaciones de acceso a datos necesarias para la persistencia y recuperación de información. Cada repositorio se especializa en el manejo de una entidad específica, implementando tanto las operaciones CRUD básicas como consultas personalizadas requeridas por la lógica de negocio. La utilización de Entity Framework Core permite abstraer las complejidades del acceso a datos, proporcionando un mecanismo robusto para la traducción entre objetos del dominio y estructuras relacionales.

El **Contexto de Datos** (DbContext) actúa como el punto central de configuración y coordinación para todas las operaciones de Entity Framework Core. Este componente define el contexto de la base de datos, estableciendo las relaciones entre entidades, configurando las restricciones de integridad referencial y gestionando el ciclo de vida de las conexiones a la base de datos. El contexto también incluye la configuración de mapeo objeto-relacional, definiendo cómo las entidades del dominio se corresponden con las tablas y columnas en SQL Server.

Las **Migraciones** constituyen el mecanismo de versionado y evolución del es-

quema de base de datos. Estas representan scripts generados automáticamente que describen los cambios estructurales necesarios para mantener la base de datos sincronizada con el modelo de datos definido en el código. Las migraciones permiten aplicar cambios de esquema de manera controlada y reversible, facilitando el despliegue de actualizaciones tanto en entornos de desarrollo como de producción, y asegurando la consistencia del modelo de datos a lo largo del ciclo de vida de la aplicación.

El **Inicializador de Datos** (Database Seeder) implementa la funcionalidad de inicialización y poblado de datos de la base de datos. Este componente se encarga de insertar datos iniciales necesarios para el correcto funcionamiento de la aplicación, incluyendo datos de configuración, usuarios por defecto y cualquier información base requerida por el sistema. El inicializador ejecuta estas operaciones de manera idempotente, verificando la existencia de los datos antes de su inserción para evitar duplicaciones y garantizar la integridad del proceso de inicialización.

La implementación de esta capa garantiza que todas las operaciones de persistencia y configuración de infraestructura se mantengan aisladas del resto de la aplicación, facilitando la testabilidad del código y permitiendo futuros cambios en la tecnología de persistencia sin impactar las capas superiores de la arquitectura.

Capa de Dominio

La capa de dominio constituye el núcleo conceptual de la aplicación, encapsulando las entidades fundamentales del sistema, que representan los conceptos centrales del negocio. Estas entidades están implementadas como clases en C# que modelan los recursos principales del sistema, tales como cursos, profesores, estudiantes y carreras.

Las entidades definen la estructura de los datos del dominio e incorporan validaciones básicas mediante anotaciones, como restricciones de longitud, campos obligatorios o relaciones con otras entidades. Si bien no implementan directamente métodos que encapsulen comportamiento complejo, sí contienen el estado y las relaciones que forman parte del modelo del dominio académico.

La lógica de negocio específica y los casos de uso se delegan a los servicios de aplicación, siguiendo un enfoque pragmático donde el modelo del dominio se mantiene como un conjunto de estructuras coherentes y cohesionadas que representan el estado de la aplicación.

Además, las interfaces de repositorio definen contratos para la persistencia y recuperación de las entidades del dominio. Estas interfaces permiten desacoplar la lógica del dominio de los detalles de infraestructura, facilitando la prueba y el mantenimiento del sistema. La implementación concreta de estos repositorios se encuentra en la capa de infraestructura.

3.2.4. Despliegue con Docker

Para garantizar la portabilidad, escalabilidad y facilidad de despliegue de la aplicación desarrollada, se ha optado por utilizar Docker como herramienta principal de virtualización a nivel de contenedores. Docker permite encapsular cada uno de los componentes de la aplicación en imágenes independientes, asegurando que cada servicio disponga de su propio entorno de ejecución, libre de conflictos con otros servicios o dependencias del sistema operativo anfitrión.

