UNIVERSIDAD DE PIURA

**FACULTAD DE INGENIERÍA**

# Diseño, desarrollo e implementación de un repositorio digital de lecciones aprendidas para la asignatura de proyectos en la Universidad de Piura

Tesis para optar el Título de
Ingeniero Industrial y de Sistemas

**Diego Ignacio Cáceres Saldaña**
**Carlo Ricardo Osores Salgado**

**Asesores:**
**Dr. Ing. Omar Armando Manuel Hurtado Jara**
**Mgtr. Ing. Carlos David Zacarías Vélez**

**Piura, julio de 2025**

**Declaración Jurada de Originalidad del Trabajo Final**

Yo, Diego Ignacio Cáceres Saldaña, egresado del Programa Académico de Ingeniería Industrial y de Sistemas de la Facultad de Ingeniería de la Universidad de Piura, identificado(a) con DNI: 70474081, declaro que:

Soy autor del trabajo final titulado:
"**Diseño, desarrollo e implementación de un repositorio digital de lecciones aprendidas para la asignatura de proyectos en la Universidad de Piura**"
El mismo que presento bajo la modalidad de Tesis para optar el Título profesional de Ingeniero Industrial y de Sistemas.

Que el trabajo se realizó en coautoría con los siguientes alumnos de la Universidad de Piura.

- Carlo Ricardo Osores Salgado, identificado con DNI: 73580511

El texto de mi trabajo final es original y no vulnera los derechos de terceros o, de ser el caso, derechos de los coautores, incluidos los derechos de propiedad intelectual, datos personales, entre otros. En tal sentido, el texto de mi trabajo final no ha sido plagiado total ni parcialmente, para lo cual, he respetado las normas internacionales de citas y referencias de las fuentes consultadas. Asimismo, el texto del trabajo final que presento no ha sido publicado ni presentado antes en cualquier medio electrónico o físico; y que la investigación, los resultados, datos, conclusiones y demás información presentada que atribuyo a mi autoría son veraces.
En caso de detectarse el incumplimiento de lo declarado asumo frente a terceros, la Universidad de Piura y/o la Administración Pública toda responsabilidad que pueda derivarse por el trabajo final presentado. Lo señalado incluye responsabilidad pecuniaria incluido el pago de multas u otros por los daños y perjuicios que se ocasionen.

La asesoría del trabajo estuvo a cargo de los siguientes docentes de la Universidad de Piura:

- Dr. Ing. Omar Armando Manuel Hurtado Jara, identificado con DNI: 18133385
- Mgtr. Ing. Carlos David Zacarías Vélez, identificado con DNI: 42176937

Declaro (declaramos) que:
Luego de haber empleado el software de coincidencia Turnitin, revisado las fuentes de información señaladas por el autor, y en razón de mi (nuestra) experiencia como investigador(es), declaro (declaramos) que las ideas expuestas en el trabajo final alcanzan las condiciones de calidad, integridad y originalidad acorde a los objetivos institucionales y estándares en materia de investigación. Finalmente, no asumo (asumimos) responsabilidad por la posible vulneración de derechos de autor en el trabajo final referido, pues tal responsabilidad es exclusiva del autor.

Fecha: 16/07/2025.

……
*Firma del asesor*[1]

……
*Firma del autor*[1]

……
*Firma del co-asesor*[1]

**Declaración Jurada de Originalidad del Trabajo Final**

Yo, Carlo Ricardo Osores Salgado, egresado del Programa Académico de Ingeniería Industrial y de Sistemas de la Facultad de Ingeniería de la Universidad de Piura, identificado(a) con DNI: 73580511, declaro que:

Soy autor del trabajo final titulado:
"**Diseño, desarrollo e implementación de un repositorio digital de lecciones aprendidas para la asignatura de proyectos en la Universidad de Piura**"
El mismo que presento bajo la modalidad de Tesis para optar el Título profesional de Ingeniero Industrial y de Sistemas.

Que el trabajo se realizó en coautoría con los siguientes alumnos de la Universidad de Piura.

- Diego Ignacio Cáceres Saldaña, identificado con DNI: 70474081

El texto de mi trabajo final es original y no vulnera los derechos de terceros o, de ser el caso, derechos de los coautores, incluidos los derechos de propiedad intelectual, datos personales, entre otros. En tal sentido, el texto de mi trabajo final no ha sido plagiado total ni parcialmente, para lo cual, he respetado las normas internacionales de citas y referencias de las fuentes consultadas. Asimismo, el texto del trabajo final que presento no ha sido publicado ni presentado antes en cualquier medio electrónico o físico; y que la investigación, los resultados, datos, conclusiones y demás información presentada que atribuyo a mi autoría son veraces.
En caso de detectarse el incumplimiento de lo declarado asumo frente a terceros, la Universidad de Piura y/o la Administración Pública toda responsabilidad que pueda derivarse por el trabajo final presentado. Lo señalado incluye responsabilidad pecuniaria incluido el pago de multas u otros por los daños y perjuicios que se ocasionen.

La asesoría del trabajo estuvo a cargo de los siguientes docentes de la Universidad de Piura:

- Dr. Ing. Omar Armando Manuel Hurtado Jara, identificado con DNI: 18133385
- Mgtr. Ing. Carlos David Zacarías Vélez, identificado con DNI: 42176937

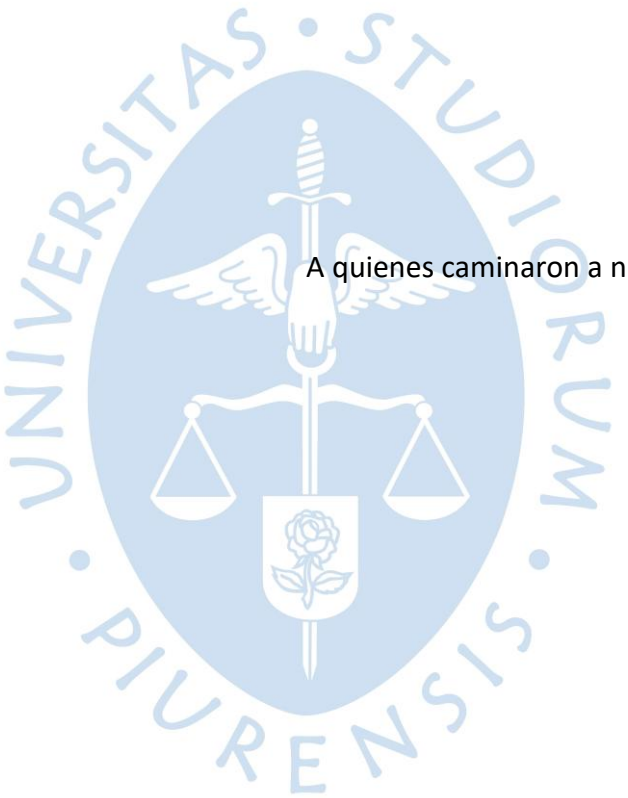
Declaro (declaramos) que:
Luego de haber empleado el software de coincidencia Turnitin, revisado las fuentes de información señaladas por el autor, y en razón de mi (nuestra) experiencia como investigador(es), declaro (declaramos) que las ideas expuestas en el trabajo final alcanzan las condiciones de calidad, integridad y originalidad acorde a los objetivos institucionales y estándares en materia de investigación. Finalmente, no asumo (asumimos) responsabilidad por la posible vulneración de derechos de autor en el trabajo final referido, pues tal responsabilidad es exclusiva del autor.

Fecha: 16/07/2025.

.....
*Firma del asesor*[1]

.....
*Firma del autor*[1]

.....
*Firma del co-asesor*[1]



**Dedicatoria**

A quienes caminaron a nuestro lado en este proceso.

**Agradecimientos**

A todos los que, de una u otra forma, formaron parte de este camino.

**Resumen**

En la era digital, la gestión del conocimiento es esencial para optimizar procesos y fomentar la innovación. La asignatura de Proyectos del programa académico de Ingeniería Industrial y de Sistemas de la Universidad de Piura enfrenta un problema: la falta de un sistema centralizado y accesible para gestionar las lecciones aprendidas generadas por los proyectos académicos. Esta ausencia provoca la pérdida de conocimiento, duplicación de esfuerzos y dificulta la mejora continua. El objetivo de esta tesis es diseñar, desarrollar e implementar un repositorio digital que facilite la captura, almacenamiento y recuperación de estas lecciones, promoviendo el aprendizaje colaborativo y la optimización de la gestión del conocimiento.

Para abordar este problema, se utilizó una metodología de desarrollo en cascada. Esta metodología permitió una planificación y diseño detallados antes de la implementación, adaptándose a los requisitos específicos del proyecto. El proceso incluyó la identificación de necesidades de los usuarios, el diseño del modelo relacional de la base de datos y la creación de una interfaz intuitiva que facilite el acceso a las lecciones aprendidas. Además, se llevaron a cabo pruebas unitarias y de integración para validar el funcionamiento del sistema.

El repositorio digital se diseñó para ser una herramienta eficiente y de fácil uso para estudiantes y docentes. Su implementación contribuyó a evitar la pérdida de conocimiento, a reducir la duplicación de esfuerzos y a fomentar una cultura de mejora continua y colaboración. Aunque los resultados de su impacto a largo plazo serán evaluados posteriormente, las pruebas iniciales indican que el sistema mejora el acceso a la información y apoya eficazmente los procesos de aprendizaje.

En conclusión, la creación de este repositorio digital representa un avance significativo en la gestión del conocimiento dentro de la Universidad de Piura. No solo facilita el intercambio de información, sino que también contribuye al desarrollo académico y profesional de los estudiantes. Este trabajo sienta las bases para futuras investigaciones y mejoras en la gestión del conocimiento en el ámbito educativo.

**Tabla de contenido**

**Lista de tablas**

## Lista de figuras

## Introducción

En la era actual, la información se ha convertido en un recurso estratégico, impulsando la evolución de la gestión del conocimiento desde métodos tradicionales hacia la digitalización y el intercambio dinámico de información. Brynjolfsson y McAfee (2014) destacan el papel crítico de la gestión del conocimiento en la era digital, ya que las tecnologías innovadoras están transformando el trabajo y la prosperidad. Gleick (2011) ofrece una perspectiva histórica sobre el desarrollo de la información, resaltando la importancia de su gestión eficiente en un mundo inundado de datos. Wallace (2007) refuerza esta idea al mostrar cómo los avances tecnológicos y pedagógicos han cambiado el enfoque hacia la maximización del conocimiento colectivo.

La gestión del conocimiento se centra principalmente en la administración de recursos humanos dentro de las organizaciones, ya que son los individuos quienes generan y aplican el conocimiento, más allá de la estructura organizacional en sí. Fomentar una cultura basada en el conocimiento se traduce en mejoras en la resolución de problemas, eficiencia en procesos y capacidad de adaptación a cambios del mercado (Whatfix, 2024). Según una encuesta realizada por Deloitte en 2020, el 75% de las empresas consideraron que compartir conocimientos era "importante o muy importante" para su éxito en los próximos 12 a 18 meses; sin embargo, solo un 9% se sentía preparado para implementar procesos de compartición de conocimiento por su cuenta (Whatfix, 2024). Además, una cultura robusta de conocimiento no solo promueve la innovación y mejora continua de productos y servicios, sino que también aumenta la satisfacción laboral, sustentando el crecimiento a largo plazo y la competitividad en mercados dinámicos. La inversión en herramientas y sistemas que promuevan esta cultura no es solo una estrategia para el éxito inmediato, sino también una apuesta por el futuro sostenible de la organización.

Dentro de este marco, las lecciones aprendidas (LA) emergen como activos esenciales que catalizan la mejora continua y la eficacia operativa, integrando conocimientos de experiencias pasadas para fortalecer la toma de decisiones futuras. La implementación consciente de la gestión de las LA facilita la evitación de errores previos y optimiza prácticas exitosas, mejorando la gestión de recursos y enriqueciendo el acervo de conocimientos institucional, esencial para futuros proyectos. Jugdev (2012) enfatiza la necesidad de un firme apoyo gerencial y una efectiva compartición de conocimiento para la gestión de las LA, destacando su importancia en el enriquecimiento de la base de conocimientos para futuras iniciativas. Este estudio subraya que “cuando fallamos en aprender de nuestros propios errores o los de otros, tendemos a repetir esos errores” (Jugdev, 2012, p. 13, traducción propia), reiterando la relevancia de estos activos en la construcción de proyectos futuros.

La sistematización del conocimiento es crucial para el aprendizaje organizacional y la innovación, como indican Turner, Keegan y Crawford (2000). Su estudio resalta cómo las organizaciones basadas en proyectos mejoran la gestión y el desarrollo profesional mediante

la captura y análisis efectivo de lecciones aprendidas. Este proceso no solo previene la repetición de errores, sino que también fomenta un ambiente de aprendizaje colaborativo, esencial para el crecimiento sostenible y la competitividad.

A pesar de estos beneficios, las organizaciones enfrentan desafíos significativos al implementar sistemas de gestión del conocimiento, incluyendo la resistencia al cambio, la necesidad de capacitación continua y los retos tecnológicos. Superar estos obstáculos es fundamental para maximizar el potencial de estas estrategias, fomentando un entorno de aprendizaje más dinámico y colaborativo.

En el contexto de la Universidad de Piura, los trabajos de Alvarado (2015) y Guerrero Chanduví et al. (2023) destacan los avances significativos en la incorporación de la gestión del conocimiento en la educación superior. Alvarado se centra en el desarrollo e implementación de un repositorio digital, subrayando cómo este recurso potencia el acceso y la diseminación del conocimiento académico. De manera similar, Guerrero Chanduví et al. examinan la adopción de un modelo de gestión del conocimiento que busca optimizar los procesos educativos y de investigación. Ambos estudios recalcan la necesidad de una infraestructura bien definida que soporte la documentación eficaz y el intercambio de información, fundamentales para estimular el aprendizaje colaborativo y fomentar una cultura de mejora e innovación continua.

