Migración del componente fron-end del software Academiapp para la empresa Webcloster SAS

Santiago Garcia Gil

Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Sede (06) Tuluá
2025

Migración del componente fron-end del software Academiapp para la empresa Webcloster SAS

Santiago Garcia Gil
Código 202060017
santiago.garcia.gil@correounivalle.edu.co

Modalidad pasantía

Director
Ms.C Royer David Estrada Esponda. Ing
Profesor de la Escuela de Ingeniería de Sistemas y Computación
royer.estrada@correounivalle.edu.co

Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Sede (06) Tuluá
2025

Índice

1. Dedicatoria **4**

2. Resumen **5**

3. Agradecimientos **7**

4. Introducción **8**

5. Descripción del problema **9**

6. Hipótesis **10**
6.1. Hipótesis General 10
6.2. Hipótesis Específicas 10
6.3. Variables 10

7. Formulación del problema **11**

8. Marco Referencial **12**
8.1. Marco Conceptual 12
8.1.1. Aplicaciones Web 12
8.1.2. Gestión Académica y Administrativa 12
8.1.3. Framework Angular 12
8.1.4. Migración de Software 12
8.1.5. Despliegue 12
8.1.6. Deuda Técnica 13
8.1.7. Rendimiento y Compatibilidad 13
8.1.8. Metodologías Ágiles 13
8.1.9. Seguridad 13

9. Marco Teórico **14**
9.1. Ingeniería de Software 14
9.2. Sistemas de Información en Entornos Educativos 14
9.3. Framework Angular como Plataforma de Desarrollo 14
9.4. Migración y Modernización de Software 14
9.5. Deuda Técnica 14
9.6. Rendimiento y Compatibilidad 15
9.7. Metodologías Ágiles y Scrum 15
9.8. Antecedentes 15
9.8.1. Q10 Académico 15
9.8.2. Sistema SINU 18
9.8.3. Moodle 18
9.8.4. Canvas LMS 19
9.9. Tabla comparativa de sistemas relacionados 21

10. Alcance **22**
10.1. Descripción general . . . 22
10.2. Alcance funcional (incluye) . . . 22
10.3. Alcance no funcional (incluye) . . . 22
10.4. Fuera de alcance (no incluye) . . . 23
10.5. Entregables . . . 24
10.6. Criterios de aceptación / métricas . . . 25
10.7. Supuestos y restricciones . . . 26
10.8. Riesgos principales . . . 26
10.9. Plan / Cronograma (resumen) . . . 26
10.10. Objetivos . . . 26
10.10.1. Objetivos específicos . . . 27

11. Metodología **27**
11.1. Gestión del proyecto . . . 27
11.1.1. Tareas y responsabilidades . . . 29
11.1.2. Planificación del proyecto . . . 32
11.1.3. Herramientas utilizadas . . . 32

12. desarrollo de proyecto **34**
12.1. Análisis del estado inicial del proyecto . . . 35
12.2. Planeación del proceso de la migración . . . 36
12.2.1. Objetivos técnicos de la migración . . . 36
12.2.2. Alcance técnico de la migración . . . 37
12.3. Gestión de dependencias y librerías externas . . . 38
12.4. Proceso de actualización del framework Angular . . . 39
12.4.1. Comandos principales utilizados . . . 39
12.5. Refactorización y mejora del código . . . 39
12.6. Pruebas funcionales y validación . . . 39
12.7. Documentación y cierre del proceso . . . 41

13. Resultados Obtenidos **42**
13.1. Migración exitosa del framework Angular . . . 42
13.2. Actualización y mejora de la sección de noticias . . . 43
13.3. Ajustes visuales y funcionales en el Dashboard . . . 44
13.4. Actualización de componentes basados en calendario . . . 45
13.5. Reducción de advertencias y elementos deprecados . . . 46
13.6. Mejora en la mantenibilidad y preparación para futuras fases . . . 46
13.7. Recomendaciones . . . 46

14. Conclusiones **48**

Índice de figuras

1. loguin Q10 Académico. 16
2. Gestión académica en Q10 Académico. 17
3. Pantalla loguin del Sistema SINU. 18
4. Entorno virtual de aprendizaje Moodle. 19
5. Demo Moodle. 19
6. Interfaz principal de Canvas LMS. 20
7. Reunión de seguimiento del equipo técnico durante el desarrollo del proyecto. 28
8. Sesión de revisión y validación de avances realizada en Webcloster SAS. . . . 28
9. Listado de tareas gestionadas en WebDesk durante la migración. 30
10. Registro de tiempo y progreso por tarea dentro de WebDesk. 30
11. Merge Request en GitLab con los cambios asociados a la migración. 31
12. Historial de commits registrado durante la actualización del front-end. 31
13. Formulario de creación de noticias en AcademiApp 43
14. Visualización de una noticia publicada en AcademiApp 44
15. Dashboard de AcademiApp tras la migración a Angular 16 45

1. Dedicatoria

Este trabajo lo dedico principalmente a Dios, quien me ha acompañado y guiado a lo largo de toda mi carrera, permitiéndome seguir adelante incluso en medio de dificultades económicas, y mostrándome siempre su cuidado y providencia. A mis padres, por inculcarme la fe en Dios y por brindarme su amor y apoyo incondicional, además de enseñarme el valor de la disciplina, la constancia, el esfuerzo y el trabajo honesto para alcanzar mis objetivos. A mis hermanos, de quienes he aprendido la paciencia y con quienes espero haber compartido, tanto como hermano mayor como menor, parte de lo que he ido aprendiendo en el camino. Finalmente, a mis amigos y a todas las personas que han confiado en mí y me han motivado a seguir avanzando en este proceso académico.

2. Resumen

Las plataformas académicas tienen que estar siempre actualizadas desde el punto de vista tecnológico para asegurar que sean estables, compatibles y fáciles de mantener. El propósito de este trabajo de grado, que se llevó a cabo en la empresa Webcloster SAS como pasantía, fue llevar a cabo una migración y actualización parcial del componente *front-end* de la plataforma AcademiApp. La actualización del marco Angular, desde la versión 14 hasta la 16, y la adecuación y refactorización de módulos importantes del sistema fueron los objetivos del proyecto.

Se usaron métodos ágiles en el desarrollo del trabajo, que se apoyaron en instrumentos para el control de versiones y la gestión de tareas, lo que posibilitó un monitoreo organizado y colaborativo de las actividades llevadas a cabo. A lo largo del proceso, se corrigieron advertencias de compilación, se renovaron dependencias obsoletas y se comprobó la operatividad de componentes esenciales como el dashboard, la sección de noticias y los módulos fundamentados en calendarios.

Como consecuencia, se logró una plataforma más estable, que es compatible con las versiones más actuales de Angular y está lista para futuras etapas de progresos tecnológicos. Gracias a la experiencia obtenida en la pasantía, se pudieron poner en práctica los conocimientos teóricos en un ambiente real de desarrollo, lo que benefició tanto al proceso de formación profesional del alumno como a la compañía.

Abstract

Academic platforms must remain constantly updated from a technological perspective in order to ensure stability, compatibility, and ease of maintenance. In this context, this undergraduate project, developed as an internship at the company Webcloster SAS, aimed to carry out a partial migration and update of the front-end component of the AcademiApp platform. The project focused on upgrading the Angular framework from version 14 to version 16, as well as adapting and refactoring key system modules.

Agile methodologies were applied during the development process, supported by version control tools and task management platforms, which enabled organized and collaborative monitoring of the activities performed. Throughout the process, compilation warnings were corrected, deprecated dependencies were updated, and the functionality of essential components such as the dashboard, the news section, and calendar-based modules was validated.

As a result, a more stable platform was achieved, compatible with the most recent versions of Angular and prepared for future stages of technological evolution. The experience gained during the internship allowed theoretical knowledge to be applied in a real development environment, benefiting both the student's professional training process and the company.

3. Agradecimientos

A todas las personas que hicieron posible la realización de este trabajo de grado. A mi director, Royer, por su guía y apoyo durante todo el proceso. A Webcloster por brindarnos a mí, y mis excelentes compañeros de trabajo, la oportunidad de ser parte del proyecto Invessoft y aprender en gran manera de el. A la Universidad del Valle por haberme recibido y permitido formarme como profesional colombiano. A mis profesores, por su dedicación y por impartirme sus conocimientos. A mis compañeros de clase, por su fraternidad y por los momentos de aprendizaje compartidos. A mi familia, por su amor, por creer en mí y por brindarme el respaldo necesario.