En el presente proyecto, se han definido tres contenedores principales, cada uno correspondiente a un componente fundamental de la arquitectura:

- **Frontend**: Este contenedor ejecuta la aplicación desarrollada en React con TypeScript. La imagen contiene todas las dependencias necesarias para servir la interfaz de usuario, permitiendo su despliegue de manera consistente en cualquier entorno compatible con Docker.
- **Backend**: El segundo contenedor aloja la Web API desarrollada con ASP.NET Core en C#. Esta imagen incluye el entorno de ejecución de .NET 8.0 y las librerías necesarias para la API REST, garantizando compatibilidad y rendimiento.
- **Base de datos**: El tercer contenedor ejecuta una instancia de Microsoft SQL Server, que actúa como sistema de gestión de base de datos relacional para la aplicación. Esta imagen está configurada para exponer los puertos necesarios y persistir los datos de manera adecuada.

Para la orquestación y gestión conjunta de estos contenedores, se ha utilizado **Docker Compose**. Esta herramienta permite definir y ejecutar aplicaciones multicontenedor mediante un archivo de configuración, donde se especifican las imágenes a utilizar, las redes internas, los volúmenes para persistencia de datos y las variables de entorno necesarias para cada servicio. De esta manera, el despliegue completo de la aplicación se reduce a la ejecución de un único comando, simplificando tanto el desarrollo como la puesta en producción.

El uso de Docker y Docker Compose no solo facilita la replicabilidad del entorno de ejecución, sino que también contribuye a la escalabilidad y mantenimiento de la solución, permitiendo la actualización o sustitución de componentes de manera aislada y controlada.

3.3. Pruebas y validación experimental

Con el fin de verificar la validez de la implementación, se llevaron a cabo distintas pruebas unitarias, manuales y de carga. En esta sección se describe el enfoque uti-

lizado para diseñar dichas pruebas y se presenta un análisis inicial de los resultados obtenidos.

3.3.1. Entorno de Pruebas

Con el objetivo de analizar el rendimiento y la eficiencia del sistema implementado, se estableció un entorno de pruebas controlado que permite obtener resultados representativos bajo condiciones similares a las de un uso real. A continuación, se describen los elementos clave del entorno configurado para la evaluación.

Las pruebas fueron ejecutadas en un equipo con las siguientes características técnicas:

- **Procesador**: AMD Ryzen 5 5625U (2.30 GHz, 6 núcleos).
- **Memoria RAM**: 16 GB DDR4.
- **Almacenamiento**: SSD NVMe de 512 GB.
- **Sistema Operativo**: Windows 11 Home.
- **Conexión a la base de datos**: Entorno local.

El equipo utilizado ofrece una capacidad de procesamiento y respuesta acorde con un entorno de operación realista, permitiendo una evaluación precisa del comportamiento del sistema durante la ejecución de consultas y tareas relacionadas.

La configuración del software utilizada para las pruebas fue la siguiente:

- **Backend**: ASP.NET Core 8 con Entity Framework Core.
- **Base de Datos**: SQL Server.
- **Cliente Web**: React 18.3.1 con TypeScript, Vite 6.0.1 y SCSS.
- **Servidor de Pruebas**: Servidor local.
- **APIs y comunicación**: RESTful APIs consumidas mediante llamadas HTTP con `fetch` nativo y Axios.
- **Herramientas de monitoreo**: Swagger para pruebas de API y documentación.
- **Autenticación**: JWT Bearer tokens con ASP.NET Core Identity.

3.3.2. Experimentos Realizados

Para evaluar la efectividad del sistema desarrollado, se llevaron a cabo una serie de experimentos centrados en la presentación de información, el rendimiento del sistema bajo alta concurrencia y el funcionamiento del módulo administrativo. Estos experimentos permitieron verificar que el sistema cumple con los requerimientos del sistema y detectar posibles mejoras en su implementación.