La adopción de plataformas de gestión del conocimiento en la educación superior mejora la eficiencia, productividad y colaboración en instituciones académicas (Bloomfire, 2021). Estas herramientas no solo facilitan un enfoque colaborativo hacia el aprendizaje, permitiendo la creación y el intercambio activo de conocimiento entre estudiantes y profesores, sino que también enriquecen la experiencia educativa al promover una interacción significativa con el material de estudio. Al preparar a los estudiantes con habilidades críticas para los desafíos futuros, estas plataformas ofrecen una ventaja competitiva sustancial.

Esta tesis es la continuación de un estudio previo titulado "Diseño de repositorio digital de lecciones aprendidas como herramienta de gestión del conocimiento: caso de estudio asignatura de proyectos” (Guerrero Chanduví et al., 2023). En la primera parte, se realizó una revisión exhaustiva de 158 proyectos de la asignatura de proyectos en la Universidad de Piura, generando un modelo de atributos y clasificaciones de las lecciones aprendidas para su almacenamiento y posterior búsqueda. En contraste, la presente segunda parte se centra en el diseño detallado, desarrollo e implementación del repositorio digital, abarcando la especificación de requisitos, desarrollo del modelo relacional, programación y puesta en marcha del sistema.

# Capítulo 1

# Estado del Arte

## 1.1 Planteamiento del problema

La presente investigación continúa el trabajo desarrollado por Jaramillo y Cabellos (2024), quienes identificaron una problemática crítica en la asignatura de Proyectos de la Universidad de Piura: la falta de un sistema estructurado para almacenar y reutilizar las lecciones aprendidas generadas por los estudiantes. Esta asignatura, oficialmente denominada "Curso de Trabajo de Investigación: Dirección de Proyectos" desde el plan de estudios 2024, se dicta tanto en el campus Piura como Lima, con la misma estructura académica y enfoque metodológico.

Según Jaramillo y Cabellos (2024), "la asignatura de Proyectos [...] tiene como propósito desarrollar competencias en la dirección de proyectos bajo los estándares internacionales (IPMA y PMI)" (p. 11). A lo largo del ciclo, "los estudiantes deben de realizar un proyecto final en equipos [...] donde se pongan en práctica y se utilicen todas las herramientas enseñadas en el curso" (p. 22). Parte esencial de este proceso es la documentación de lecciones aprendidas (LA), que son incluidas en los informes semanales.

Sin embargo, como indican las autoras, "no se cuenta con un sistema para que estudiantes futuros puedan tener acceso a las LA de alumnos que ya cursaron la asignatura y realizaron proyectos en distintos rubros" (p. 11). Esto ha generado una pérdida de conocimiento valioso, dificultando la mejora continua y el aprendizaje colectivo.

Frente a esta situación, esta tesis desarrolla e implementa un repositorio digital funcional para la sistematización, búsqueda y reutilización de lecciones aprendidas, contribuyendo al fortalecimiento del proceso formativo en la asignatura. Si bien el curso se dicta en ambos campus de la Universidad de Piura (Piura y Lima), la presente solución ha sido diseñada e implementada exclusivamente para el campus Piura, en atención a un pedido específico del profesor responsable de la asignatura en dicha sede, donde se identificó originalmente la problemática según lo reportado por Guerrero Chanduví et al. (2023).

Este problema impacta en multiples dimensiones el proceso formativo:

- Pérdida de conocimiento: Sin un repositorio centralizado, el conocimiento generado se dispersa y se pierde, impidiendo su reutilización en futuros proyectos. Jugdev (2012) subraya la importancia de gestionar eficazmente las lecciones aprendidas para evitar la repetición de errores y optimizar prácticas exitosas.
- Duplicación de esfuerzos: La falta de un sistema organizado lleva a que estudiantes y docentes deban redescubrir soluciones y metodologías previamente desarrolladas. Este problema también se observa en organizaciones reales: Turner et al. (2000) indican que muchas organizaciones repiten los mismos errores al no

capturar ni compartir adecuadamente el conocimiento generado en proyectos anteriores (traducción propia, p. 3).

- Curva de aprendizaje empinada: En ausencia de un repositorio estructurado, es razonable suponer que los nuevos estudiantes podrían enfrentar una curva de aprendizaje más empinada, al no contar con acceso sistemático a las experiencias previas de otros equipos. Bloomfire (2021) señala que una plataforma de gestión del conocimiento puede mejorar la capacitación desde el primer día y facilitar el aprendizaje organizacional en contextos educativos.
- Falta de mejora continua: La ausencia de un mecanismo para documentar y compartir lecciones aprendidas impide la implementación de mejoras continuas en procesos y prácticas educativas. Turner, Keegan y Crawford (2000) señalan que la sistematización del conocimiento es crucial para el aprendizaje organizacional y la innovación.
- Ineficiencia en la gestión del conocimiento: Una gestión del conocimiento fragmentada reduce la eficacia operativa y la capacidad de la Universidad de Piura para aprovechar plenamente el conocimiento generado. Alvarado (2015) indica que un modelo de gestión del conocimiento puede mejorar significativamente la eficiencia operativa al centralizar la documentación y facilitar el acceso a conocimientos previos.

El establecimiento de un repositorio digital de lecciones aprendidas tiene el potencial de resolver estos problemas al proporcionar una plataforma centralizada y accesible para la captura, almacenamiento y recuperación de conocimientos. Este repositorio permitirá a los estudiantes y profesores acceder fácilmente a la información relevante, facilitando el aprendizaje colaborativo y la mejora continua. Además, fomentará una cultura de compartición de conocimientos y de optimización de prácticas basadas en experiencias previas, contribuyendo así a la innovación y eficiencia en los proyectos académicos.

## 1.2 Justificación

La implementación de un repositorio digital de lecciones aprendidas en la asignatura de proyectos de la Universidad de Piura es crucial para mejorar la gestión del conocimiento. Este sistema facilitará el acceso y la diseminación del conocimiento generado en los proyectos académicos, promoviendo una cultura de aprendizaje colaborativo y mejora continua.

### 1.2.1 *Beneficios del Repositorio Digital:*

- Acceso centralizado al conocimiento: Un repositorio digital permitirá que estudiantes y profesores accedan fácilmente a un banco centralizado de lecciones aprendidas, evitando la pérdida de información valiosa. Según Nupap (2022), un sistema de

gestión del conocimiento (KMS) facilita la creación, almacenamiento y acceso tanto al conocimiento tácito como explícito, asegurando su uso y sostenibilidad a largo plazo.

- Eficiencia y Efectividad: Al evitar la duplicación de esfuerzos y aprovechar soluciones ya desarrolladas, se mejora la eficiencia en la ejecución de nuevos proyectos. Thanachawengsakul et al. (2019) afirman que un sistema de repositorio digital de conocimientos aumenta la eficiencia de la gestión del conocimiento al permitir la verificación y el uso colaborativo mediante el aprendizaje automático.
- Mejora Continua: La documentación sistemática de lecciones aprendidas facilita la identificación de áreas de mejora y la implementación de cambios en tiempo real. Turner, Keegan y Crawford (2000) señalan que el aprendizaje experiencial fomenta la competencia y la mejora continua al capturar y analizar las lecciones aprendidas.
- Aprendizaje Colaborativo: La accesibilidad a lecciones aprendidas fomentará un ambiente de aprendizaje colaborativo, donde los estudiantes puedan aprender de las experiencias de otros. Bloomfire (2021) menciona que una plataforma de gestión del conocimiento mejora la capacitación y la colaboración al asegurar que toda la información relevante esté disponible para todos los miembros de la organización.
- Innovación y Creatividad: Al tener acceso a una base de conocimientos rica y diversa, los estudiantes y profesores estarán mejor equipados para desarrollar ideas innovadoras y soluciones creativas. Según el estudio de MacDowell (2023), el acceso a tecnologías inmersivas y repositorios de conocimiento promueve la creatividad y la innovación entre los jóvenes al proporcionarles las herramientas y el conocimiento necesarios para explorar nuevas ideas y soluciones

En resumen, el repositorio digital no solo será una herramienta tecnológica, sino también un catalizador para la transformación educativa, impulsando la excelencia académica y la innovación en la Universidad de Piura.

## 1.3 Objetivos

### 1.3.1 *Objetivo General*

Diseñar, desarrollar e implementar un repositorio digital en la Universidad de Piura, específicamente para la asignatura de proyectos del programa académico de Ingeniería Industrial y de Sistemas, con el fin de optimizar la gestión del conocimiento a través del reúso de lecciones aprendidas.

### 1.3.2 *Objetivos Específicos*

- Evaluar las necesidades y requisitos específicos de los usuarios potenciales del repositorio digital.
- Desarrollar un modelo relacional para la base de datos que soporte la gestión eficiente de lecciones aprendidas.
- Implementar una interfaz de usuario intuitiva que facilite la captura, almacenamiento y recuperación de lecciones aprendidas.
- Validar el desempeño del sistema a través de pruebas unitarias y de integración.

# Capítulo 2

# Marco Teórico

En la educación superior, la gestión del conocimiento es esencial para el desarrollo académico y profesional de los estudiantes. La creación y reutilización del conocimiento, especialmente a través de herramientas digitales, se ha convertido en un aspecto crítico. Según Nonaka y Takeuchi (1995), el conocimiento se genera mediante un proceso continuo de conversión entre conocimiento tácito y explícito, que las plataformas digitales facilitan. Davenport y Prusak (2000) afirman que las organizaciones deben implementar mecanismos que permitan identificar, compartir y aplicar el conocimiento disponible en sus equipos, procesos y rutinas. Aunque su enfoque se centra en entornos corporativos, estos principios también pueden aplicarse a instituciones académicas, donde el conocimiento generado en cursos y proyectos puede gestionarse estratégicamente mediante repositorios digitales.

La sistematización del conocimiento en las instituciones académicas también se relaciona con la mejora continua. Turner, Keegan y Crawford (2000) señalan que la captura y análisis efectivo de las lecciones aprendidas en las organizaciones fomenta un ambiente de aprendizaje colaborativo, necesario para el crecimiento sostenible y la competitividad. Por su parte, Bloomfire (2021) menciona que una plataforma de gestión del conocimiento no solo mejora la capacitación, sino que también enriquece la experiencia educativa al promover la colaboración y la interacción significativa con el material de estudio.

Es por ello que la presente tesis se orienta al desarrollo de un repositorio digital que facilite la gestión del conocimiento en la asignatura de Proyectos. En este sentido, en este capítulo se abordarán los fundamentos teóricos que sustentan el proyecto, proporcionando un marco conceptual que permita comprender la importancia y los beneficios de la implementación de este tipo de soluciones en el ámbito académico.

## 2.1 Desarrollo de Software

El desarrollo de software es un proceso estructurado que abarca diversas actividades organizadas en fases, desde el análisis de requerimientos hasta el mantenimiento del sistema. Estas fases conforman el Ciclo de Vida del Desarrollo de Software (SDLC), cuya implementación efectiva permite gestionar la complejidad y asegurar la calidad del producto final (Sommerville, 2016)

Existen varios enfoques para aplicar el SDLC, siendo los más relevantes para este estudio los siguientes:

- Modelo en cascada: Propone un enfoque secuencial en el que cada fase (análisis, diseño, implementación, pruebas, despliegue y mantenimiento) debe completarse

antes de avanzar a la siguiente. Es adecuado cuando los requerimientos están bien definidos desde el inicio y hay poca tolerancia al cambio (Sommerville, 2016)

- Modelo iterativo e incremental: En este modelo, el software se desarrolla a través de ciclos repetitivos (iteraciones), donde cada versión incluye incrementos de funcionalidad. Esto permite construir versiones parciales del sistema que evolucionan con base en retroalimentación temprana. (Sommerville, 2016) señala que este modelo ofrece mayor flexibilidad ante cambios y mejora la validación progresiva del producto (p. 52).

- Metodologías ágiles: Nacidas como alternativa a los modelos tradicionales, las metodologías ágiles priorizan la colaboración constante, la entrega continua de valor y la adaptación al cambio. El Agile Manifesto establece valores fundamentales como la interacción entre individuos, el software funcionando, la colaboración con el cliente y la respuesta al cambio (Beck et al., 2001). Entre los enfoques más conocidos destacan Scrum, que estructura el trabajo en sprints, y Extreme Programming (XP), que promueve prácticas como pruebas automatizadas y desarrollo iterativo (Sommerville, 2016, pp. 61-63).

En este capítulo se explorarán dos enfoques relevantes: el modelo en cascada, seleccionado para este proyecto debido a la estabilidad de los requisitos, y las metodologías ágiles, cuyo uso parcial ayudó a optimizar el desarrollo.

### 2.1.1 *Modelo de desarrollo en cascada*

El modelo de desarrollo en cascada es una metodología secuencial que estructura el proceso de desarrollo de software en fases lineales: análisis de requisitos, diseño, implementación, verificación y mantenimiento. Cada etapa debe completarse formalmente antes de avanzar a la siguiente, lo que facilita una planificación anticipada y un control riguroso del proceso (Sommerville, 2016).

Este modelo ha sido tradicionalmente utilizado en proyectos donde los requisitos están claramente definidos desde el inicio y se espera poca variabilidad durante el desarrollo. En estos casos, su estructura ordenada permite una documentación completa, una trazabilidad clara y un seguimiento efectivo del avance del proyecto (Sommerville, 2016).