4. Introducción

Hoy en día, las instituciones de educación superior afrontan el reto de manejar una cantidad cada vez mayor de datos administrativos y académicos, al tiempo que intentan proporcionar a su comunidad universitaria servicios más rápidos, accesibles e integrados. En estas circunstancias, las aplicaciones web se han establecido como instrumentos fundamentales para perfeccionar procesos internos, disminuir las cargas operativas y enriquecer la experiencia de los alumnos, así como del profesorado y el personal administrativo. Esto se debe al rol protagónico que ha desempeñado el software en la digitalización de diferentes sectores[1]. Además, varias universidades necesitan plataformas que centralicen su información y disminuyan la fragmentación entre sistemas, lo que posibilita una administración más eficaz y coherente de sus procesos[2].

Webcloster SAS, compañía enfocada en desarrollar soluciones tecnológicas para la administración de organizaciones, creo la plataforma AcademiApp con el objetivo de unificar y actualizar los trámites académicos y administrativos de universidades y otros centros educativos. No obstante, la actualización y optimización de los elementos de la plataforma se vuelve indispensable para garantizar un mejor desempeño, estabilidad, capacidad de mantenimiento y compatibilidad con tecnologías emergentes debido a la evolución continua de las prácticas óptimas y los frameworks de desarrollo.

En este trabajo de grado, desarrollado bajo la modalidad de pasantía, se aborda la migración del componente front-end de AcademiApp. Este procedimiento comprende tres cosas: actualizar el marco Angular a una versión más actualizada, realizar una refactorización de componentes y ajustar el código a las normas vigentes. La pasantía se realizó en el entorno real de desarrollo de Webcloster SAS, incorporando metodologías ágiles como Scrum y herramientas de gestión visual fundamentadas en Kanban. Esto posibilitó un progreso iterativo y controlado del proyecto[2].

Este documento presenta el contexto institucional y técnico de la empresa anteriormente mencionada, el planteamiento del problema que impulsa la migración, el marco teórico y conceptual que fundamenta el desarrollo, así como la metodología empleada, el plan de trabajo, los resultados obtenidos y el impacto del proyecto. El propósito final de este trabajo es ayudar o contribuir a la evolución de la plataforma AcademiApp, fortaleciendo su arquitectura front-end y brindandole valor a la empresa mediante una solución tecnológica actualizada, eficiente y sostenible.

5. Descripción del problema

Las instituciones de educación superior diariamente administran una gran cantidad de informacion que esta relacionada con procesos académicos administrativos y de apoyo institucional. Para contrarestar estas necesidades, la empresa Webcloster SAS llevo a cabo la plataforma Academiapp, una herramienta que integra diversos módulos destinados a la gestión universitaria. No obstante, la versión actual del front-end de la aplicacion desarrollada presenta ciertas limitaciones derivadas del uso de librerias deprecadas dadas por el uso de una versión desactualizada del freamwork Angular, lo que afecta la estabilidad y rendimiento de la aplicación.

El uso del framework obsoleto implica dificultades en la incorporacion de nuevas funcionalidades que en versiones mas estables se podrian llevar a cabo, ademas, tambien hay problemas de incompatibilidad con librerías mas modernas, incrementos de la deuda técnica y mayor exposición a vulnerabilidades de seguridad. Adicionalmente, el codigo ya existente contiene componentes y dependencias que han quedado deprecadas, lo cual dificulta la evolucion del sistema y reduce la eficiencia del equipo de desarrollo al realizar nuevos ajustes o nuevas implemtentaciones.

En cuanto a las funcionalidades disponibles, también se identificaron vacíos importantes. Por ejemplo, la plataforma no contaba con un sistema de pagos en línea, lo que impedía automatizar el proceso de inscripción a eventos pagos. Esto representaba una limitación tanto para la operatividad como para el potencial financiero del sistema, ya que no permitía generar ingresos ni atraer aliados estratégicos o patrocinadores que pudieran aprovechar una infraestructura confiable para gestionar cobros.

A esto se suma que la empresa necesita que AcademiApp mantenga una arquitectura sólida y actualizada para garantizar su continuidad como producto comercial, especialmente porque las instituciones usuarias demandan sistemas estables, rápidos y compatibles con equipos y navegadores modernos. En su estado actual, el front-end no cumple completamente con dichos requerimientos, lo que compromete la calidad del servicio ofrecido por la plataforma.

Por lo anterior, se señala como problema principal la exigencia de migrar y actualizar el front-end de AcademiApp, renovando y mejorando su estructura tecnológica, actualizando su código y alineándolo con prácticas de desarrollo actuales. Resolver este problema es fundamental para asegurar la sostenibilidad del software, optimizar su rendimiento y permitir que la plataforma continúe creciendo y adaptándose a las necesidades del sector educativo.

6. Hipótesis

6.1. Hipótesis General

H1. La migración del front-end de *AcademiApp* a una versión mas reciente del framework Angular. Comparado con la situacion actual, el rendimiento, la mantenibilidad y la capacidad de escalar del sistema se incrementaran considerablemente con la version previa.

6.2. Hipótesis Específicas

H1.1 La actualización del framework Angular disminuirá la deuda técnica y posibilitará la incorporación con librerías y herramientas modernas.

H1.2 El tiempo necesario se reducirá si se refactorizan los componentes que están depreciados. para implementar nuevas funciones y solucionar fallos en la plataforma.

H1.3 La modernización del front-end aumentara la compatibilidad de *AcademiApp* con navegadores, dispositivos y estándares tecnológicos de la actualidad.

H1.4 La implementación de buenas prácticas de desarrollo y una estructura modular mejorará la estabilidad y mantenibilidad del código por parte del equipo de desarrollo.

6.3. Variables

Variable Independiente (VI): Migración del front-end a una versión actualizada de Angular.

Variables Dependientes (VD):

- Mantenibilidad del código.
- Rendimiento del sistema.
- Escalabilidad del software.
- Compatibilidad con tecnologías modernas.
- Nivel de deuda técnica.

7. Formulación del problema

¿De qué manera se puede migrar y actualizar el front-end de AcademiApp a una versión más moderna del framework Angular para optimizar la capacidad de mantenimiento, el rendimiento, la adaptabilidad y la escalabilidad del sistema, disminuir la deuda técnica y facilitar su desarrollo en el futuro?

8. Marco Referencial

8.1. Marco Conceptual

8.1.1. Aplicaciones Web

Son sistemas de software que se pueden acceder mediante un navegador y permiten que los usuarios y los servidores intercambien información y realicen procesos. Las aplicaciones web se han convertido en herramientas imprescindibles para centralizar operaciones, automatizar funciones y aumentar la eficiencia en las instituciones modernas, entre las que se incluyen los centros educativos. Su importancia en aumento se debe a la función estrategica que desempeña el software en los procesos de administracion y produccion. [1].

8.1.2. Gestión Académica y Administrativa

La gestión académica incluye procedimientos como el control de las notas, la matrícula, la programación académica, el seguimiento a los alumnos y la gestión del personal docente. La gestión incluye actividades internas, como la gestión de usuarios, los informes, los trámites digitales y el soporte a nivel institucional. La unificación de estos procedimientos en un solo sistema simplifica la toma de decisiones, disminuye la redundancia de datos y optimiza la calidad del servicio. Por esta razón, las organizaciones necesitan sistemas sólidos y centralizados que aseguren la continuidad, la seguridad y la eficacia en el tratamiento de datos.[2, 5].

8.1.3. Framework Angular

Angular es un marco de desarrollo front-end que se basa en TypeScript y está creado para desarrollar aplicaciones web que sean escalables, mantenibles y estructuradas. Proporciona herramientas como componentes, servicios, gestión de rutas, inyección de dependencias y un conjunto de buenas prácticas estandarizadas.[6].

8.1.4. Migración de Software

La migración es el proceso de actualizar, cambiar o modernizar componentes de un sistema en el proceso de migración tiene como objetivo optimizar su arquitectura, compatibilidad o rendimiento.[7, 8].

8.1.5. Despliegue

Es el proceso de hacer que una aplicación web sea accesible para los usuarios finales. Implica preparar y empaquetar una aplicación para su implementación, ejecutar pruebas para validar la funcionalidad y el rendimiento, optimizar las bases de datos y las estrategias de respaldo, implementar medidas de seguridad sólidas y lanzar la versión en vivo de la aplicación al entorno de producción[19].

8.1.6. Deuda Técnica

La deuda técnica hace referencia a las repercusiones de aplicar soluciones rápidas o que se basen en tecnologías anticuadas, sin terminar tareas pendientes que perjudicarán la calidad del software en el futuro. Entre sus consecuencias están el aumento de la complejidad, los problemas de mantenimiento, los riesgos para la seguridad y las extensas duraciones al implementar nuevas funcionalidades. Una estrategia para disminuir esta deuda es la migración tecnológica.[8].