Experimento 1: Evaluación de la Presentación de Información

Objetivo: Verificar que la información de las asignaturas electivas y las diferentes secciones del sistema se presentan de manera estructurada, clara y accesible para los usuarios según su perfil (Administrador, Profesor, Estudiante).

Metodología:

1. Se realizaron pruebas de usabilidad con un grupo de usuarios representativos de los tres perfiles del sistema: administradores de la Dirección de Formación de Pregrado, profesores y estudiantes de diferentes facultades y años académicos.
2. Se evaluó la facilidad de navegación en las funcionalidades principales del sistema, asegurando que las secciones como consulta de asignaturas disponibles, información detallada de cursos, reglas de inscripción y gestión académica fueran comprensibles y organizadas.
3. Se midió el tiempo promedio que tardaban los usuarios en encontrar información específica dentro del sistema, como localizar una asignatura electiva disponible según su perfil académico o acceder a las calificaciones obtenidas.
4. Se verificó que la presentación de las diferentes secciones del sistema (gestión de entidades, estadísticas, reportes académicos) fuera accesible y fácil de entender para cada tipo de usuario.
5. Se solicitó retroalimentación sobre la claridad de la interfaz, la disposición de los datos y la coherencia en la experiencia de usuario según el rol asignado.

Resultados Esperados:

- Presentación estructurada de la información de asignaturas y las funcionalidades del sistema, con un diseño intuitivo adaptado a cada perfil de usuario.
- Accesibilidad rápida a la información académica sin confusión ni ambigüedades, respetando los niveles de acceso correspondientes a cada actor.
- Identificación de mejoras en la interfaz para optimizar la experiencia del usuario y facilitar los procesos de inscripción, gestión y consulta académica.

Experimento 2: Evaluación del Rendimiento del Sistema bajo Condiciones de Alta Concurrencia

Objetivo: Evaluar el comportamiento y la capacidad de respuesta del sistema bajo condiciones de alta demanda, simulando el escenario crítico del lanzamiento de un curso electivo donde múltiples estudiantes intentan aplicar simultáneamente a una asignatura recién publicada.

Metodología:

1. Se diseñó un experimento de pruebas de carga utilizando **Artillery.js**[14], una herramienta de código abierto especializada en pruebas de rendimiento para aplicaciones web, seleccionada por su capacidad para generar múltiples usuarios virtuales concurrentes y simular patrones de tráfico realistas.

2. La configuración del experimento incluyó los siguientes parámetros técnicos: URL objetivo dirigida al entorno de desarrollo local, duración del test de 30 segundos, tasa de llegada de 3 usuarios nuevos por segundo, generando aproximadamente 90 usuarios virtuales únicos durante la prueba.

3. El escenario de prueba replicó el flujo completo de aplicación a un curso electivo, comenzando con la autenticación de cada usuario virtual mediante credenciales únicas generadas dinámicamente, seguido de un tiempo de reflexión de 2 segundos para simular el comportamiento humano real, y culminando con el envío de la solicitud de aplicación al curso, incluyendo carta de motivación y año académico.

4. Se implementó un sistema de usuarios secuenciales para asegurar que cada aplicación proviniera de un usuario diferente, reflejando la realidad académica donde cada estudiante puede aplicar únicamente una vez por curso.

5. El patrón de carga concentrada fue diseñado para generar una carga intensa en un período corto, simulando el comportamiento típico observado en sistemas académicos donde los estudiantes tienden a aplicar inmediatamente después del lanzamiento de un curso popular.

Resultados Esperados:

- Obtención de métricas cuantitativas sobre el tiempo de respuesta de las solicitudes de autenticación y aplicación al curso bajo condiciones de alta concurrencia.

- Medición de la tasa de éxito y fallo de las transacciones, así como el rendimiento del sistema durante picos de demanda.

[14]https://www.artillery.io/

- Evaluación del comportamiento del sistema ante la concurrencia de escritura en la base de datos y identificación de posibles cuellos de botella.
- Determinación de la capacidad máxima del sistema para manejar aplicaciones simultáneas sin degradación significativa del rendimiento o pérdida de datos.