Sin embargo, su uso también ha sido cuestionado en entornos más dinámicos. Meyer (2014) sostiene que el modelo en cascada se mantiene vigente en la enseñanza más como referencia histórica que como una práctica actual, y lo considera superado frente a metodologías más iterativas. En respuesta a estas limitaciones, surgieron enfoques como las metodologías ágiles, que valoran la interacción entre individuos, el software funcionando, la colaboración con el cliente y la respuesta ante el cambio por encima de los procesos rígidos y la planificación estricta (Beck et al., 2001).

A pesar de ello, investigaciones recientes muestran que el modelo en cascada sigue siendo adecuado en contextos específicos. Saravanos y Curinga (2023) concluyen, a partir de simulaciones del ciclo de vida del software, que este enfoque resulta efectivo en proyectos con requisitos estables y equipos de tamaño reducido. En estos casos, permite una estimación más precisa de tiempos y recursos, además de reducir la necesidad de coordinación compleja, lo que lo convierte en una opción viable para desarrolladores individuales.

Dado que el presente proyecto fue desarrollado por un equipo pequeño y los requerimientos estaban claramente definidos desde el inicio, se optó por el modelo en cascada como marco principal. Para compensar algunas de sus limitaciones, se integraron prácticas puntuales propias de las metodologías ágiles —como revisiones frecuentes y ajustes iterativos— que mejoraron la eficiencia sin alterar la estructura secuencial del modelo.

### 2.1.2 *Metodologías Ágiles: Scrum*

Aunque el presente proyecto adoptó inicialmente el modelo en cascada, se integraron prácticas propias de metodologías ágiles para optimizar el proceso de mejora continua del producto. Las metodologías ágiles surgieron como una respuesta a las limitaciones de los enfoques tradicionales, promoviendo valores orientados a la flexibilidad, la colaboración y el enfoque en resultados. Como establece el Manifiesto por el Desarrollo Ágil de Software, los autores “valoran más la colaboración con el cliente sobre la negociación contractual” y “la respuesta ante el cambio sobre seguir un plan” (Beck et al., 2001).

Una de las metodologías ágiles más ampliamente utilizadas es Scrum, un marco de trabajo que organiza el desarrollo en iteraciones cortas llamadas sprints. En cada sprint, se define un conjunto de tareas priorizadas, se avanza de forma incremental y se realiza una revisión del progreso al final del ciclo. Según Hossain (2023), Scrum permite gestionar el cambio de manera efectiva mediante la entrega continua de valor y la retroalimentación frecuente.

En este proyecto, se utilizó el modelo en cascada para el desarrollo del producto mínimo viable (MVP), basado en requerimientos previamente definidos. Una vez completado el MVP, se llevaron a cabo sesiones de retroalimentación con docentes de la asignatura y con los asesores del proyecto de tesis. Estas sesiones permitieron identificar ajustes y mejoras necesarias en la estructura, contenido y funcionalidad del repositorio.

A partir de estas observaciones, se adoptó una dinámica de trabajo inspirada en Scrum, estructurando el desarrollo en ciclos breves orientados a objetivos concretos. Esta forma de organización permitió priorizar funcionalidades, implementar cambios de manera incremental y evaluar continuamente los avances con los asesores.

Las principales prácticas ágiles integradas fueron:

- Planificación iterativa basada en las observaciones recibidas tras el MVP
- Desarrollo incremental por bloques de funcionalidad
- Revisiones periódicas con los asesores para validar avances y definir próximos pasos
- Flexibilidad en la implementación para adaptar mejoras sin alterar la arquitectura base

Estas prácticas contribuyeron a mantener un proceso de desarrollo eficiente y adaptativo, reforzando la calidad del producto final sin comprometer la estructura metodológica definida inicialmente.

## 2.2 Tecnologías Utilizadas

Para la implementación del repositorio digital, se seleccionaron tecnologías que garantizan eficiencia, escalabilidad y mantenibilidad. En este capítulo se presentan los conceptos generales sobre *framework*, lenguajes de programación y arquitecturas de software, mientras que en el siguiente capítulo se detallarán las tecnologías específicas utilizadas en el desarrollo del proyecto.

### 2.2.1 *Framework de Desarrollo*

Un *framework* de desarrollo es un conjunto integrado de artefactos de software, como clases, objetos y componentes, que colaboran para proporcionar una arquitectura reutilizable para una familia de aplicaciones relacionadas (Sommerville, 2016). Estos marcos de trabajo promueven el uso de buenas prácticas de programación, favorecen la reutilización del código y proporcionan una estructura prediseñada que facilita tanto la construcción como el mantenimiento de aplicaciones.

En el desarrollo web, existen diversos *framework* ampliamente utilizados, entre los que destacan Ruby on Rails, Django y Laravel, cada uno con enfoques, lenguajes y características particulares.

Ruby on Rails, desarrollado en el lenguaje Ruby, es un *framework* de código abierto que implementa el patrón arquitectónico Modelo-Vista-Controlador (MVC). Este patrón organiza la estructura de una aplicación en tres componentes principales: el modelo, encargado de representar los datos y la lógica de negocio; la vista, responsable de la presentación visual de los datos al usuario; y el controlador, que gestiona las solicitudes del usuario, procesa la lógica correspondiente y selecciona qué vista mostrar. Esta separación de responsabilidades facilita la mantenibilidad del código, la reutilización de componentes y una mayor claridad en la organización del sistema (Sommerville, 2016).

Rails se distingue, además, por su enfoque en la productividad del desarrollador, apoyado en principios como convención sobre configuración, que reduce la necesidad de decisiones explícitas en la configuración del entorno, y Don't Repeat Yourself (DRY), que promueve la reutilización del código y la eliminación de redundancias. El *framework* incluye herramientas que automatizan tareas comunes, como la generación de formularios, migraciones de base de datos y validaciones, lo que permite un desarrollo más rápido y estructurado. Asimismo, cuenta con un ecosistema maduro y una comunidad activa que ofrece una amplia variedad de bibliotecas reutilizables, contribuyendo a su versatilidad y escalabilidad (Railsware, 2023).

Django, por su parte, es un *framework* web gratuito y de código abierto escrito en Python, diseñado para facilitar el desarrollo rápido de aplicaciones seguras, escalables y con diseño limpio. Proporciona funcionalidades integradas para el manejo de bases de datos, autenticación de usuarios, administración de contenido, validación de formularios y generación dinámica de HTML mediante su motor de plantillas. Uno de sus mayores atributos es la seguridad: ofrece protección automática contra ataques comunes como inyecciones SQL, XSS y CSRF. Además, su robustez y escalabilidad han sido comprobadas en aplicaciones de alto tráfico como Instagram o Disqus (IBM, 2021).

Laravel, desarrollado en PHP, es un *framework* moderno que busca proporcionar una experiencia de desarrollo sencilla y expresiva. También implementa el patrón MVC, incluye un sistema de plantillas propio llamado Blade, y ofrece herramientas integradas para autenticación, enrutamiento, migraciones, pruebas automatizadas y más. Su ecosistema incluye servicios adicionales como Laravel Forge, Vapor y Nova, orientados a facilitar el despliegue, el escalamiento y la administración de aplicaciones modernas (Laravel, 2025). Laravel se presenta como una solución progresiva y flexible, adecuada para proyectos de distintos niveles de complejidad dentro del ecosistema PHP.

En el contexto del presente proyecto, la elección de Ruby on Rails respondió tanto a criterios técnicos como estratégicos. Desde una perspectiva técnica, Rails permite acelerar significativamente el desarrollo gracias a su estructura clara, su enfoque en la automatización de tareas repetitivas y su arquitectura coherente (Railsware, 2023). Además, el equipo de desarrollo ya contaba con experiencia previa en Ruby, lo que permitió reducir la curva de aprendizaje y avanzar con mayor eficiencia. A nivel práctico, el amplio ecosistema de Rails y la disponibilidad de bibliotecas facilitaron la implementación de funcionalidades clave sin partir desde cero. Finalmente, su compatibilidad con el patrón MVC y su orientación a buenas prácticas de desarrollo permitieron estructurar el repositorio digital de forma ordenada, escalable y fácil de mantener.

Por estas razones, Ruby on Rails fue considerado la opción más adecuada para el desarrollo del sistema propuesto, permitiendo cumplir con los objetivos del proyecto de manera eficiente, estructurada y sostenible.

### 2.2.2 Lenguajes de Programación

Los lenguajes de programación permiten expresar instrucciones que una computadora puede interpretar y ejecutar. Según Sommerville (2016), estos pueden clasificarse en lenguajes de bajo nivel, que ofrecen control detallado sobre el hardware y requieren conocimientos técnicos específicos, y lenguajes de alto nivel, que son más legibles y permiten desarrollar software de manera más eficiente al abstraer la complejidad del sistema.

En el desarrollo web moderno, se distinguen dos componentes esenciales: el *backend*, encargado de gestionar la lógica de negocio, la base de datos y la interacción con el servidor, y el *frontend*, que se ocupa de la interfaz visual y la experiencia del usuario. El *backend* suele desarrollarse con lenguajes como Python, Ruby o PHP, mientras que el *frontend* se construye con tecnologías como HTML, CSS y JavaScript (Deshmukh et al, 2025).

Para este proyecto se eligió Ruby como lenguaje principal del *backend*, aprovechando su integración con Ruby on Rails para construir una aplicación estructurada, segura y eficiente. Esta elección fue detallada previamente en el capítulo 2.2.1, donde se explicó su alineación con los objetivos del proyecto y la experiencia del equipo de desarrollo.

En el *frontend* se utilizó JavaScript para añadir interactividad y dinamismo a la interfaz, permitiendo gestionar eventos del usuario y actualizar el contenido sin recargar la página. Complementariamente, HTML se utilizó para estructurar el contenido de cada vista, y CSS para definir su presentación visual. Aunque HTML y CSS no son lenguajes de programación, son fundamentales en el desarrollo web moderno, ya que permiten una separación clara entre contenido y estilo, mejorando la organización del código y facilitando su mantenimiento (Jain et al., 2024). Esta combinación de tecnologías permite crear interfaces accesibles, adaptables y coherentes en distintos dispositivos.

### 2.2.3 Arquitecturas de *Software*

La arquitectura de software describe la estructura general de un sistema y define cómo interactúan sus componentes fundamentales. De acuerdo con Sommerville (2016), una arquitectura de software especifica las principales decisiones estructurales, estableciendo los módulos de un sistema, sus responsabilidades y las relaciones entre ellos. A continuación, se presentan tres de las arquitecturas más comunes:

#### 2.2.3.1 Arquitectura Monolítica:

Consiste en una única aplicación en la que todos los módulos están integrados y ejecutándose dentro del mismo proceso. Aunque puede ser más sencilla de implementar inicialmente, su mantenimiento se complica a medida que el sistema crece, debido a la alta

interdependencia de sus componentes. Dragoni et al. (2017) definen un monolito como “una aplicación de software cuyos módulos no pueden ejecutarse de manera independiente” (p. 196). Entre sus principales limitaciones se encuentran la dificultad para escalar partes específicas del sistema, los tiempos prolongados de despliegue, y la dependencia de un solo *stack* tecnológico.

#### 2.2.3.2 Arquitectura de Microservicios:

Este estilo divide el sistema en pequeños servicios autónomos, cada uno encargado de una funcionalidad específica y comunicándose entre sí mediante mensajes o APIs. Esta arquitectura permite una mayor flexibilidad tecnológica, escalabilidad por componente y despliegues independientes. Al-Debagy y Martinek (2018) destacan ventajas como la tolerancia a fallos, la facilidad de mantenimiento y la alineación entre equipos de desarrollo y componentes de software. Sin embargo, también se enfrentan a desafíos como la complejidad en la orquestación y el mayor uso de recursos debido a la comunicación distribuida.

#### 2.2.3.3 Arquitectura Cliente-Servidor:

Este modelo divide el sistema en dos partes principales: el cliente, que es la interfaz de usuario, y el servidor, que gestiona la lógica de negocio y el acceso a los datos. Este tipo de arquitectura es la base de muchas aplicaciones web modernas, donde el *frontend* (cliente) interactúa con el *backend* (servidor) a través de HTTP u otros protocolos de red. Sommerville (2016) señala que esta separación permite una mayor reutilización de componentes del servidor y una independencia relativa del cliente, aunque también puede generar problemas de latencia si no se gestiona adecuadamente la comunicación.

En el siguiente capítulo se detallará qué tipo de arquitectura se empleó en la implementación del repositorio digital, justificando su elección en función del contexto y los requerimientos del proyecto.

### 2.2.4 Modelo entidad-relación*:*

El modelo entidad-relación (DER) es una herramienta fundamental en el desarrollo de sistemas de información, ya que permite representar de manera abstracta y comprensible los elementos que conforman una base de datos y cómo se relacionan entre sí. Esta representación conceptual facilita el análisis, diseño y comunicación entre los diferentes actores involucrados en el desarrollo del sistema.

En el contexto de sistemas como repositorios digitales, el DER resulta especialmente útil para garantizar una estructura lógica de la información que permita su adecuada organización, búsqueda y reutilización. Este modelo, al enfocarse en las entidades y sus relaciones, permite identificar con claridad los datos relevantes, sus atributos, y cómo interactúan dentro del sistema. Esto contribuye a mantener la integridad, consistencia y escalabilidad del sistema.