8.1.7. Rendimiento y Compatibilidad

La optimización de recursos, la rapidez de carga y la fluidez en la interacción están vinculadas con el rendimiento del front-end. La aplicación debe ser capaz de funcionar de manera adecuada en distintos sistemas operativos, dispositivos y navegadores para que sea considerada compatible. Actualizar dependencias, modernizar Angular y refactorizar componentes optimiza los dos ámbitos, lo que potencia la experiencia del usuario final.[6].

8.1.8. Metodologías Ágiles

Scrum es un marco ágil para la gestión de proyectos que organiza el trabajo en ciclos iterativos llamados sprints.[9].

8.1.9. Seguridad

La seguridad hace referencia a la capacidad de un software para seguir funcionando correctamente bajo ataques maliciosos o fallas internas del sistema,[20] garantizando la integridad de la información y operaciones.

9. Marco Teórico

9.1. Ingeniería de Software

La ingeniería de software se enfoca en el estudio de los procesos, métodos y herramientas necesarios para crear sistemas confiables y fáciles de mantener a lo largo del tiempo. Sommerville señala que el desarrollo de software debe seguir una estructura que permita su adaptación y mejora continua [3]. Por su parte, Pressman destaca que una arquitectura bien definida, junto con métodos adecuados y una correcta gestión de los requisitos, resulta clave para asegurar la calidad del sistema [4].

9.2. Sistemas de Información en Entornos Educativos

Los sistemas de información permiten integrar los procesos de una organización y facilitan una toma de decisiones más eficiente. Stair y Reynolds señalan que estos sistemas son especialmente importantes en organizaciones donde existen múltiples procesos que dependen entre sí [5]. En el ámbito académico, Acosta et al. resaltan la importancia de contar con plataformas que centralicen la información y respalden las actividades institucionales [2].

9.3. Framework Angular como Plataforma de Desarrollo

Angular es un framework desarrollado por Google para la construcción de aplicaciones web basadas en componentes. Su documentación oficial resalta las mejoras continuas en rendimiento, seguridad y compatibilidad que incorporan las versiones recientes, advirtiendo que utilizar versiones antiguas incrementa riesgos de compatibilidad y mantenimiento [6].

9.4. Migración y Modernización de Software

La migración de software es el proceso de la actualización o modificación de un sistema con el fin de optimizar su eficiencia, sustentabilidad o compatibilidad. Investigaciones recientes indican que la deuda técnica se incrementa de manera natural en sistemas de larga vida útil, lo cual complica su mantenimiento si no se aplican prácticas de modernización periódica [11, 12]. La refactorización sistemática es clave para asegurar la continuidad del proyecto y permitir su evolución [7].

9.5. Deuda Técnica

La deuda técnica es el precio que se acumula cuando se toman decisiones técnicas imprescindibles de manera tardía o cuando se aplican soluciones rápidas. Atlassian define este concepto como el "costo futuro generado por decisiones tecnológicas apresuradas" y advierte que su aumento tiene un impacto negativo en la calidad y sostenibilidad del software [13]. Según IBM, la deuda técnica encarece los gastos, dificulta el progreso del sistema y complica el mantenimiento [14]. Techopedia añade que la deuda técnica surge cuando el código no sigue buenas prácticas, generando acumulación de problemas que dificultan su manejo [15].

9.6. Rendimiento y Compatibilidad

La compatibilidad garantiza que una aplicación funcione en diversos dispositivos y navegadores, mientras que el rendimiento alude a la velocidad con la que esta responde a los usuarios. La documentación de Angular señala que es esencial tener versiones actualizadas del framework para asegurar que se optimicen el rendimiento y el soporte tecnológico [6].

9.7. Metodologías Ágiles y Scrum

Scrum es un marco ágil que organiza los proyectos en ciclos iterativos denominados *sprints*. Schwaber y Sutherland describen Scrum como un proceso que promueve transparencia, inspección y adaptación continua [9]. Deemer et al. destacan que Scrum facilita la comunicación del equipo y mejora la gestión de tareas en proyectos de desarrollo y migración [2].

9.8. Antecedentes

9.8.1. Q10 Académico

Q10 Académico es un sistema de gestión académica ampliamente utilizado en instituciones de educación superior para la administración de procesos académicos y administrativos. Entre sus principales funcionalidades se encuentran el manejo de matrículas, control académico, gestión de notas, generación de reportes y seguimiento a estudiantes y docentes. Esta plataforma se caracteriza por su enfoque institucional, permitiendo centralizar la información académica y apoyar la toma de decisiones administrativas de manera eficiente.

Figura 1: loguin Q10 Académico.

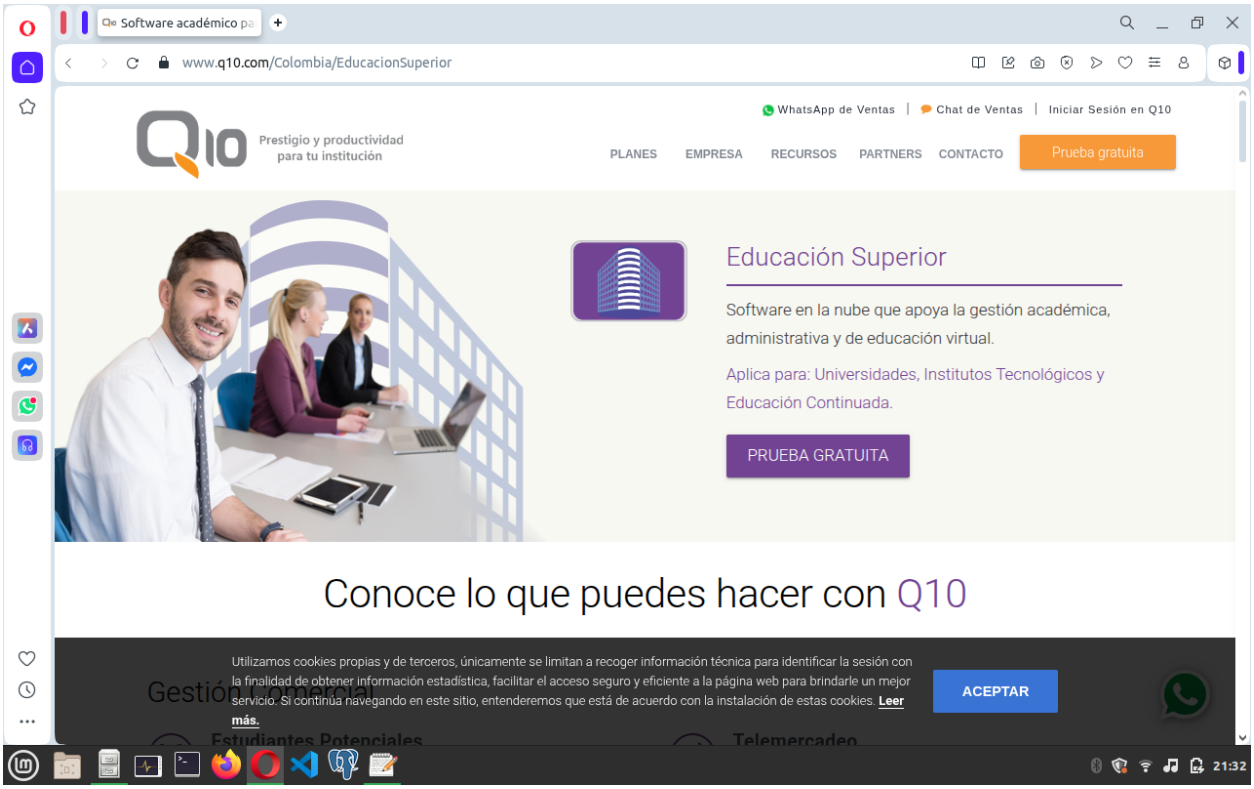

Figura 2: Gestión académica en Q10 Académico.

9.8.2. Sistema SINU

El Sistema SINU es una plataforma académica utilizada por diversas instituciones educativas para la gestión integral de la información estudiantil. Este sistema permite administrar procesos como inscripciones, historial académico, control de cursos, gestión de docentes y generación de informes institucionales. SINU se orienta principalmente a cubrir las necesidades administrativas internas, ofreciendo una solución centralizada para el manejo de datos académicos.