Experimento 3: Evaluación del Módulo Administrativo

Objetivo: Verificar que el módulo administrativo del sistema permita gestionar eficientemente la información académica almacenada, incluyendo asignaturas electivas, usuarios, entidades institucionales y reglas de inscripción, garantizando la integridad y consistencia de los datos.

Metodología:

1. Se probó la creación de nuevas asignaturas electivas y su correcta inserción en la base de datos, validando que toda la información académica (objetivos, contenidos, requisitos, capacidad) se almacene de manera íntegra y estructurada.
2. Se evaluó la funcionalidad de edición y eliminación de datos existentes en entidades del sistema como asignaturas, centros, facultades y carreras y validando la coherencia de los cambios realizados por el administrador.
3. Se verificó la gestión de reglas de inscripción y configuraciones del sistema, permitiendo su creación, edición y actualización por parte del administrador de la Dirección de Formación de Pregrado.
4. Se probó el sistema de autenticación y autorización, validando que solo usuarios con rol de administrador accedan al módulo de administración y que los profesores y estudiantes mantengan acceso restringido según su perfil.
5. Se implementaron pruebas unitarias automatizadas para validar cada funcionalidad del módulo administrativo, incluyendo operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre todas las entidades del sistema, verificando el correcto manejo de casos límite y errores.

Resultados Esperados:

- Correcto almacenamiento y visualización de nuevas asignaturas electivas y demás entidades, con validación automática de la información ingresada.
- Modificaciones reflejadas en tiempo real sin afectar la estructura de la base de datos ni la experiencia de otros usuarios del sistema.

- Eliminación de registros sin inconsistencias ni pérdida de integridad referencial.
- Acceso restringido al módulo administrativo mediante autenticación segura y control de roles diferenciado.
- Funcionamiento correcto de todas las pruebas unitarias implementadas, garantizando la confiabilidad del código y la detección temprana de errores.

3.3.3. Análisis de Resultados

Una vez realizados los experimentos, se recopilaron datos que permitieron evaluar el rendimiento y la eficacia del sistema en condiciones de alta concurrencia, en la presentación de la información y en el módulo administrativo. A continuación, se detallan los resultados obtenidos en cada caso.

Resultados de Experimento 1

Las pruebas de usabilidad demostraron que la estructura y organización de las funcionalidades del sistema facilitaron la consulta y gestión de información académica. Se identificaron los siguientes hallazgos:

- **Claridad en la presentación:**
  La mayoría de los usuarios encontraron la interfaz intuitiva y comprendieron rápidamente la disposición de las secciones dentro de cada perfil de usuario, así como la organización de la información de asignaturas electivas.
- **Tiempo de acceso a la información:**
  En promedio, los usuarios tardaron menos de 5 segundos en localizar una funcionalidad específica dentro de su perfil correspondiente, lo que indica un diseño eficiente adaptado a las necesidades académicas.
- **Visualización de secciones funcionales:**
  Los usuarios pudieron acceder sin dificultades a las diferentes secciones del sistema, incluyendo la consulta de asignaturas disponibles, gestión de inscripciones y acceso a información académica personalizada.
- **Retroalimentación sobre la interfaz:**
  Se recibieron sugerencias para mejorar la organización de ciertos elementos visuales relacionados con la presentación de reglas de inscripción y estadísticas académicas, lo que llevó a pequeños ajustes en el diseño.

En general, los resultados muestran que la representación estructurada de la información cumple con los requerimientos del sistema de gestión de asignaturas electivas, garantizando una navegación clara y accesible para todos los perfiles de usuario.

Resultados de Experimento 2

Las pruebas de carga demostraron que el sistema mantiene un comportamiento estable y confiable bajo condiciones de alta concurrencia. Se identificaron los siguientes hallazgos:

- **Estabilidad del sistema:**
  El sistema procesó exitosamente todas las sesiones de usuario sin fallos críticos, completando los 90 usuarios virtuales creados con una tasa de finalización del 100 %, lo que indica una arquitectura robusta capaz de manejar la carga concurrente.