Desde una perspectiva más amplia de modelado conceptual, Al-Fedaghi & Modhaffar (2021) destacan que los modelos abstractos, como los diagramas entidad-relación, son esenciales en la representación inicial de sistemas complejos, ya que actúan como “planos conceptuales” que conectan los requisitos del sistema con su posterior implementación. Este tipo de modelos permiten una visualización clara de las relaciones estructurales entre los componentes del sistema, facilitando tanto su análisis como su documentación.

En esta investigación, el uso del modelo entidad-relación responde a la necesidad de estructurar la información de manera lógica para capturar las lecciones aprendidas registradas por los estudiantes. Esta estructuración no solo permite una mejor categorización y acceso, sino que también favorece la sostenibilidad del conocimiento acumulado a lo largo del tiempo.

## 2.3 Importancia de los Repositorios Digitales

La implementación de repositorios digitales en la educación superior contribuye significativamente a mejorar la eficiencia, la productividad y la colaboración institucional, al facilitar la organización, acceso y reutilización del conocimiento generado en actividades académicas (Alvarado, 2015). En el ámbito universitario, los proyectos juegan un papel fundamental en la formación de los estudiantes, ya que permiten aplicar conocimientos teóricos a problemas reales. Sin embargo, cuando la experiencia y el conocimiento adquiridos durante estos proyectos no se documentan de manera sistemática, se produce una pérdida valiosa de información (Jugdev, 2012; Wallace, 2007).

Para evitar esta situación, es esencial gestionar las lecciones aprendidas, entendidas como el conjunto de conocimientos obtenidos a lo largo del desarrollo de un proyecto. Estas pueden incluir estrategias exitosas, errores detectados, dificultades superadas y soluciones implementadas, las cuales permiten optimizar procesos futuros y prevenir la repetición de fallos (Jugdev, 2012).

Alvarado (2015) destaca que un modelo de gestión del conocimiento puede mejorar significativamente la eficiencia operativa en instituciones educativas, al centralizar la documentación y facilitar el acceso a conocimientos previos. Una de las herramientas más utilizadas para este fin es el Sistema de Gestión del Conocimiento (KMS, por sus siglas en inglés), que permite crear, almacenar, compartir y reutilizar tanto el conocimiento tácito como el explícito (Nupap, 2022).

De acuerdo con Nupap (2022), un KMS se compone de cuatro elementos principales: infraestructura tecnológica, procesos organizacionales, contenido de conocimiento y usuarios. Para su desarrollo, se aplican modelos conceptuales como el SECI de Nonaka y Takeuchi (1995), que describe el proceso de conversión del conocimiento en cuatro fases: socialización, externalización, combinación e internalización. Este modelo permite transformar el conocimiento tácito individual en conocimiento explícito accesible, y luego

reconvertirlo en conocimiento tácito colectivo, facilitando así su sostenibilidad dentro de la organización.

Además, plataformas como los repositorios digitales refuerzan el aprendizaje colaborativo y preparan a los estudiantes con habilidades clave para entornos profesionales cada vez más exigentes (Bloomfire, 2021). La sistematización de las lecciones aprendidas no solo evita la pérdida de conocimiento, sino que también reduce la duplicación de esfuerzos, promueve el aprendizaje organizacional y fomenta una cultura de mejora continua (Gleick, 2011; Wallace, 2007).

La adopción de estas soluciones tecnológicas en el entorno universitario, como el repositorio digital propuesto en esta tesis, responde a las tendencias actuales en educación superior que buscan integrar innovación, sostenibilidad y competitividad en sus procesos formativos.

# Capítulo 3

# Diseño

## 3.1 Elección de tecnología

Para el desarrollo del repositorio digital de lecciones aprendidas, se seleccionaron tecnologías que garantizan un desarrollo eficiente, escalable y mantenible. Las decisiones tecnológicas se basaron en la capacidad de estas herramientas para satisfacer los requisitos del proyecto, incluyendo rendimiento, seguridad, facilidad de uso y soporte comunitario.

### 3.1.1 *Framework y Lenguaje de Programación*

Para el desarrollo del sistema, se optó por Ruby on Rails, un *framework* conocido por su rapidez y eficiencia en la creación de aplicaciones web. Su enfoque en la productividad y la simplicidad permite a los desarrolladores construir aplicaciones robustas en menos tiempo.

Rails se destaca por su consistencia, el respaldo de una comunidad de desarrollo activa y una amplia disponibilidad de librerías reutilizables, lo que lo convierte en una alternativa sólida para el desarrollo sostenible de aplicaciones web complejas (SoluteLabs, 2023). Además, su arquitectura basada en el patrón Modelo-Vista-Controlador (MVC) facilita una clara separación de responsabilidades: el modelo gestiona la lógica de negocio y los datos, la vista se encarga de la presentación visual y el controlador coordina la interacción entre ambos. Esta organización mejora significativamente la mantenibilidad del código y su escalabilidad a largo plazo.

### 3.1.2 *Frontend (Interfaz de Usuario)*

Para la interfaz de usuario, se utilizaron HTML, CSS y JavaScript, tecnologías estándar en el desarrollo web. HTML define la estructura del contenido, CSS permite personalizar su apariencia y JavaScript agrega interactividad para mejorar la experiencia del usuario. Estas tecnologías fueron elegidas por su compatibilidad con todos los navegadores y su capacidad para crear interfaces intuitivas y visualmente atractivas (Shvachych, Aleksieiev, Havrylov, & Mamuzić, 2024).

### 3.1.3 *Base de Datos*

Como sistema de gestión de bases de datos se seleccionó MySQL, una tecnología ampliamente utilizada en entornos web por su facilidad de integración, estabilidad y compatibilidad con múltiples lenguajes y frameworks. Su integración nativa con Ruby on Rails, su bajo costo operativo y su amplia documentación la convierten en una opción adecuada para proyectos de pequeña a mediana escala como el presente repositorio digital.

Además, estudios recientes han demostrado que, mediante un diseño adecuado —incluyendo estrategias como la normalización, la indexación y el particionamiento— MySQL puede alcanzar niveles óptimos de rendimiento y escalabilidad en aplicaciones exigentes (Yadav et

al. 2024). Estas características permiten que MySQL maneje eficientemente datos estructurados y consultas complejas, manteniendo la coherencia y la eficiencia operativa del sistema.

### 3.1.4 *Servidor Web*

Para la gestión del tráfico web y la entrega de contenido, se consideraron Apache y Nginx, dos de los servidores web más utilizados en la industria. Apache destaca por su flexibilidad y amplia compatibilidad con diferentes tecnologías, mientras que Nginx es reconocido por su alto rendimiento y eficiencia en la gestión de múltiples conexiones simultáneas. La combinación de ambos servidores permite optimizar el rendimiento del sistema, garantizando tiempos de respuesta rápidos y una experiencia de usuario fluida (Tiwari, 2023).

Además, Apache cuenta con una arquitectura modular que permite habilitar o deshabilitar funcionalidades de forma personalizada, lo cual incrementa su flexibilidad y escalabilidad. Esta estructura también favorece la resiliencia operativa, ya que cada módulo funciona de manera independiente, asegurando que el sistema siga funcionando incluso ante fallos puntuales (Chaw, 2023). Estas características lo convierten en una opción sólida para entornos educativos y de investigación, donde la estabilidad, trazabilidad y facilidad de mantenimiento son prioritarias.

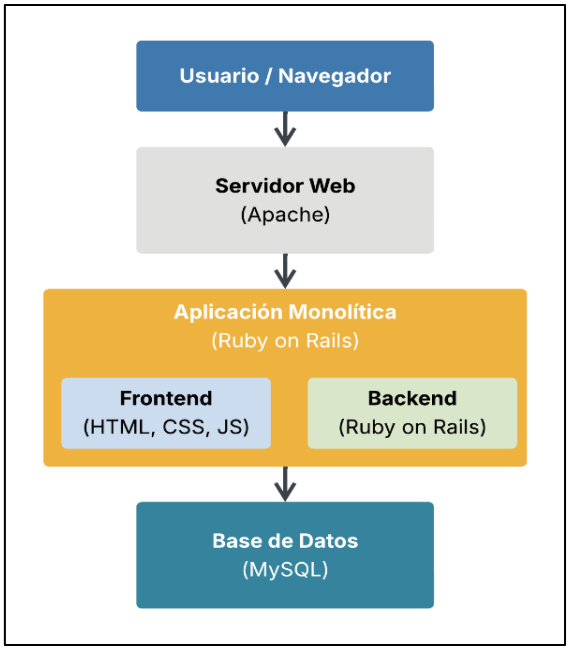
## 3.2 Arquitectura de Software

La arquitectura del repositorio digital sigue un modelo monolítico, en el cual todas las funcionalidades de la aplicación están integradas en un solo código base y se despliegan como una única unidad. Esta estructura facilita el desarrollo y mantenimiento, ya que todos los componentes residen en una misma aplicación y no requieren una comunicación distribuida entre servicios.

La figura 1 presenta un diagrama de alto nivel que ilustra la arquitectura del sistema. En él se muestra cómo un usuario accede al repositorio digital a través del navegador, el cual envía las peticiones al servidor web (Apache). Este, a su vez, comunica las solicitudes a una aplicación monolítica desarrollada con Ruby on Rails, que gestiona tanto el frontend (HTML, CSS, JavaScript) como el backend (lógica y procesamiento de datos). Finalmente, la información se almacena o recupera desde una base de datos MySQL.

**Figura 1**

*Diagrama de Arquitectura de Software*




*Nota*. Diagrama de la arquitectura de software mostrando el modelo monolítico y cómo interactúan los componentes principales (*Frontend*, *Backend*, Base de Datos, Servidor Web).

El uso de una arquitectura monolítica en este contexto presenta diversas ventajas. Según Al-Debagy y Martinek (2018), este tipo de arquitectura mostró mejor rendimiento en escenarios de concurrencia, con un 6% más de rendimiento en throughput respecto a microservicios. Además, su simplicidad facilita la implementación, el despliegue y la trazabilidad del sistema, lo que la hace idónea para proyectos de escala moderada como el presente.

Sin embargo, también existen desventajas. Las aplicaciones monolíticas pueden volverse difíciles de escalar de forma granular, ya que cualquier modificación o actualización requiere el despliegue completo de la aplicación. Asimismo, el acoplamiento de componentes puede dificultar la adopción de nuevas tecnologías o la reutilización de partes específicas del sistema.

A pesar de estas limitaciones, la elección de una arquitectura monolítica fue estratégica: se priorizó una implementación rápida, confiable y coherente con los recursos y necesidades del proyecto. Ruby on Rails se integra de manera natural con este enfoque arquitectónico, ya que fue diseñado originalmente como un framework monolítico, proporcionando herramientas integradas para el enrutamiento, controladores, modelos y

vistas en un solo entorno. Asimismo, MySQL se acopla eficientemente como sistema de almacenamiento en este tipo de arquitectura, facilitando su configuración y mantenimiento sin requerir infraestructura distribuida adicional.

Por tanto, la combinación de Ruby on Rails con una base de datos MySQL dentro de una arquitectura monolítica permite una solución coherente, funcional y alineada con los objetivos del proyecto, optimizando el uso de recursos y simplificando el proceso de desarrollo y despliegue.

## 3.3 Diseño de Perfiles de Usuario

El diseño de perfiles de usuario es un componente fundamental en el desarrollo de sistemas, ya que permite establecer diferentes niveles de acceso y funcionalidades, garantizando que cada tipo de usuario interactúe con el sistema de manera segura y eficiente (Sommerville, 2016). En este proyecto, se definieron dos perfiles principales: Administrador y Usuario Regular. Esta decisión se tomó tras una reunión con el profesor responsable de la asignatura, quien indicó que estos perfiles cubrían adecuadamente las necesidades operativas y pedagógicas del curso en el campus Piura.

El Administrador tiene acceso completo al sistema. Entre sus funcionalidades destacan: gestionar usuarios, asignar roles, editar lecciones de otros usuarios, y configurar categorías, áreas y niveles del repositorio. Este perfil está dirigido principalmente a los docentes a cargo de la asignatura, quienes requieren supervisar y organizar la información disponible.

El Usuario Regular, conformado por estudiantes y otros docentes participantes, puede acceder al repositorio, registrar nuevas lecciones aprendidas, y editar únicamente las que ha creado. Estos usuarios no tienen permisos para modificar la configuración del sistema ni para intervenir sobre el contenido de otros.

Las funcionalidades habilitadas para cada perfil fueron definidas a partir de los requerimientos identificados en la reunión con el profesor, y están orientadas a garantizar trazabilidad y control sobre las acciones críticas del sistema. Cada operación realizada queda registrada en el historial de actividad, permitiendo mantener un seguimiento claro de los cambios efectuados.

La tabla 1 presenta un resumen comparativo de los permisos asociados a cada perfil, detallando las funcionalidades disponibles para el Administrador y el Usuario Regular.

**Tabla 1**

*Tabla de permisos de usuario*

| Funcionalidad | Administrador | Usuario Regular |
|---|---|---|
| Acceder a lecciones aprendidas | ✓ | ✓ |

| Funcionalidad | Administrador | Usuario Regular |
|---|---|---|
| Registrar lecciones aprendidas | ✓ | ✓ |
| Editar lecciones propias | ✓ | ✓ |
| Editar lecciones de otros usuarios | ✓ | ✗ |
| Crear y gestionar usuarios | ✓ | ✗ |
| Asignar roles o permisos | ✓ | ✗ |
| Configurar categorías, áreas y niveles | ✓ | ✗ |
| Total de funcionalidades habilitadas | 7 | 3 |

*Nota.* Tabla que compara los perfiles de usuario (Administrador y Usuario Regular), destacando sus permisos y funcionalidades en el sistema.