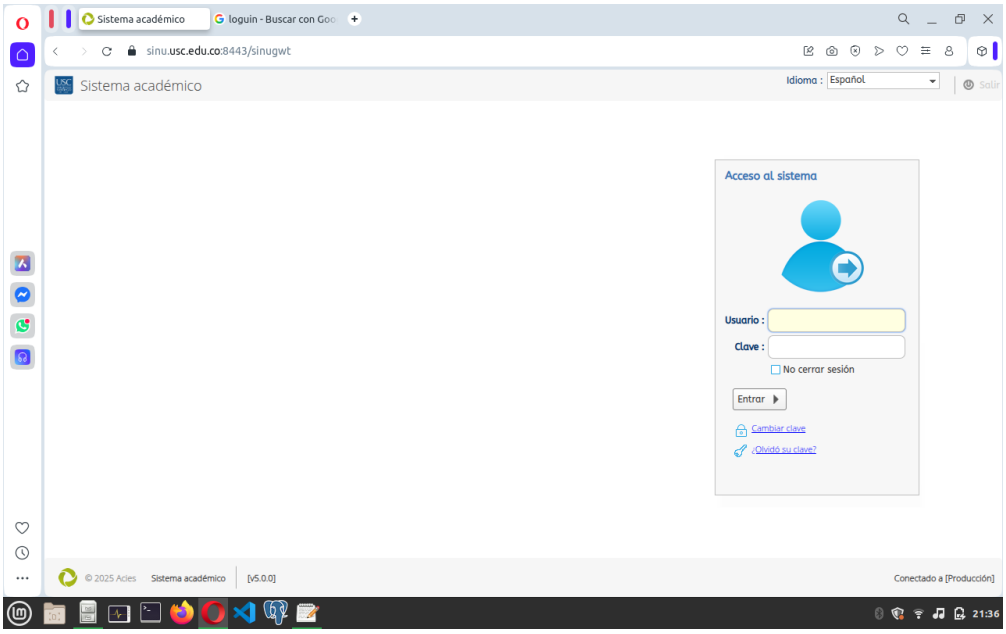

Figura 3: Pantalla loguin del Sistema SINU.

9.8.3. Moodle

Moodle es una plataforma de gestión del aprendizaje (LMS) de código abierto, ampliamente adoptada en el ámbito académico. Su principal objetivo es apoyar los procesos de enseñanza y aprendizaje mediante la creación de cursos virtuales, gestión de contenidos, evaluaciones, foros y seguimiento del desempeño de los estudiantes. Gracias a su carácter flexible y personalizable, Moodle permite a las instituciones adaptar la plataforma a sus necesidades pedagógicas y administrativas.

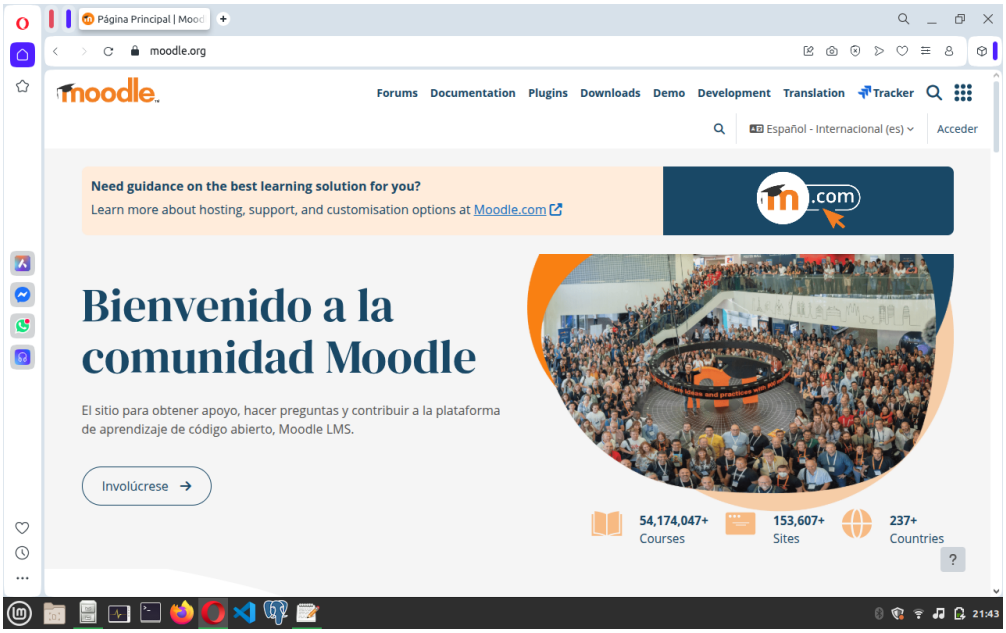

Figura 4: Entorno virtual de aprendizaje Moodle.

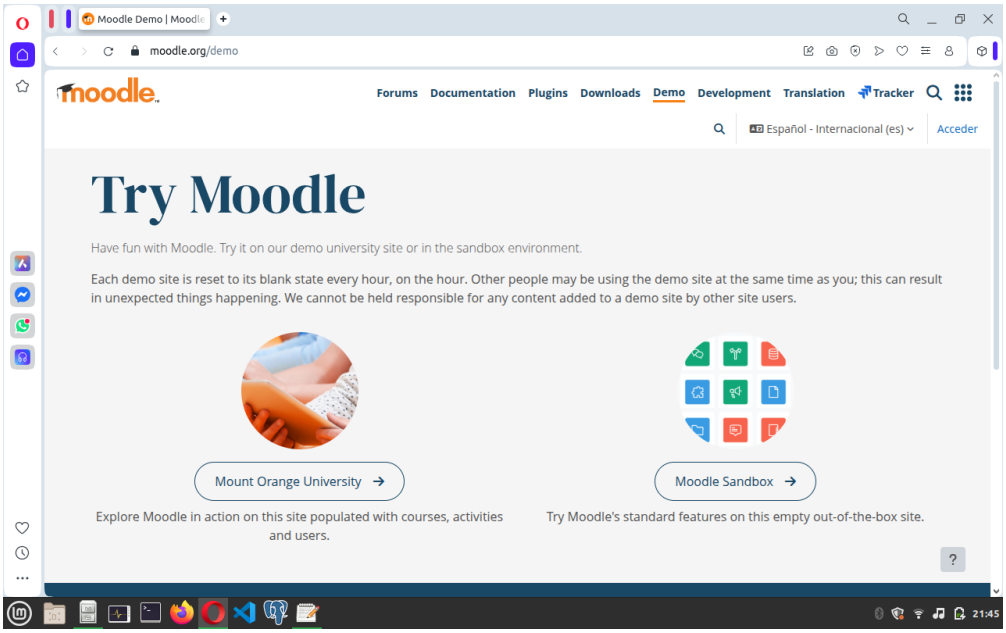

Figura 5: Demo Moodle.

9.8.4. Canvas LMS

Canvas LMS es una plataforma moderna de gestión del aprendizaje utilizada por instituciones educativas a nivel internacional. Se caracteriza por su interfaz intuitiva y su enfoque en la experiencia del usuario, facilitando la gestión de cursos, evaluaciones, contenidos y comunicación entre docentes y estudiantes. Canvas LMS ofrece integración con múltiples

herramientas externas y servicios en la nube, lo que lo convierte en una solución robusta para entornos educativos digitales.

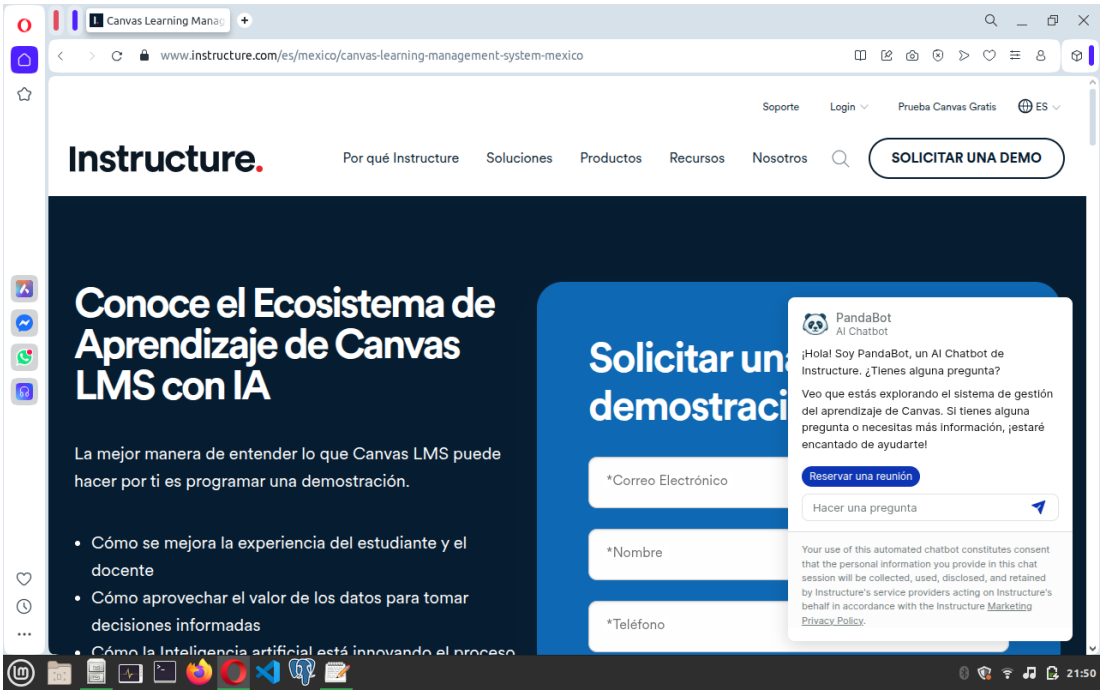

Figura 6: Interfaz principal de Canvas LMS.