- **Rendimiento de las transacciones:**
  Se registraron 180 solicitudes HTTP totales con una tasa de éxito del 95 % (171 respuestas exitosas código 200) y únicamente 9 respuestas con código 400, indicando un manejo eficiente de las solicitudes de aplicación al curso electivo.

- **Tiempo de respuesta del sistema:**
  El sistema mantuvo tiempos de respuesta aceptables con una media de 2.4 segundos para todas las solicitudes, demostrando capacidad de procesamiento adecuada durante picos de demanda académica.

- **Comportamiento temporal de la carga:**
  Como se observa en la Figura 3.3, el sistema experimentó un incremento gradual en el número de usuarios activos durante los primeros 10 segundos de la prueba, alcanzando un pico de 24 usuarios concurrentes, seguido de una disminución controlada que confirma el procesamiento exitoso de las solicitudes.

- **Capacidad de procesamiento:**
  El sistema demostró una tasa de procesamiento constante de 6 solicitudes por segundo, manteniendo esta capacidad a lo largo de toda la duración del experimento sin degradación significativa del rendimiento.

En general, los resultados muestran que el sistema posee la capacidad técnica necesaria para manejar escenarios de alta demanda típicos del lanzamiento de cursos electivos, garantizando la integridad de los datos y la disponibilidad del servicio durante períodos críticos de inscripción académica.

Resultados de Experimento 3

Las pruebas en el módulo administrativo confirmaron su funcionalidad y facilidad de uso para la gestión de asignaturas electivas y entidades del sistema académico. Se obtuvieron los siguientes resultados:

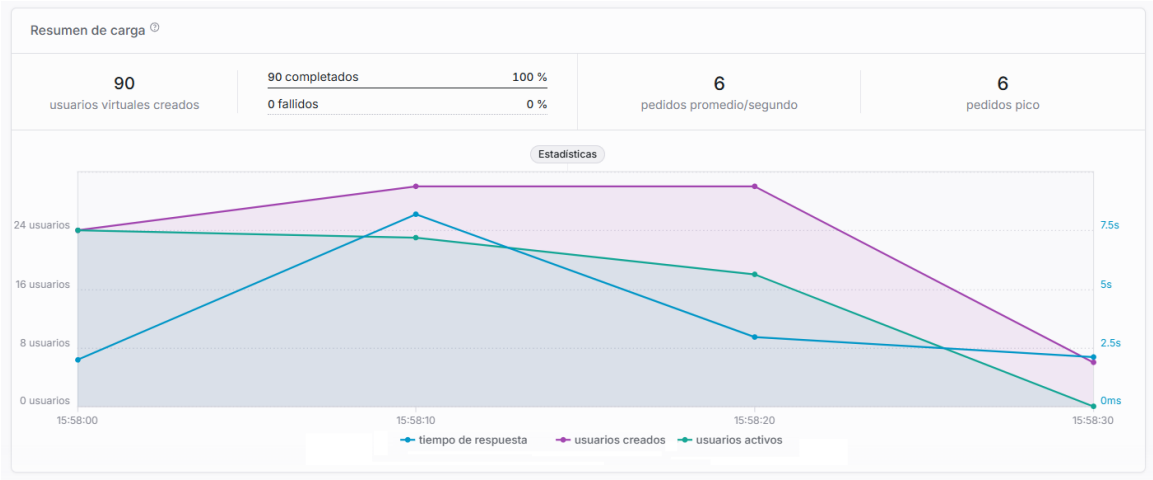

Figura 3.3: Comportamiento del sistema durante la prueba de carga concurrente. La gráfica muestra la evolución temporal de usuarios creados (línea morada), usuarios activos (línea verde) y tiempo de respuesta (línea azul) durante los 30 segundos de ejecución del experimento.