### 3.4 Diseño de modelo entidad relación

El modelo entidad-relación (DER) constituye la base lógica de la estructura de datos del repositorio digital de lecciones aprendidas. Este modelo no solo permite capturar la información de manera estructurada, sino también categorizarla y establecer filtros que mejoren su accesibilidad para futuros usuarios del sistema. La lógica del modelo se alinea con los principios de normalización y buenas prácticas de diseño de bases de datos relacionales, asegurando integridad, coherencia y facilidad de mantenimiento a lo largo del tiempo (Al-Fedaghi, 2021).

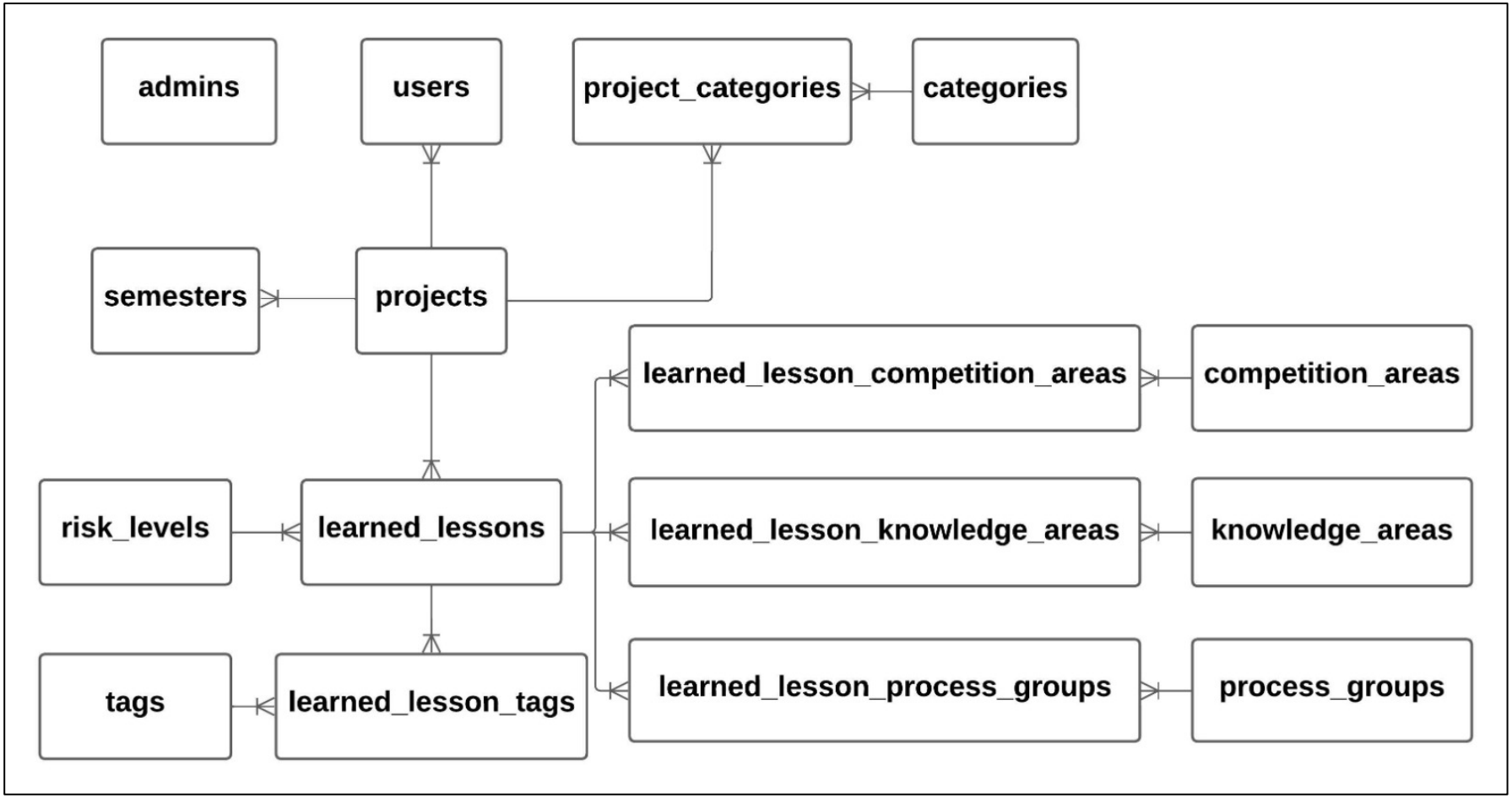
El diseño del DER se elaboró a partir del análisis de requerimientos funcionales del sistema, en coordinación con el docente responsable del curso. A través de este análisis se identificaron las entidades necesarias para capturar de forma completa y flexible la información generada por los usuarios. Las entidades principales son:

- learned_lessons: núcleo del sistema, representa cada lección registrada. Contiene atributos como name, lesson_text, documentation (fecha de creación), source_information, risk_level_id, y claves foráneas que la vinculan con otras entidades.
- users: representa a los usuarios del sistema. Incluye atributos como name, email, password_hash y is_admin, que define su rol como administrador o usuario regular.
- project: representa el proyecto académico en el que se enmarca la lección. Incluye campos como semester, user_id y category_id.
- tags, knowledge_area, process_group, completion_area: entidades auxiliares para clasificar y contextualizar las lecciones. Están relacionadas con learned_lessons mediante asociaciones muchos-a-muchos, implementadas a través de tablas intermedias como lesson_tags o lesson_process_groups.

Estas entidades y relaciones se reflejan en la Figura 2, que presenta el DER final del sistema. En este diagrama se visualizan las claves primarias, atributos principales, claves foráneas y tablas intermedias utilizadas para establecer las relaciones entre entidades.

**Figura 2**

*Diagrama Entidad-Relación (DER) del sistema*



## 3.5 Diseño de interfaz gráfica

La interfaz gráfica de usuario (GUI) del repositorio digital fue diseñada bajo un enfoque centrado en la usabilidad y la simplicidad, con el objetivo de facilitar la interacción de los usuarios y garantizar una experiencia accesible e intuitiva. Se emplearon principios de diseño consistentes con las guías de diseño de interfaces modernas, priorizando la consistencia visual, la jerarquía clara de elementos y la simplicidad funcional. La estructura y distribución de componentes se definieron a partir del análisis de requerimientos funcionales y del flujo de uso esperado por estudiantes y docentes.

El diseño sigue una arquitectura basada en componentes reutilizables, desarrollados con tecnologías web estándares (HTML, CSS y JavaScript), y estructurados en vistas dinámicas conectadas a través de rutas internas gestionadas por Ruby on Rails. Todas las pantallas mantienen una disposición coherente, con un menú lateral persistente, una barra superior de navegación y secciones de contenido claramente delimitadas.

### 3.5.1 *Pantalla de Inicio*

La figura 3 muestra la pantalla de inicio del sistema, donde se concentran los elementos clave para una navegación rápida: un menú de navegación lateral que da acceso a funcionalidades como visualización de proyectos, categorías, niveles de riesgo, áreas de conocimiento y competencia; una barra de búsqueda en la parte superior central; y un menú de usuario en la esquina superior derecha, desde donde se accede a configuraciones del perfil y cierre de sesión.

La figura 3 se muestra al final del capítulo.

### 3.5.2 *Pantalla de Gestion de Usuarios*

La figura 4 presenta la vista utilizada por administradores para registrar nuevos usuarios en un proyecto. Incluye:

- Campos de visualización del proyecto: nombre, descripción y semestre.
- Campos de entrada del usuario: nombre, apellidos, correo, contraseña, DNI y un selector para definir si el usuario será responsable del proyecto.

Estos campos fueron definidos en coordinación con el docente de la asignatura, alineados a los requerimientos operativos y pedagógicos del curso.

La figura 4 se muestra al final del capítulo.

### 3.5.3 *Pantalla de Registro de Lecciones Aprendidas*

La figura 5 muestra la interfaz para registrar una nueva lección aprendida. Incluye campos como:

- Nombre de la lección

- Fecha del suceso
- Fecha de documentación
- Situación
- Acción tomada
- Resultado
- Recomendaciones
- Nivel de riesgo

Todos estos campos fueron definidos en función del formato empleado en la asignatura y validados para asegurar integridad, completitud y relevancia de la información ingresada.

La figura 5 se muestra al final del capítulo.

### 3.5.4 *Pantalla de Búsqueda de Lecciones Aprendidas*

La Figura 6 muestra la pantalla desde la cual los usuarios pueden buscar y filtrar lecciones aprendidas. Ofrece:

- Barra de búsqueda por texto libre
- Filtros dinámicos: etiquetas, áreas de competencia, áreas de conocimiento, grupos de proceso y categorías
- Resultados en forma de tarjetas con vista previa del contenido

Esta vista mejora la eficiencia en la localización de información relevante y está directamente conectada con las pantallas de detalle y edición de lecciones.

La Figura 6 se presenta al final del capitulo.