9.9. Tabla comparativa de sistemas relacionados

Funcionalidad	**Q**	**S**	**M**	**C**
Gestión de estudiantes	✓	✓	✓	✓
Gestión de docentes	✓	✓	✓	✓
Matrículas y planes de estudio	✓	✓		
Calificaciones	✓	✓	✓	✓
Control de acceso	✓	✓		
Reportes académicos	✓	✓		
Gestión administrativa	✓	✓		
Mensajería interna				✓
Integraciones externas			✓	✓
Actualizaciones frecuentes			✓	✓
Compatibilidad con frameworks modernos				✓

Tabla 1: Comparativa de funcionalidades entre sistemas académicos relacionados.

- Q: Q10 Académico.
- S: Sistema SINU.
- M: Moodle.
- C: Canvas LMS.

Como se observa en la Tabla 1, las plataformas analizadas cubren varias funcionalidades académicas y administrativas; sin embargo, ninguna integra de forma completa ambos enfoques. Esta situación evidencia la necesidad de una solución que unifique estas características y se adapte a los requerimientos de las instituciones educativas.

10. Alcance

10.1. Descripción general

El presente proyecto comprende la **migración y modernización del componente front-end** de la plataforma *AcademiApp* (desarrollada por Webcloster SAS). El objetivo principal es actualizar la versión de Angular utilizada, refactorizar componentes y dependencias mas importantes, y establecer una base de código modular que sea eficiente y mas fácil de mantener. Esto hará mas sencilla la incorporación de nuevas funcionalidades y disminuirá la deuda técnica acumulada [6, 13, 11]. La evaluación y las primeras tareas se basan en el informe técnico interno de migración (de Angular 15 a 16) y en el calendario de actividades elaborado durante la pasantía.

10.2. Alcance funcional (incluye)

- Modificación de la configuración de 'environments' y actualización del marco Angular (se migró a Angular 16). [6].
- Adaptación y refactorización de los componentes que aparecen en el informe técnico (módulos y componentes que se han dado prioridad en el plan de migración).
- Para garantizar la compatibilidad con Angular 16, se requiere la actualización y verificación de librerías externas esenciales (como por ejemplo: ng-bootstrap, ngx-toastr, ag-grid, ngx-translate, etc.).
- Ajuste a componentes *standalone* cuando sea necesario (importación explícita de RouterOutlet y otras directivas) y corrección de errores de enrutamiento. [7, 12]
- Aplicación de buenas prácticas de calidad: separación de la configuración (AppConfig → environments), remoción del código hardcoded, disminución de la deuda técnica en los casos más críticos y refactorización.[2, 8]
- Comprobación de compatibilidad entre diferentes plataformas (navegadores modernos) y pruebas funcionales elementales.

10.3. Alcance no funcional (incluye)

- Mejora de la mantenibilidad (en cuanto a la estructura modular y a la documentación mínima para los módulos que han sido migrados). [2, 7]
- Disminución cuantificable de la deuda técnica en módulos priorizados (consulte la sección de métricas).
- Empleo de la metodología ágil (Scrum) para programar sprints de migración y entregas progresivas. [2]

10.4. Fuera de alcance (no incluye)

- Reescritura completa del producto desde el principio (no se llevará a cabo una reescritura integral de la plataforma entera).
- Migraciones o modificaciones en el back-end (APIs, bases de datos), a excepción de ajustes específicos requeridos para que sea compatible con el front-end.
- Implementación de nuevas funciones importantes no vinculadas a la migración (las cuales se dejarán para etapas posteriores).
- Se llevarán a cabo pruebas básicas en el ambiente de prueba y exhaustivas en infraestructuras de producción.

Actividad / Elemento	**Incluye**	**No incluye**
Migración de Angular 15 a Angular 16	✓	
Actualización de librerías externas (ngx-translate, ng-bootstrap, etc.)	✓	
Refactorización de componentes obsoletos	✓	
Implementación de componentes standalone	✓	
Optimización y reducción de deuda técnica	✓	
Nuevas funcionalidades para AcademiApp		✓
Cambios en backend o base de datos		✓
Reescritura completa de la plataforma		✓
Pruebas de carga o estrés		✓

Tabla 2: Alcance funcional y no funcional del proyecto.

10.5. Entregables

1. Repositorio con la versión de front-end migrada (con una etiqueta o rama identificable) y un registro que documente las modificaciones.

2. Registro de cambios y compatibilidades de bibliotecas (lista de bibliotecas actualizadas y versiones sugeridas).

3. Documentación técnica corta: guía de despliegue (environments), notas de migración y checklist de verificación por módulo.

4. Informe final que registre: las dificultades detectadas, las soluciones implementadas, las métricas comparativas (previas y posteriores) y los consejos para el mantenimiento.

5. Plan de actividades tipo Scrum (con un backlog priorizado, sprints realizados y aprendizajes adquiridos). [2]

Entregable	**Descripción**	**Responsable**
Código migrado a Angular 16	Proyecto ejecutable, con ajustes en routing, componentes y librerías.	Pasante
Registro de versiones de librerías	Documento con versiones actualizadas y compatibilidades.	Pasante
Documentación técnica	Notas de migración, guía de build, lista de cambios implementados.	Pasante
Checklist de verificación	Validación funcional básica por módulo.	Pasante / Webcloster
Informe final de trabajo	Documento académico con análisis, resultados y recomendaciones.	Pasante

Tabla 3: Entregables del proyecto de migración.

10.6. Criterios de aceptación / métricas

Los criterios para considerar la migración aceptada incluirán, entre otros:

- **Compilación exitosa** del front-end en configuración de `production` sin fallos graves. (consultar la guía de actualización de Angular). [6]
- **Pruebas funcionales** (smoke tests) realizadas en los módulos seleccionados como prioritarios (lista fundamentada en un informe interno).
- **Reducción de deuda técnica** en módulos priorizados: Disminución de la cantidad de problemas críticos o çode smells"que se reportan (objetivo: una reducción relativa del 30 % o más en el conjunto priorizado; este objetivo puede ser modificado dependiendo de los resultados). Según investigaciones, medir y priorizar la deuda técnica mejora el mantenimiento y el lead time. [13, 11]
- **Documentación mínima** disponible para despliegue y mantenimiento (variables de entorno, endpoints y pasos de build).

Criterio	**Método de verificación**	**Condición de aceptación**
Compilación en modo producción	Ejecución de `ng build -configuration production`.	Sin errores críticos.
Compatibilidad funcional del front-end	Pruebas básicas en módulos priorizados.	Todas las rutas y vistas funcionales.
Reducción de deuda técnica	Revisión de code smells y warnings.	Reducción 30 % en módulos migrados.
Compatibilidad de librerías externas	Instalación y pruebas de funcionamiento.	Todas las dependencias actualizadas.
Documentación técnica mínima	Revisión del entregable técnico.	Aprobación del supervisor.

Tabla 4: Criterios de aceptación del proyecto.