- **Creación de asignaturas:**
  Se logró insertar nuevas asignaturas electivas sin errores, reflejándose correctamente en la base de datos y en la interfaz de usuario, incluyendo toda la información académica requerida (objetivos, contenidos, requisitos y capacidad).

- **Edición y actualización:**
  Los cambios realizados en los registros de asignaturas, usuarios y entidades institucionales se aplicaron sin inconsistencias, manteniendo la integridad referencial de la base de datos.

- **Eliminación de registros:**
  Se verificó que la eliminación de asignaturas, usuarios y entidades no afectara otras partes del sistema, asegurando integridad en la base de datos.

- **Gestión de reglas de inscripción:**
  Se confirmó que los administradores podían crear, editar y actualizar las reglas de inscripción que determinan la elegibilidad de estudiantes para cada asignatura electiva según su perfil académico.

- **Acceso restringido:**
  Solo los usuarios autenticados con rol de administrador pudieron acceder al módulo administrativo, garantizando seguridad en la gestión de la información

académica y manteniendo la separación de responsabilidades por perfil de usuario.

- **Pruebas unitarias:**
  Todas las pruebas unitarias implementadas se ejecutaron exitosamente, validando el correcto funcionamiento de las operaciones CRUD y el manejo de casos límite en las funcionalidades administrativas.

En general, el módulo administrativo cumple con los requerimientos del sistema, permitiendo la gestión eficiente de asignaturas electivas y demás entidades, con acceso restringido para mantener la integridad de los datos.

En conjunto, estas pruebas tanto automáticas como manuales, aseguran que las funciones clave de la lógica de negocio operen de manera adecuada. La plataforma demuestra ser una herramienta eficaz para la gestión de asignaturas electivas universitarias, cumpliendo con los objetivos planteados en el proyecto.

Conclusiones

A partir del desarrollo de este trabajo, se logró implementar un sistema web integral para la gestión de asignaturas electivas en la Universidad de La Habana, ofreciendo una solución tecnológica que responde a las necesidades particulares de la Dirección de Formación de Pregrado (DFP). Esta plataforma permite organizar, supervisar y automatizar el proceso de inscripción, validación y matrícula, cumpliendo con el propósito general de esta investigación.

Se diseñaron funcionalidades específicas para cada actor del proceso educativo, asegurando el control de acceso según rol. En este sentido, se cumplió el objetivo de brindar una interfaz para la propuesta de asignaturas por parte del profesorado, así como un módulo de revisión y validación para el rol administrador, facilitando la toma de decisiones con base en criterios personalizables.

La inscripción estudiantil se implementó respetando los cupos definidos y considerando parámetros como carrera, facultad y año académico, garantizando la equidad en el acceso a los cursos. Asimismo, se automatizó la asignación de estudiantes, la generación de actas académicas y reportes, reduciendo significativamente las tareas manuales y los posibles errores humanos, lo cual mejora la eficiencia del proceso administrativo-académico.

La funcionalidad del sistema fue validada a través de pruebas manuales y automatizadas, incluyendo pruebas de carga con usuarios simultáneos, pruebas unitarias y pruebas de usabilidad con actores reales del proceso. Los resultados obtenidos evidencian que el sistema mantiene un comportamiento estable bajo condiciones de alta demanda, presenta una interfaz clara y accesible, y ofrece una experiencia de usuario coherente según el perfil.

En consecuencia, se considera que los objetivos específicos definidos en esta tesis han sido alcanzados en su totalidad. El sistema desarrollado representa una solución robusta, eficiente y alineada con las normativas y dinámicas organizativas de la Universidad de La Habana.