**Figura 3**

*Captura pantalla de la Pantalla de Inicio*

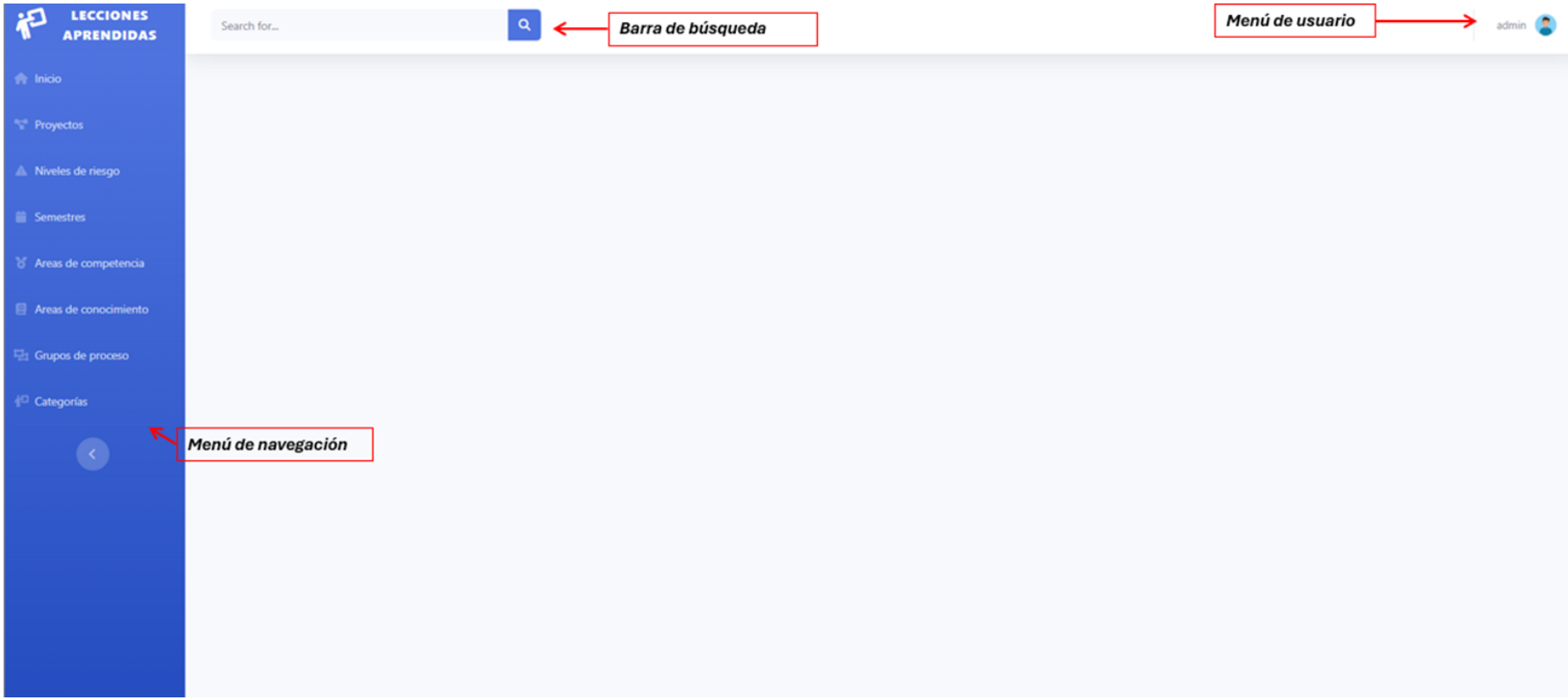


**Figura 4**

*Captura pantalla de la Pantalla de Gestión de Usuarios*

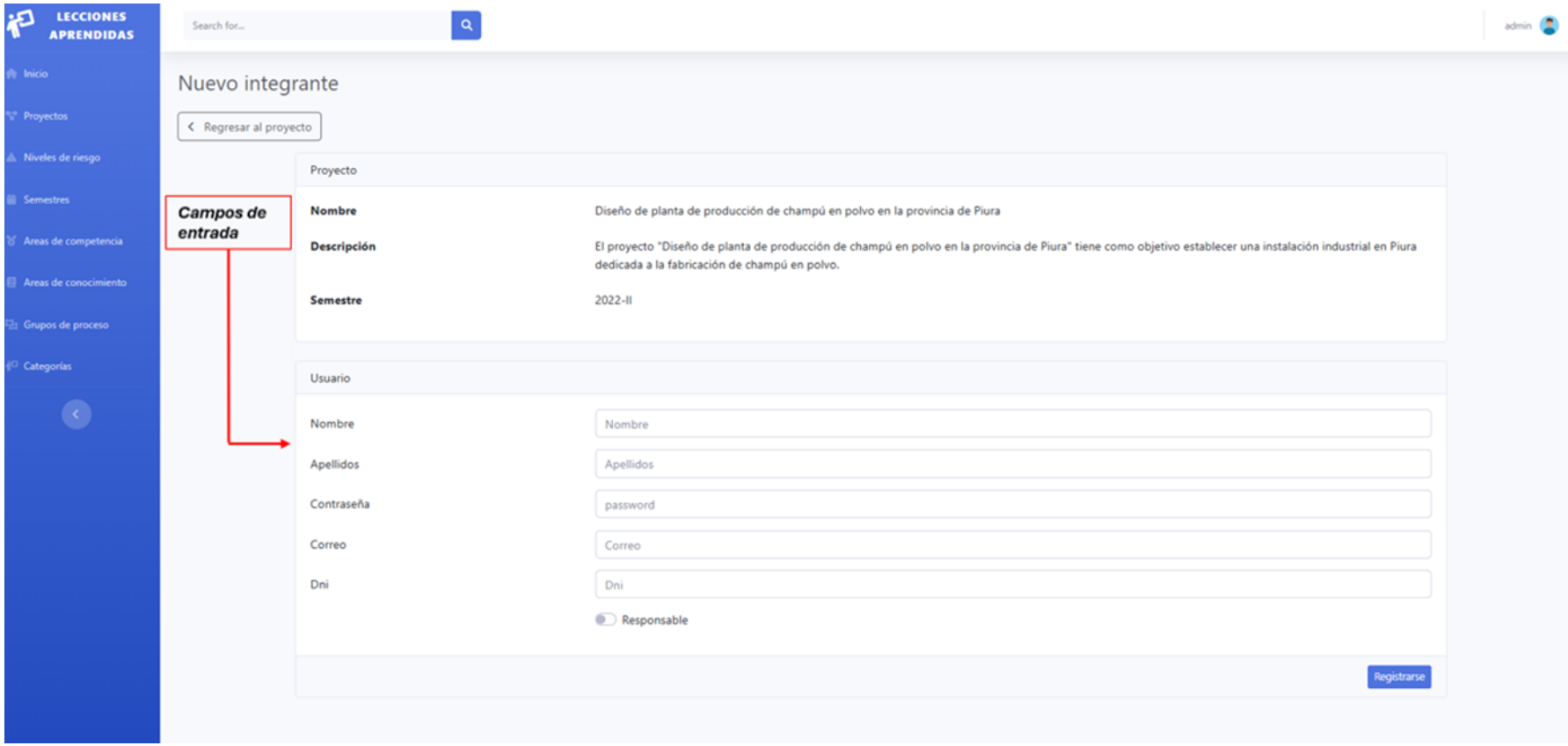


**Figura 5**

*Captura pantalla de la Pantalla de Registro de Lecciones Aprendidas*

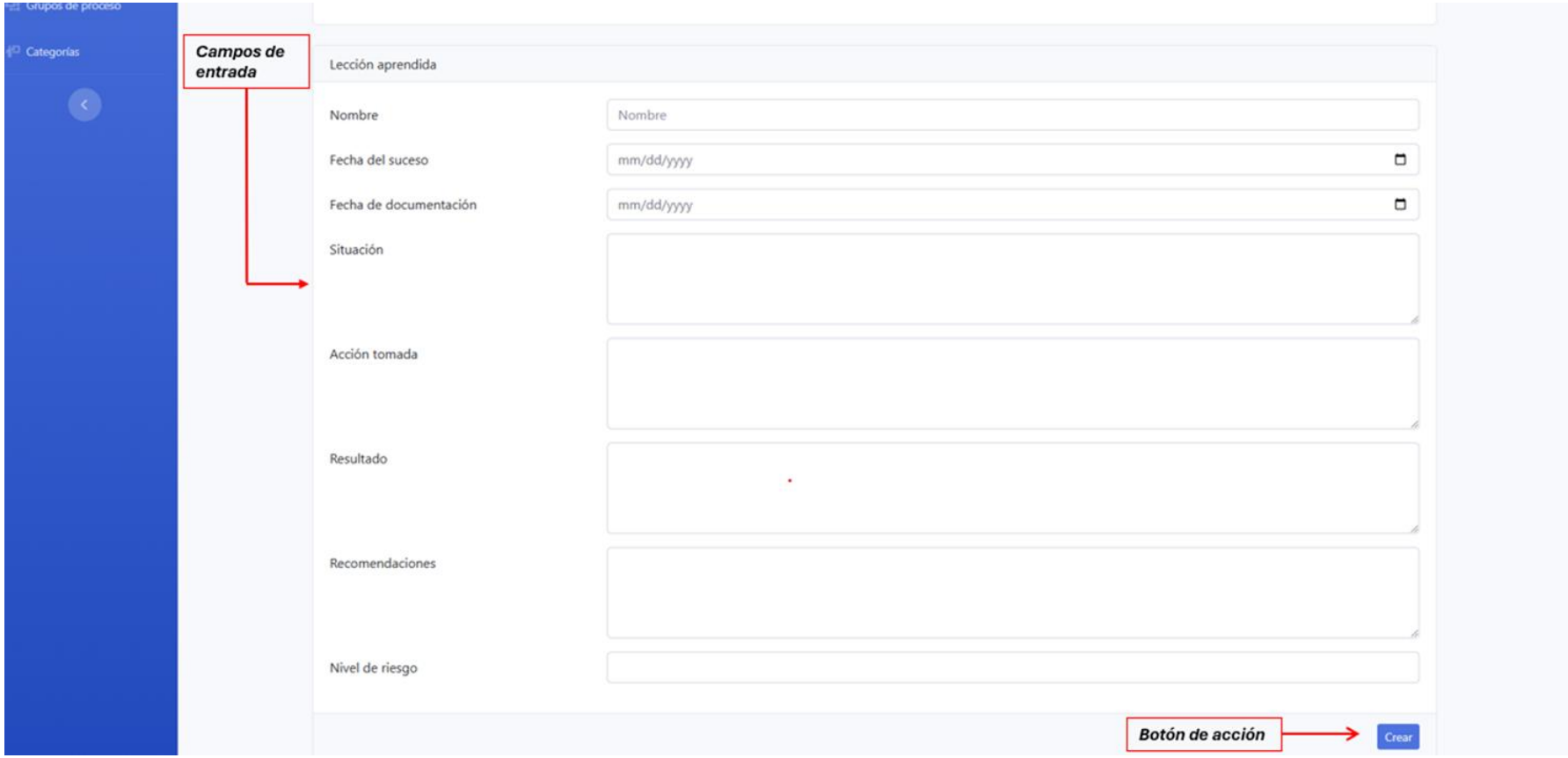
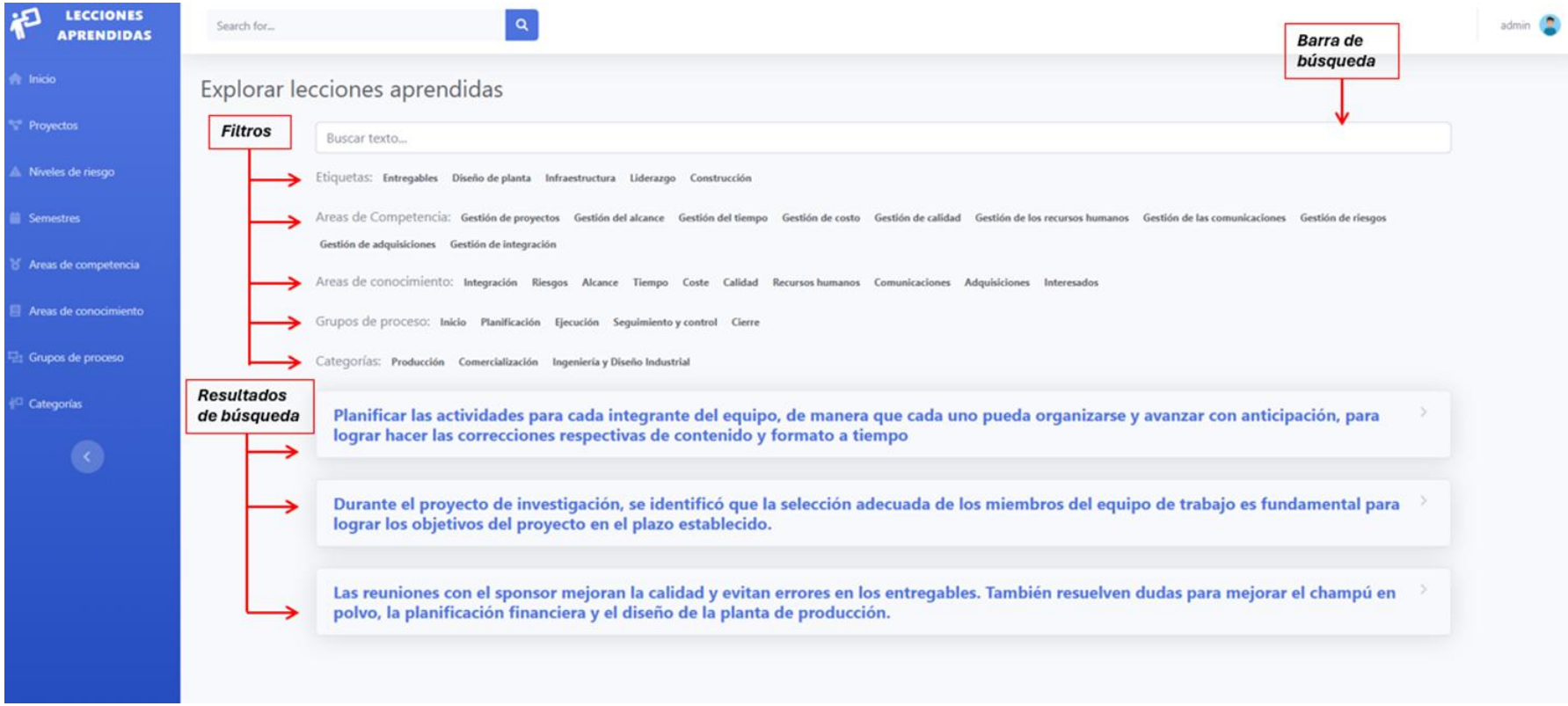


**Figura 6**

*Captura pantalla de la Pantalla de Búsqueda de Lecciones Aprendidas*



# Capítulo 4

# Desarrollo e Implementación

## 4.1 Construcción del aplicativo

La construcción del aplicativo se realizó empleando la metodología en cascada, seleccionada por su idoneidad en contextos donde los requerimientos son estables y claramente definidos desde el inicio. Aunque este modelo presenta limitaciones frente a cambios durante el desarrollo, resulta adecuado cuando se dispone de una visión completa del producto final, ya que facilita una planificación rigurosa, una ejecución secuencial y una validación estructurada de cada fase del proyecto (Mishra & Dubey, 2013). En este caso, los requerimientos fueron establecidos desde etapas tempranas y validados conjuntamente con el profesor responsable de la asignatura, sin modificaciones sustanciales a lo largo del proceso, lo cual respaldó la elección de esta metodología.

### 4.1.1 *Estructura*

La estructura del aplicativo se compone de varios módulos interconectados, cada uno con responsabilidades específicas para cubrir las funcionalidades esenciales del repositorio digital. La modularización es una práctica recomendada en el desarrollo de software, ya que mejora la mantenibilidad y escalabilidad del sistema (Dragoni, et al., 2017).

Estos módulos incluyen la gestión de usuarios, el registro de lecciones aprendidas, la búsqueda y recuperación de información, y la administración del sistema.

- Gestión de Proyectos: Este módulo permite registrar proyectos académicos realizados por los estudiantes, incluyendo atributos como nombre, descripción, semestre, categoría del proyecto y usuarios asociados. Fue diseñado para mantener un registro histórico organizado y filtrable de todos los trabajos desarrollados en la asignatura.
- Gestión de Usuarios: Este módulo permite a los administradores crear, editar y eliminar usuarios. También gestiona los roles de acceso (administrador o usuario regular), lo que garantiza el control sobre las funcionalidades disponibles para cada tipo de usuario. Se definió para cubrir las necesidades básicas de autenticación y autorización del sistema.
- Registro de Lecciones Aprendidas: Proporciona un formulario estructurado para ingresar una nueva lección. Incluye campos como título, texto de la lección, proyecto relacionado, nivel de riesgo, fuente de información, fecha de documentación, área de conocimiento, grupo de procesos y tipo de cierre (completado, cancelado, etc.). La estructura del formulario se definió tomando como base los formatos de informe actualmente usados por los estudiantes, mejorados para asegurar integridad y consistencia.

- Búsqueda y Recuperación: Este módulo facilita la localización eficiente de lecciones aprendidas previamente registradas. Los filtros implementados incluyen: semestre, categoría del proyecto, palabras clave, nombre del proyecto, autor, grupo de procesos, área de conocimiento y nivel de riesgo. Se optó por estos filtros en base al análisis de cómo los profesores y estudiantes consultan información histórica, priorizando criterios de búsqueda relevantes para la planificación y ejecución de nuevos proyectos.
- Administración del Sistema: Este módulo incluye funciones para configurar categorías, áreas, niveles de riesgo, roles de usuario y otros elementos estructurales del sistema. Fue diseñado para brindar flexibilidad a los administradores en la personalización y mantenimiento del repositorio.

La definición de estos módulos se realizó en base al análisis funcional de la asignatura, revisión de buenas prácticas en sistemas de gestión del conocimiento, y validación directa con el profesor responsable del curso.