10.7. Supuestos y restricciones

- El equipo de Webcloster SAS proveerá acceso al repositorio y a entornos de prueba necesarios para la verificación.
- Las APIs del back-end permanecerán estables durante la ventana de pruebas; cualquier cambio de contrato será gestionado fuera del alcance de la migración o como tarea adicional negociada.
- Se parte de la base de que la migración será incremental y priorizada, aprovechando el soporte para adopción gradual de componentes *standalone* en Angular. [7, 12]

10.8. Riesgos principales

- Incompatibilidades de bibliotecas de terceros que necesiten una reescritura parcial. (Mitigación: plan de reversión por versión y pruebas de compatibilidad).
- Se ha documentado en investigaciones sobre la modernización y la deuda técnica que el lead time se incrementa temporalmente en los trabajos de desarrollo debido a la curva de aprendizaje de nuevas APIs. [13, 11]
- Posibles retrocesos debido a alteraciones en rutas, dependencias o comportamiento del DOM; se pueden mitigar con pruebas incrementales y revisiones de código. [6, 2]

10.9. Plan / Cronograma (resumen)

El proyecto se plantea en fases iterativas (Scrum):

1. **Diagnóstico y priorización de módulos** (inventario y plan de migración).
2. **Actualización de entorno y librerías críticas** (entorno de desarrollo y CI). [6]
3. **Migración por bloques** (sprints por áreas funcionales: seguridad, usuarios, estudiantes, docentes, reportes).
4. **Pruebas e integración** (smoke tests, verificación cross-browser).
5. **Entrega e informe final** (documentación, checklist y recomendaciones).

Observación: la estrategia incremental (migración por bloques y uso de componentes *standalone*) permite minimizar el riesgo de interrupciones en producción y facilita la validación continua [7, 12].

10.10. Objetivos

Actualizar la versión de Angular empleada en el frontend de la aplicación Academiapp de Webcloster SAS, garantizando un código actualizado, compatible y operativo conforme a las normas fijadas por la compañía.

10.10.1. Objetivos específicos

- Analizar el estado actual del *front-end* de la plataforma AcademiApp, identificando los principales requerimientos técnicos necesarios para realizar la migración a una versión más reciente del framework Angular.
- Definir y ejecutar un plan de migración que permita actualizar el framework Angular de forma ordenada, asegurando la compatibilidad de las dependencias y la continuidad del funcionamiento de la plataforma.
- Refactorizar y migrar los componentes, módulos y servicios principales del sistema, aplicando buenas prácticas de desarrollo para mejorar la organización del código y su mantenibilidad.
- Evaluar el correcto funcionamiento de la plataforma migrada mediante pruebas funcionales básicas, con el fin de verificar la estabilidad, el rendimiento y la correcta operación de las funcionalidades esenciales.

11. Metodología

La metodología utilizada para el desarrollo del proyecto de migración y modernización del front-end de AcademiApp fue llevado a cabo con un método ágil, que fusionó practicas de **Scrum** y **Kanban** mediante el enfoque **ScrumBan**. Esta perspectiva permitió una administración que se puede modificar, iterar y aumentar, adecuándose a las exigencias técnicas y al ritmo de actualización de los módulos priorizados.

Se organizaron etapas de trabajo breves en las que se abordaron actividades puntuales, tales como la actualización de dependencias, la reestructuración de componentes, la revisión de bibliotecas externas, la verificación de la compatibilidad con Angular 16 y la realización de pruebas funcionales esenciales. Cada una de estas etapas comprendió procesos de planificación, ejecución, análisis técnico y registro detallado de los avances obtenidos.

11.1. Gestión del proyecto

La metodología ágil **ScrumBan** se utilizó para administrar el proyecto, y es conocida por su capacidad para amalgamar la estructura de Scrum con la versatilidad de Kanban. Se realizaron encuentros de forma regular con el fin de programar las tareas, hacer un seguimiento del progreso y llevar a cabo retrospectivas al término de cada ciclo de desarrollo.

Estas reuniones tuvieron como objetivos:

- Analizar el estado de los módulos priorizados.
- Identificar dependencias críticas que debían actualizarse.
- Validar avances con el Product Manager.
- Ajustar prioridades según la retroalimentación del equipo.

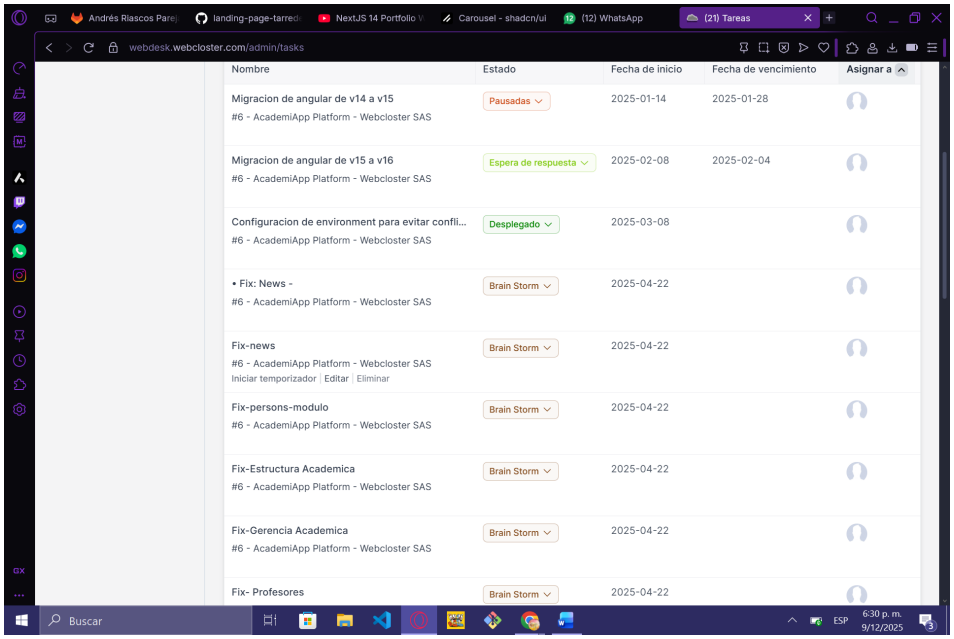

Figura 7: Reunión de seguimiento del equipo técnico durante el desarrollo del proyecto.

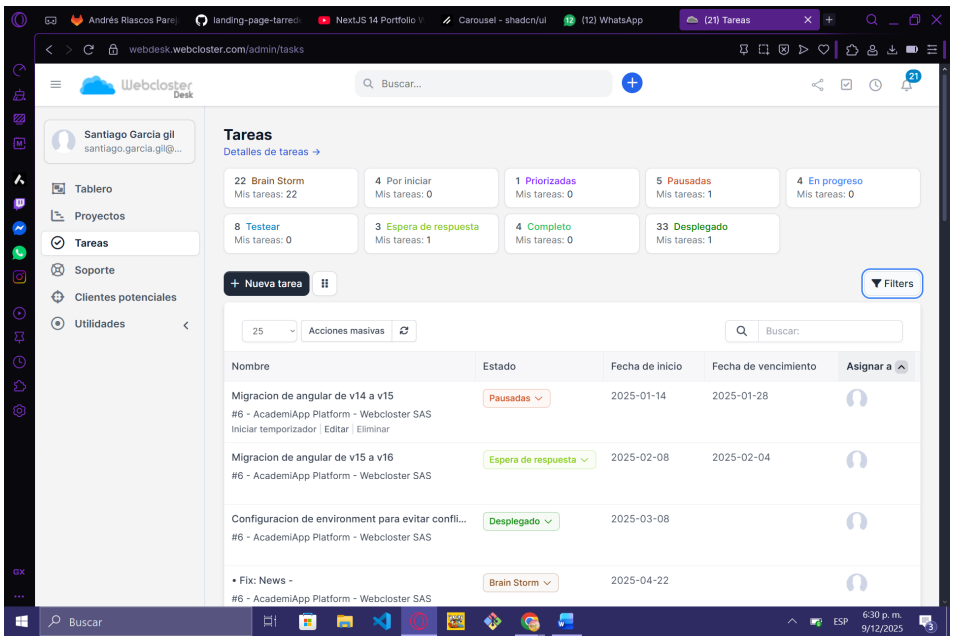

Figura 8: Sesión de revisión y validación de avances realizada en Webcloster SAS.

11.1.1. Tareas y responsabilidades

La gestión operativa del proyecto se centralizó en **WebDesk**, una plataforma interna de Webcloster SAS que implementa una metodología de tablero Kanban para la administración de tareas. Las actividades se clasificaron según su estado, abarcando categorías como *Pendiente de arranque*, *En desarrollo*, *Suspendidas* y *Completadas (Desplegadas)*. Cada ítem de trabajo registraba datos relevantes, tales como su nivel de prioridad, el tiempo estimado frente al tiempo real consumido y una descripción técnica detallada.

Adicionalmente, **GitLab** funcionó como el sistema principal de control de versiones, permitiendo realizar una gestión rigurosa y ordenada del código fuente. Se emplearon ramas específicas para la migración, se generaron solicitudes de fusión (*merge requests*) y se mantuvo un registro minucioso y auditable de todas las modificaciones efectuadas (*commits*).