Recomendaciones

Con base en los resultados obtenidos y las oportunidades de mejora identificadas durante el desarrollo de este sistema de gestión de asignaturas electivas, se proponen las siguientes recomendaciones para futuras investigaciones y versiones del sistema:

1. **Incorporar pruebas de integración al conjunto de validaciones del sistema:**
   El sistema fue sometido a pruebas unitarias de carga y de usabilidad, no obstante se recomienda añadir pruebas de integración que permitan verificar de forma automática la interacción correcta entre los diferentes módulos. Esta práctica contribuiría a detectar posibles inconsistencias en los flujos de datos y garantizar una mayor robustez en el comportamiento general del sistema.

2. **Desarrollar una funcionalidad de recomendación de asignaturas basada en el historial académico y consultas en lenguaje natural:**
   Se sugiere implementar un módulo inteligente que, mediante técnicas de aprendizaje automático, sea capaz de analizar el historial de asignaturas electivas cursadas previamente por el estudiante, así como interpretar consultas expresadas en lenguaje natural sobre intereses personales o preferencias temáticas, para sugerir asignaturas electivas más adecuadas. Esta funcionalidad contribuiría a una personalización más profunda del proceso de inscripción y una mejor alineación entre los cursos y los perfiles estudiantiles.

3. **Diseñar una aplicación móvil complementaria al sistema web:**
   Con el objetivo de mejorar el acceso, la interacción y la frecuencia de uso por parte de los usuarios, se recomienda el desarrollo de una aplicación móvil nativa o híbrida que permita realizar las principales operaciones del sistema (inscripción, seguimiento, consultas, etc.) desde dispositivos móviles. Esta solución incrementaría la accesibilidad, especialmente entre los estudiantes, y fomentaría el uso continuo de la plataforma.

4. **Implementar un sistema de mensajería interna:**
   Se sugiere incluir una funcionalidad de comunicación directa dentro del sistema,

que permita la interacción entre profesores y estudiantes mediante mensajes o chat interno. Esta herramienta favorecería la resolución de dudas y el acompañamiento docente, sin depender de plataformas externas de mensajería.

Estas recomendaciones representan oportunidades de mejora que podrían implementarse en futuras versiones del sistema, contribuyendo a su evolución hacia una plataforma más completa, inteligente y accesible para todos los actores del proceso educativo universitario.

Bibliografía

Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4th). Addison-Wesley Professional. (Vid. pág. 6).

Boyer, J., & Mili, H. (2011). *Agility and Discipline Made Easy: Practices from OpenUP and RUP*. Addison-Wesley Professional. (Vid. pág. 5).

Bratko, I. (2012). *Prolog Programming for Artificial Intelligence* (4th). Addison-Wesley. (Vid. pág. 5).

Clocksin, W. F., & Mellish, C. S. (2003). *Programming in Prolog: Using the ISO Standard* (5th). Springer-Verlag. (Vid. pág. 5).

Conway, M. E. (1968). How do Committees Invent? *Datamation*, *14*(4), 28-31 (vid. pág. 6).

Evans, E. (2004). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional. (Vid. págs. 6, 7, 9).

Fowler, M. (2013). *Domain-Specific Languages*. Addison-Wesley Professional. (Vid. pág. 5).

Giarratano, J., & Riley, G. (2005). *Expert Systems: Principles and Programming* (4th). Cengage Learning. (Vid. pág. 4).

Khononov, V. (2021). *Learning Domain-Driven Design*. O'Reilly Media. (Vid. págs. 7, 11, 12).

Richards, M., & Ford, N. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media. (Vid. págs. 10-12).

Vernon, V. (2013). *Implementing Domain-Driven Design*. Pearson Education. (Vid. pág. 8).

Vernon, V. (2016). *Domain-Driven Design Distilled*. Addison-Wesley Professional. (Vid. pág. 7).

Zeiträg, Y., & Figueira, J. R. (2023). Automatically evolving preference-based dispatching rules for multi-objective job shop scheduling. *Journal of Scheduling*, *26*(3), 289-314. https://doi.org/10.1007/s10951-023-00783-9 (vid. pág. 5).