### 4.1.2 *Código*

El desarrollo del código se realizó utilizando Ruby on Rails para el *backend* y HTML, CSS, y JavaScript para el *frontend*. Ruby on Rails se utiliza debido a su enfoque en la productividad y el desarrollo ágil de aplicaciones, facilitando la implementación de soluciones escalables (Hartl, 2016). El uso de tecnologías *frontend* modernas permite una experiencia de usuario más interactiva y dinámica.

**Figura 7**

*Captura pantalla de un extracto del código de desarrollo*

```ruby
class LearnedLesson < ApplicationRecord
  belongs_to :project
  belongs_to :risk_level
  has_many :learned_lesson_competition_areas, dependent: :destroy
  has_many :competition_areas, through: :learned_lesson_competition_areas
  has_many :learned_lesson_knowledge_areas, dependent: :destroy
  has_many :knowledge_areas, through: :learned_lesson_knowledge_areas
  has_many :learned_lesson_process_groups, dependent: :destroy
  has_many :process_groups, through: :learned_lesson_process_groups
  has_many :learned_lesson_tags, dependent: :destroy
  has_many :tags, through: :learned_lesson_tags
  strip_attributes only: %i[name]
  validates :name, presence: true
  before_save :set_defaults
      carlo, 14 months ago • se hizo la primera verisón del buscador full ja…
  private

  Windsurf: Refactor | Explain | Generate Function Comment | ×
  def set_defaults
    self.event_date = event_date.to_datetime if event_date
    self.documentation_date = documentation_date.to_datetime if documentation_date
  end
end
```

### 4.1.3 *Distribución*

La distribución del sistema se realizó en un entorno de producción configurado con Apache 2.4.x sobre Ubuntu. Apache fue seleccionado por su estabilidad, su compatibilidad con el framework Ruby on Rails y su facilidad de configuración en entornos de desarrollo educativo. Esta elección permitió implementar una solución segura y mantenible, adaptable a las necesidades específicas del curso y a los recursos disponibles.

Durante el proceso, se definieron directivas específicas para la gestión del tráfico, control de acceso y entrega de contenido estático, aprovechando la estructura modular del servidor. Esta característica técnica no solo facilita la personalización del comportamiento del servidor, sino que también incrementa su robustez y resiliencia ante errores o fallos puntuales, ya que cada módulo puede operar de forma independiente (Chaw, 2023).

El entorno resultante permitió implementar el sistema de forma controlada y replicable, sin requerir herramientas externas ni automatizaciones complejas.

## 4.2 Implementación

Durante esta etapa se realizaron pruebas manuales con el objetivo de validar que las funcionalidades del sistema se comportaran de acuerdo con lo esperado. En lugar de utilizar casos de prueba predefinidos o documentación formal, el equipo se enfocó en evaluar directamente los principales flujos funcionales desde la perspectiva del usuario final.

Este tipo de pruebas corresponde a un enfoque ad hoc, común en proyectos académicos o de pequeña escala, donde la validación práctica y contextual resulta suficiente para garantizar el cumplimiento de los requisitos esenciales (Sommerville, 2016). Como señala Gupta (2020), cuando estas pruebas se orientan adecuadamente a los flujos más representativos del sistema, pueden ser altamente efectivas para detectar errores funcionales relevantes y mejorar la calidad del producto.

En este proyecto se llevaron a cabo dos tipos principales de prueba: pruebas de integración y pruebas de aceptación, ambas ejecutadas manualmente en un entorno de pruebas local.

### 4.2.1 *Pruebas de Integración*

Estas pruebas verificaron la correcta interacción entre los componentes del sistema. Se evaluaron flujos como:

- Registro de una nueva lección mediante el formulario web.
- Aplicación de filtros por autor, categoría o palabras clave.
- Visualización ordenada de resultados.
- Control de acceso según el rol del usuario.

El equipo de desarrollo ejecutó las pruebas con datos representativos, observando directamente los resultados. No se detectaron errores críticos, pero se aplicaron mejoras menores en la presentación de datos y usabilidad.

### 4.2.2 *Pruebas de Aceptación*

Realizadas junto al docente responsable del curso, estas pruebas validaron que el sistema respondiera a los objetivos pedagógicos definidos.

Se revisaron aspectos como:

- Claridad de los formularios.
- Navegación del sistema.
- Utilidad de los filtros.
- Formato y legibilidad de la información.

La retroalimentación fue positiva. Se implementaron ajustes menores en textos, etiquetas y diseño visual antes del despliegue final.

### 4.2.3 *Consideraciones Finales*

Las pruebas ad hoc realizadas fueron adecuadas para validar la funcionalidad del sistema dentro del contexto académico. Su ejecución sobre flujos clave y con participación directa del usuario permitió detectar errores sin recurrir a métodos formales.

Sommerville (2016) reconoce que las pruebas informales son válidas en entornos educativos, siempre que se cumplan los objetivos funcionales. Por su parte, Gupta (2020) señala que las pruebas ad hoc, bien orientadas, mejoran la calidad del producto. Sin embargo, ambos coinciden en que este enfoque debe complementarse con métodos formales en contextos críticos o regulados.

## 4.3 Despliegue

El despliegue del repositorio digital se realizó en el servidor de producción de la Universidad de Piura, utilizando Apache/2.4.52 en Ubuntu como servidor web. La aplicación fue instalada manualmente, sin un proceso automatizado de integración o monitoreo continuo.

Esta decisión se debió a limitaciones propias del servidor proporcionado por la universidad. Considerando además que se trata de un proyecto académico, de uso interno y con bajo volumen de tráfico, se optó por una instalación funcional y estable, priorizando la simplicidad operativa. Esta elección se alinea con lo señalado por Sommerville (2016), quien indica que en entornos educativos o con recursos limitados, es común realizar despliegues manuales como una solución práctica y suficiente para sistemas pequeños.

Para verificar la disponibilidad y correcta configuración del sistema, se utilizó el siguiente comando en la terminal:

**Figura 8**

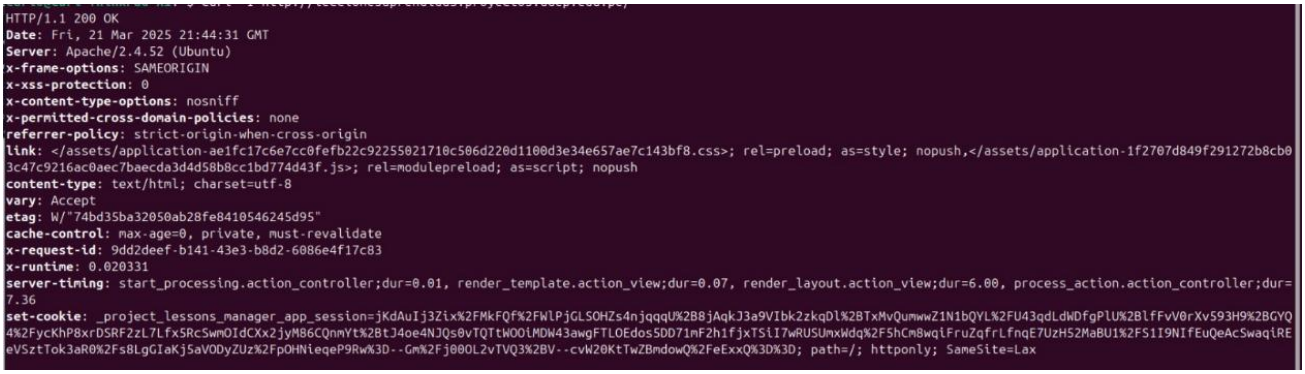
*Comando utilizado para verificar la disponibilidad del repositorio*

```nginx
curl -I http://leccionesaprendidas.proyectos.udep.edu.pe/
```

Este comando permite consultar únicamente los encabezados de la respuesta HTTP enviada por el servidor, lo cual es útil para comprobar el estado del servicio. La respuesta obtenida fue:

**Figura 9**

*Respuesta del servidor Apache*




La respuesta indica que el servidor se encuentra en funcionamiento y que el sistema es accesible mediante el protocolo HTTP, devolviendo un código 200 OK, que representa una solicitud exitosa.

A pesar de no haberse implementado un sistema formal de monitoreo o despliegue continuo, el entorno de producción responde correctamente y permite el acceso completo a las funcionalidades del repositorio digital.

# Capítulo 5

# Resultado y Evaluación

## 5.1 Reconocimiento de la Importancia de la Evaluación

La evaluación del impacto del repositorio digital es fundamental para determinar su efectividad en la gestión del conocimiento y el aprendizaje colaborativo dentro de la asignatura de Proyectos. Aunque esta tesis no incluye una evaluación empírica directa posterior a su implementación, se proponen criterios que permitirán medir su utilidad en futuras aplicaciones.

Durante las pruebas preliminares y presentaciones internas, algunos docentes ofrecieron retroalimentación positiva, destacando la interfaz intuitiva del sistema y la facilidad para acceder a las lecciones aprendidas como elementos que facilitan un entorno colaborativo. Como parte de esta validación inicial, el sistema fue presentado en el 28.º Congreso Internacional de Dirección e Ingeniería de Proyectos (CIDIP/ICPME 2024), organizado por AEIPRO-IPMA Spain y la Universidad de Jaén. En dicho evento, que reunió a representantes académicos y profesionales de distintos países, la propuesta recibió una acogida favorable, generando incluso el interés de otras instituciones en replicar su uso.

Si bien estos comentarios no constituyen una evaluación formal sistematizada, sí representan una primera validación práctica del enfoque del sistema y de su alineación con las necesidades reales del entorno educativo. Según Zamiri y Esmaeili (2024), el intercambio de conocimiento en comunidades de aprendizaje es clave para optimizar los resultados educativos, lo que refuerza la necesidad de evaluar periódicamente la eficacia de herramientas digitales como este repositorio.

## 5.2 Criterios e Indicadores de Evaluación Propuestos

Para llevar a cabo una evaluación efectiva del impacto del repositorio, se proponen algunas métricas y criterios basados en estudios recientes:

- Eficiencia en la Recuperación de Información: Se puede medir el tiempo que los usuarios tardan en encontrar lecciones aprendidas antes y después del uso del repositorio. Una reducción significativa en este tiempo indicaría una mejora en la eficiencia de búsqueda (Zamiri & Esmaeili, 2024).
- Frecuencia de Uso: El nivel de compromiso de los estudiantes con las actividades de aprendizaje en línea se relaciona con factores como la comunicación, la interacción y la presencia del docente. Estudios recientes han demostrado que una mayor participación en actividades como foros, cuestionarios en línea y tareas colaborativas aumenta la frecuencia de uso de las herramientas educativas digitales, lo que puede influir positivamente en el aprendizaje (Vermeulen & Volman, 2024). Además, un

compromiso constante con actividades en línea ha sido identificado como un factor clave para el éxito en entornos de aprendizaje en línea (Rajabalee & Santally, 2021)

- Reutilización de Conocimiento: El uso del repositorio debe facilitar la transferencia de conocimientos previos a nuevos proyectos. La frecuencia con que se consultan lecciones anteriores puede ser un buen indicador de esta (Santos-Hermosa et al., 2017).
- Satisfacción del Usuario: La satisfacción con plataformas digitales influye en su aceptación. Se puede evaluar mediante encuestas que valoren usabilidad, claridad en la navegación y percepción de utilidad (Rajabalee & Santally, 2021).
- Impacto en el Aprendizaje: El uso activo del repositorio debería mejorar la comprensión, retención de contenidos y resultados académicos. Este impacto podría medirse a través de indicadores cualitativos y cuantitativos, como calificaciones o percepción docente (Bergdahl et al., 2020).

### 5.3 Evaluación Continua y Ajustes

Además de una evaluación inicial, se propone implementar un proceso de evaluación continua, que incluya:

- Monitoreo del uso del sistema
- Recolección periódica de retroalimentación de los usuarios
- Actualización de funcionalidades en función de las necesidades académicas

Este enfoque de mejora continua está respaldado por estudios recientes que destacan la importancia de sistemas adaptativos en entornos educativos digitales. Según Lai et al. (2022), el desarrollo de soluciones efectivas en e-learning requiere ajustes permanentes en tecnología y procesos pedagógicos. Asimismo, Giannakos et al. (2022) afirman que un sistema que evoluciona con su comunidad logra un mayor impacto en la experiencia y el rendimiento de los estudiantes.

## Conclusiones

La implementación del repositorio digital representa un avance significativo en la gestión del conocimiento en la Universidad de Piura, particularmente en el contexto de la asignatura de Proyectos. Al integrar funcionalidades como la gestión de usuarios, la categorización de proyectos y un sistema eficiente de búsqueda, esta herramienta facilita el intercambio de conocimientos y promueve una cultura de aprendizaje colaborativo, en línea con los objetivos pedagógicos del curso.

Este desarrollo reafirma la relevancia de las herramientas digitales en el enriquecimiento de los procesos educativos, como también se discute en la investigación de Guerrero Chanduví et al. (2023). A pesar de los desafíos técnicos y operativos enfrentados durante el desarrollo, el sistema cumple con su objetivo: proporcionar un entorno educativo dinámico y en mejora continua.

La puesta en marcha del repositorio se concibe con la expectativa de facilitar su integración en los procesos educativos, fortaleciendo la participación activa de estudiantes y docentes. Su diseño intuitivo y el enfoque en la usabilidad y accesibilidad garantizan una experiencia de usuario positiva, fomentando el registro y consulta de lecciones aprendidas como parte del ciclo de mejora académica.

Aunque los resultados concretos del impacto del repositorio solo podrán evaluarse tras un periodo de uso sostenido, la respuesta inicial por parte del profesorado ha sido positiva, destacando la facilidad de acceso a la información y la claridad de la interfaz como elementos clave para fomentar un entorno colaborativo. Este feedback preliminar refuerza las expectativas sobre su potencial como herramienta educativa.