Con el propósito de asegurar una comunicación fluida y una coordinación eficiente del equipo, se estableció un canal en **Telegram**. Este espacio permitió la difusión inmediata de directrices técnicas, la validación oportuna de los avances logrados y la resolución ágil de inquietudes o incidencias surgidas durante el proceso.

Las etapas específicas que estructuraron el flujo de tareas fueron:

- **Pendiente de arranque**
- **En curso**
- **Suspendidas**
- **Esperando retroalimentación**
- **Completadas y en Producción (Desplegadas)**

La coordinación interna del equipo se apoyó en el chat de Telegram, lo que garantizó la sincronización en tiempo real, la comunicación de pautas técnicas y la respuesta inmediata a cualquier consulta durante el proceso de migración.

Finalmente, el uso de **GitLab** fue fundamental para la administración del código fuente. En esta plataforma se gestionaron las ramas dedicadas al proyecto, se ejecutaron los procesos de *merge requests*, se dejó constancia de las modificaciones y se validaron los ajustes en conjunto con el equipo técnico.

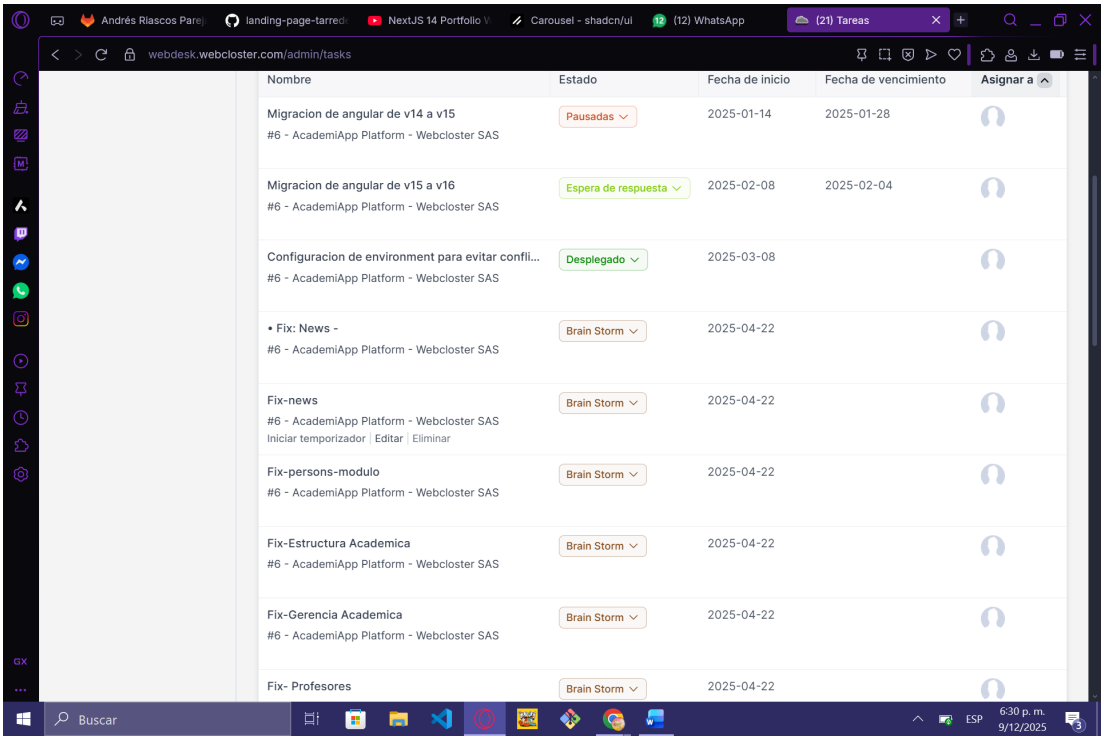

Figura 9: Listado de tareas gestionadas en WebDesk durante la migración.

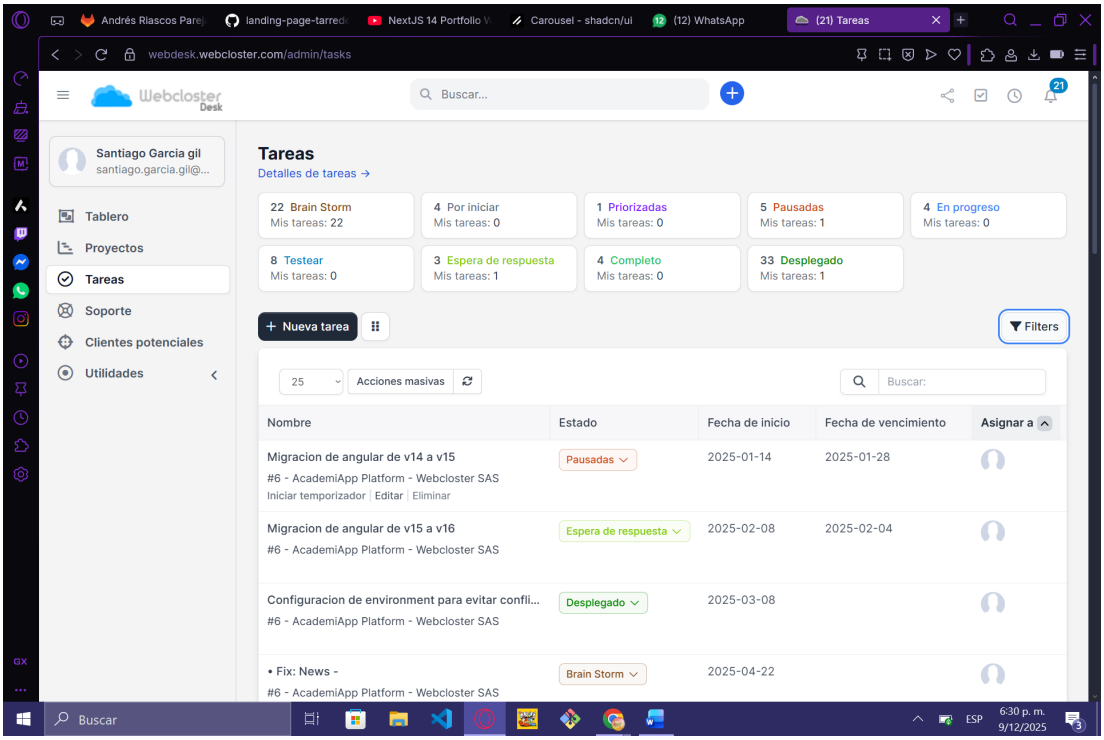

Figura 10: Registro de tiempo y progreso por tarea dentro de WebDesk.

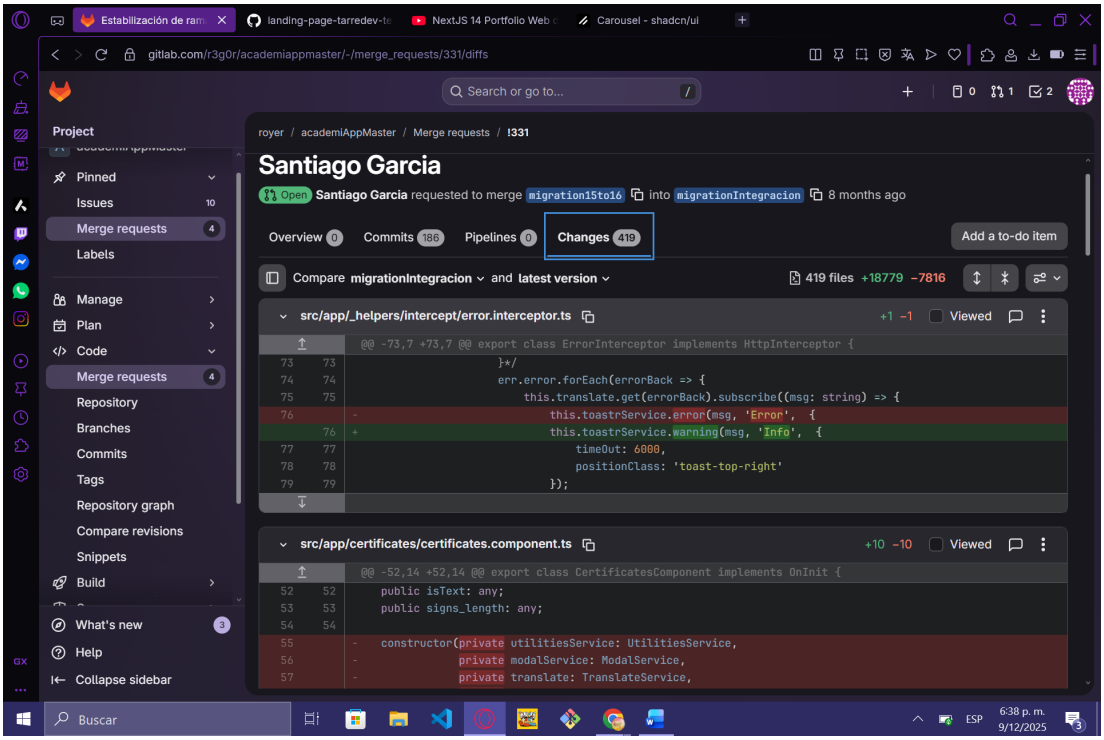

Figura 11: Merge Request en GitLab con los cambios asociados a la migración.