Finalmente, se reconoce la importancia de establecer un proceso de evaluación continua, que permita identificar oportunidades de mejora y adaptar la herramienta a las necesidades cambiantes de la comunidad académica. La literatura reciente enfatiza que las plataformas digitales en la educación no solo promueven el aprendizaje activo, sino que también fortalecen el desarrollo de competencias y optimizan los procesos de enseñanza-aprendizaje (Lai et al., 2022; Giannakos et al., 2022). Este enfoque no solo confirmará las expectativas iniciales, sino que también proporcionará valiosas perspectivas sobre la efectividad de las estrategias de gestión del conocimiento en contextos educativos.

## Recomendaciones

Si bien esta tesis no contempló una evaluación cuantitativa del impacto del repositorio digital, se sugiere que, una vez implementado y en funcionamiento, se desarrollen estudios posteriores que analicen su efectividad en distintos aspectos del entorno académico.

Como se detalló en el capítulo anterior, existen diversas métricas que permitirían evaluar su impacto en la gestión del conocimiento y el aprendizaje. Se recomienda aplicar estos criterios a través de estudios estructurados que validen sus resultados y orienten la evolución futura del sistema.

Estas investigaciones contribuirán a consolidar la herramienta como parte integral de los procesos educativos, garantizando su mejora continua y su alineación con las necesidades reales de estudiantes y docentes.

## Glosario

**Arquitectura Monolítica:** Modelo arquitectónico en el cual todos los componentes del sistema están integrados en una sola unidad desplegable.

**Back-end:** Parte del sistema encargada de la lógica de negocio, el procesamiento de datos y la comunicación con la base de datos.

**Base de datos relacional:** Sistema de almacenamiento de datos organizado en tablas relacionadas entre sí mediante claves primarias y foráneas.

**Controlador:** Componente del patrón MVC encargado de recibir las solicitudes del usuario, procesarlas y retornar una respuesta adecuada.

**Despliegue:** Proceso de publicación de una aplicación en un entorno de producción para su uso real.

**Entidades:** Elementos o conceptos representados en una base de datos que almacenan información relevante para el sistema.

***Framework*:** Entorno de trabajo que proporciona una estructura base y herramientas para facilitar el desarrollo de software.

**Front-end:** Parte visual de una aplicación web con la que el usuario interactúa directamente.

**Gestión del Conocimiento:** Proceso sistemático de capturar, estructurar y difundir información útil dentro de una organización para mejorar la toma de decisiones.

**Interfaz Gráfica de Usuario (GUI):** Diseño visual que permite la interacción del usuario con el sistema mediante elementos gráficos como botones, formularios y menús.

**Lecciones Aprendidas (LA):** Conocimientos adquiridos a través de experiencias pasadas que se documentan para evitar errores futuros y replicar buenas prácticas.

**Modelo Entidad-Relación (DER):** Representación gráfica de las entidades de una base de datos y sus relaciones.

**Pantalla de Búsqueda:** Interfaz diseñada para facilitar el filtrado y recuperación de registros almacenados en el sistema.

**Pantalla de Registro:** Interfaz que permite la introducción y validación de datos para su posterior almacenamiento.

**Pantalla de Gestión de Usuarios:** Módulo de administración de cuentas de usuarios y permisos en el sistema.

**Pruebas de Aceptación:** Evaluaciones realizadas por los usuarios finales para confirmar que el sistema cumple con los requisitos funcionales establecidos.

**Pruebas de Integración:** Verificación del correcto funcionamiento conjunto de los distintos módulos del sistema.

**Pruebas Unitarias:** Validaciones de componentes individuales del sistema para asegurar su funcionamiento independiente.

**Repositorio Digital:** Plataforma tecnológica que centraliza y organiza recursos digitales para su consulta, edición y conservación a largo plazo.

**Roles de Usuario:** Clasificación de usuarios según sus privilegios de acceso y funcionalidades habilitadas dentro del sistema.

**Ruby on Rails:** *Framework* de desarrollo web en Ruby que sigue el patrón MVC, ideal para aplicaciones de rápida implementación.

**Usabilidad:** Medida de la facilidad con la que los usuarios pueden interactuar de manera efectiva con una interfaz.

**Lista de abreviaturas**

CSS: Hojas de Estilo en Cascada (Cascading Style Sheets).

DER: Diagrama Entidad-Relación.

GUI: Interfaz Gráfica de Usuario (Graphic User Interface).

HTML: Lenguaje de Marcado de Hipertexto (Hypertext Markup Language).

KMS: Sistema de Gestión del Conocimiento (Knowledge Management System).

LA: Lecciones Aprendidas.

MVC: Modelo-Vista-Controlador.

SQL: Lenguaje de Consulta Estructurado (Structured Query Language).

UI: User Interface (Interfaz del Usuario).

UX: User Experience (Experiencia de Usuario).

**Referencias**


Al-Debagy, O., & Martinek, P. (2018). A Comparative Review of Microservices and Monolithic Architectures. In *Proceedings of the 18th IEEE International Symposium on Computational Intelligence and Informatics (CINTI 2018)* (pp. 000149-000154). Budapest, Hungary: IEEE.

Al-Fedaghi, S., & Modhaffar, M. (2021). Software engineering meets systems engineering: Conceptual modeling applied to engineering operations. *International Journal of Computer Science and Network Security*, 329-342. doi:10.22937/IJCSNS.2021.21.10.47

Alvarado Pérez, E. (2015). *Modelo para implantar la gestión del conocimiento en instituciones de educación superior: caso la Facultad de Ingeniería de la Universidad de Piura. Universidad de Piura.* Universidad de Piura.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Manifesto for Agile Software Development*. Retrieved from Agile Manifesto: https://agilemanifesto.org/

Bergdahl, N., Nouri , J., & Fors , U. (2020). Disengagement, engagement and digital skills in technology-enhanced learning. *Educ Inf Technol 25*, 957-982. doi:10.1007/s10639-019-09998-w

Bloomfire. (2021). *The Importance of Knowledge Management in Higher Education*. Retrieved from Bloomfire: https://bloomfire.com/blog/knowledge-management-in-higher-education/

Brynjolfsson, E., & McAfee, A. (2014). *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies.* W. W. Norton & Company.

Chaw, M. (2023). Exploring Web Technologies: A Comprehensive Study on Apache Web Server, AJAX, Cookies, HTTP Sessions, and Web Development Languages. .

Davenport, T. H., & Prusak, L. (1998). *Working Knowledge: How Organizations Manage What They Know.* Harvard Business School Press.

Deshmukh , R., Bhandarkar, A., & Tawar, O. (2025). Web Back-End Development Technologies. *International Research Journal of Modernization in Engineering, Technology and Science*, 6520-6525.

Dragoni, N., Giallorenzo, S., Lluch-Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, Today, and Tomorrow. In M. Mazzara, & B. Meyer, *Present and Ulterior Software Engineering* (p. Pages: 195). Cham: Springer International Publishing.

Farley, D., & Humble, J. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Addison-Wesley Professional.

Giannakos, M. N., Mikalef, P., & Pappas , I. O. (2022). Systematic Literature Review of E-Learning Capabilities to Enhance Organizational Learning. *Inf Syst Front 24*, 619-635. doi:10.1007/s10796-020-10097-2

Gleick, J. (2011). *The Information: A History, a Theory, a Flood.* Pantheon Books.

Guerrero Chanduví, D. A., Hurtado Jara, O., Zacarías Vélez, C., Jaramillo Córdova, D. d., & Cabellos Román, L. F. (2023). Diseño de repositorio digital de lecciones aprendidas como herramienta de gestión del conocimiento caso de estudio asignatura de proyectos. *Proceedings from the International Congress on Project Management and Engineering.* AEIPRO. Retrieved from http://dspace.aeipro.com/xmlui/handle/123456789/3510

Gupta, A. (2020). Beyond the Script: Elevating Software Product Quality with Structured Ad-Hoc / Exploratory Testing - A Case Study. *International Journal of Science and Research (IJSR)*, 1644-1648. doi:10.21275/SR24608152029

Hartl, M. (2016). *Ruby on Rails Tutorial: Learn Web Development with Rails.* Addison-Wesley Professional: Boston, MA.

Hossain, A. (2023). A Systematic Mapping Study on Scrum and Kanban in Software Development. *Åbo Akademi University, The Faculty of Science and Engineering*. doi:NBN:fi-fe2023060552468

IBM. (2021, Octubre 6). *What is Django?* Retrieved from https://www.ibm.com/think/topics/django

Jain, R., Shrivastava, V., Pandey, A., & Sharma, A. (2024). Modern Web Development using CSS & HTML. *International Journal of Emerging Science and Engineering*, 13-16.

Jaramillo Cordova, D., & Cabellos Roman, L. (2024). *Diseño de un repositorio digital para el reúso del conocimiento de lecciones aprendidas: caso de estudio en la Universidad de Piura con los estudiantes de la asignatura de Proyectos.* Piura: Universidad de Piura.

Jugdev, K. (2012). Learning from Lessons Learned: Project Management Research Program. *American Journal of Economics and Business Administration*, 4(1), 13-22. Retrieved from https://www.researchgate.net/publication/258933493_Learning_from_Lessons_Learned_Project_Management_Research_Program

Lai, J. W., De Nobile, J., Bower , M., & Breyer , Y. (2022). Comprehensive evaluation of the use of technology in education – validation with a cohort of global open online learners. *Educ Inf Technol 27*, 9877-9911. doi:10.1007/s10639-022-10986-w

Lavarel. (2025, Febrero 2024). *Laravel 12.x Documentation*. Retrieved from https://laravel.com/docs/12.x

MacDowell, P. (2023). VR let my creativity out: Youth creating with immersive learning technologies. *International Journal of Emerging and Disruptive Innovation in Education: VISIONARIUM*, Article 3. doi:10.62608/2831-3550.1012

Meyer, B. (2014). *Agile! The Good, the Hype and the Ugly.* Springer International Publishing. doi:10.1007/978-3-319-05155-0

Mishra, A., & Dubey, D. (2013). A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios. *International Journal of Advance Research in Computer Science and Management Studies*, 64-69.

Nonaka, I., & Takeuchi, H. (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* New York, NY: Oxford University Press.

Nupap, S. (2022). Knowledge Management System by applying Knowledge Creating Company: Transforming Tacit to Explicit Knowledge. *2022 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, 439-444. doi:10.1109/ECTIDAMTNCON53731.2022.9720388

Railsware. (2023, Abril 3). *Why Use Ruby on Rails for Your Product*. Retrieved from https://railsware.com/blog/why-use-ruby-on-rails-for-your-product/

Rajabalee, Y., & Santally, M. (2021). Learner satisfaction, engagement and performances in an online module: Implications for institutional e-learning policy. *Educ Inf Technol 26*, 2623–2656. doi:10.1007/s10639-020-10375-1

Santos-Hermosa, G., Ferran-Ferrer, N., & Abadal, E. (2017). Repositories of Open Educational Resources: An Assessment of Reuse and Educational Aspects. *Repositories of Open Educational Resources: An Assessment of Reuse and Educational Aspects. The International Review of Research in Open and Distributed Learning.*, 18. doi:10.19173/irrodl.v18i5.3063

Shvachych, G., Aleksieiev, M., Havrylov, R., & Mamuzić, I. (2024). Development of a web platform for software installation. *17th International symposium of the Croatian Metallurgical Society - SHMD 2024* (pp. 489-489). Zagreb: Hrvatsko metalurško društvo. Retrieved from https://www.croris.hr/crosbi/publikacija/resolve/croris/825208

SoluteLabs. (2023). *Ruby on Rails: Why It Remains a Leader in Web Development*. Retrieved from solutelabs: https://www.solutelabs.com/blog/ruby-on-rails-web-development

Sommerville, I. (2016). *Software Engineering 10th Edición.* Pearson Education Limited.

Thanachawengsakul, N., Wannapiroon, P., & Nilsook, P. (2019). Synthesis of digital knowledge engineering repository management system. *International Journal of e-Education, e-Business, e-Management and e-Learning, 9*(4), 348-356. doi:10.17706/ijeeee.2019.9.4.348-356

Tiwari, H. (2023). *A Comparative Analysis of Nginx and Apache Web Servers: Performance, Scalability, and Features.* Retrieved from Insights2Techinfo: https://insights2techinfo.com/a-comparative-analysis-of-nginx-and-apache-web-servers-performance-scalability-and-features/

Turner, J., Keegan, A., & Crawford, L. (2000). Learning by Experience in the Project-Bases Organization. *Research Conference 2000: Project Management Research at the Turn of the Millennium*. Retrieved from https://www.pmi.org/learning/library/learning-experience-project-based-organization-8534

Vermeulen, E., & Volman, M. (2024). Promoting Student Engagement in Online Education: Online Learning Experiences of Dutch University Students. *Technology, Knowledge and Learning 29*, 941-961. doi:10.1007/s10758-023-09704-3

Wallace, D. (2007). *Knowledge Management: Historical and Cross-Disciplinary Themes.* Libraries Unlimited.

Whatfix. (2024). *The Importance of a Knowledge Sharing Culture.* Retrieved from https://www.whatfix.com/blog/knowledge-sharing-culture/

Yadav, B., Shrivastava, V., & Pandev, A. (2024). MySQL: Database Design for Performance and Scalability. *International Journal of Research Publication and Reviews*, 7631–7638.

Zamiri, M., & Esmaeili, A. (2024). Methods and Technologies for Supporting Knowledge Sharing within Learning Communities: A Systematic Literature Review. *Administrative Sciences*, 14, 17. doi:10.3390/admsci14010017