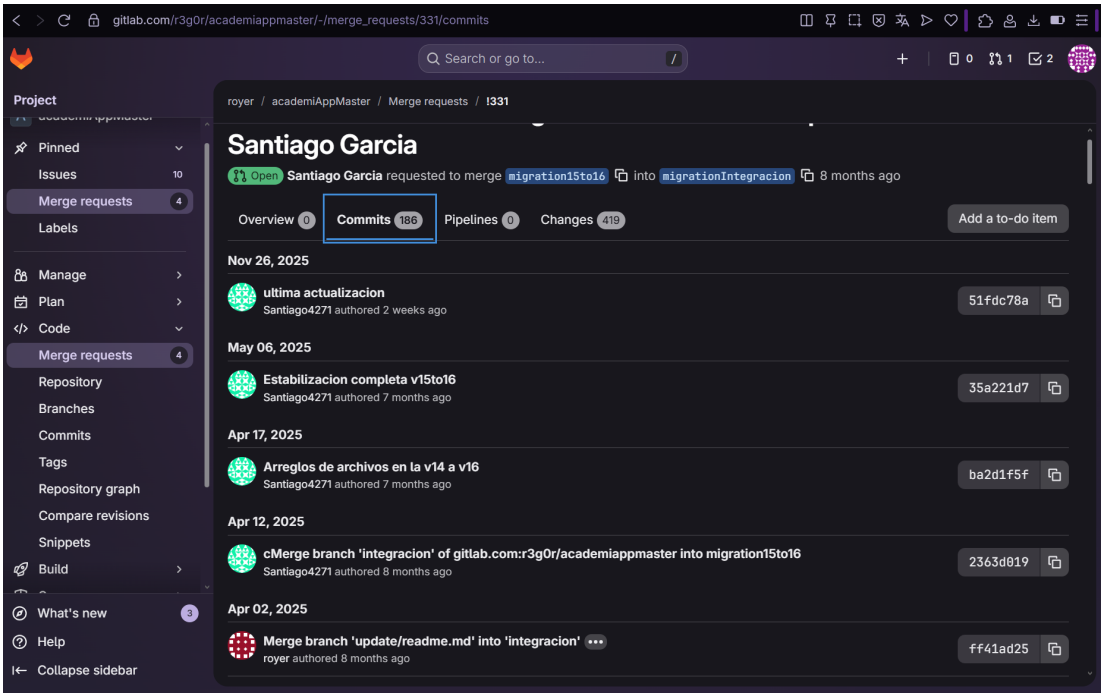

Figura 12: Historial de commits registrado durante la actualización del front-end.

11.1.2. Planificación del proyecto

Se llevó a cabo la planificación del proyecto de manera iterativa, configurando ciclos de trabajo en función de la prioridad y la complejidad de los módulos. Cada tarea contenía:

- Fecha de inicio.
- Fecha estimada de finalización.
- Tiempo efectivo trabajado.
- Responsable asignado.
- Estado del avance.

Dentro de WebDesk se cargaron las tareas correspondientes a:

- Migración de Angular 14 a 15.
- Migración de Angular 15 a 16.
- Configuración de *environment* para evitar conflictos.
- Corrección de módulos como: *news*, *persons*, *estructura académica*, *professors*, entre otros.

Cada etapa del trabajo fue corroborada con el Product Manager y el equipo técnico de Webcloster SAS, lo que garantizó que la migración estuviera a la altura de los estándares internos en cuanto a calidad y compatibilidad del sistema AcademiApp.

11.1.3. Herramientas utilizadas

La gestión de la migración, la depuración y el control del proyecto demandó una gama de instrumentos técnicos durante su desarrollo:

- **Angular 15 y 16:** Framework principal utilizado para la construcción y migración del front-end.
- **Node.js y NPM:** Ambiente para gestionar dependencias y ejecutar scripts del proyecto.
- **GitLab:** Control de versiones, manejo de ramas, *merge requests* y revisión de código.
- **WebDesk:** Plataforma para gestionar tareas, tiempos y seguimiento del proyecto.
- **Telegram:** Canal de comunicación de soporte y coordinación interna.
- **VS Code:** Editor principal para desarrollo, refactorización y pruebas.

- **Navigadores con DevTools (Chrome, Edge):** Pruebas, depuración y validación del comportamiento del front-end.
- **Documentación oficial de Angular:** Fuente primaria de referencia para cambios, versiones, APIs y buenas prácticas.

12. desarrollo de proyecto

El desarrollo del proyecto se ejecutó en el entorno productivo de Webcloster SAS y estuvo orientado a la modernización del componente front-end de la plataforma AcademiApp. Este proceso contempló la migración del framework Angular desde la versión 14 hacia Angular 16, la actualización de dependencias internas y externas, la refactorización de módulos priorizados y la validación del sistema mediante pruebas funcionales. La ejecución del proyecto se llevó a cabo bajo una metodología ágil, utilizando GitLab para el control de versiones y WebDesk como herramienta principal de gestión de tareas.

La necesidad de modernizar el *front-end* se presentó en respuesta a los progresos la evolución del marco Angular y la estabilidad de las tecnologías web, que introduce avances importantes en estabilidad, rendimiento y mantenibilidad En sus versiones más actuales. En este contexto, el proyecto abordó la migración tendrá en cuenta el cambio más importante desde Angular 14 hasta Angular 16. fue expuesto durante la transición entre Angular 15 y Angular 16.

A lo largo de la pasantía, se realizaron actividades de análisis, implementación, verificación, documentación y comunicación continua con el equipo técnico de Webcloster SAS, garantizando un flujo de trabajo organizado y un producto final estable y compatible con tecnologías recientes.

12.1. Análisis del estado inicial del proyecto

La primera tarea del proyecto consistió en un análisis exhaustivo de la situación inicial del componente front-end de AcademiApp. Para entender la arquitectura que ya existe, reconocer los peligros técnicos y establecer el verdadero alcance de la migración, este período fue esencial.

El sistema estaba desarrollado con Angular 14, utilizando una base de código. eficaz, pero con claros indicios de que la tecnología está obsoleta, como por ejemplo componentes con complejidad ciclomática elevada, dependencias que no se actualizan y estructuras que complicaban la conservación a largo plazo.

El diagnóstico permitió establecer qué partes del sistema serían priorizadas durante la migración y qué elementos debían ser refactorizados para ajustarse a las nuevas prácticas del framework.

12.2. Planeación del proceso de la migración

Tras la evaluación del estado inicial del sistema, se determinó una estrategia técnica para migrar el marco Angular desde su versión 14 hasta la versión dieciséis. Debido a que los cambios entre Angular 14 y la versión 15 fueron escasos, La mayor parte del esfuerzo se dedicó a la migración de Angular 15 a Angular 16.

La planificación no se restringió solo a determinar las etapas técnicas de la actualización, sino que además incluyó elementos de organización, como la sincronización con el equipo de desarrollo, la priorización de tareas y la Gestión de tiempos durante el periodo de las prácticas. Ya que la plataforma se encuentra en uso activo, fue necesario definir una táctica que habilitara Hacer modificaciones sin perjudicar la operación usual del sistema.

Además, se examinaron los posibles escenarios de error relacionados con la actualización del marco de trabajo, con la inclusión de incompatibilidades en las bibliotecas, comportamientos imprevistos durante la ejecución y errores de compilación. Este análisis anterior posibilitó prever soluciones y determinar acciones. correctivas previas al comienzo de la migración.

12.2.1. Objetivos técnicos de la migración

Los objetivos técnicos definidos para esta fase fueron:

- Garantizar la compatibilidad del proyecto con Angular 16.
- Reducir la deuda técnica existente.
- Mejorar la mantenibilidad del código.
- Optimizar la organización de módulos y componentes.
- Minimizar el impacto visual en la interfaz del usuario final.

12.2.2. Alcance técnico de la migración

Con el objetivo de delimitar con precisión el ámbito técnico del proyecto, Enfocar los esfuerzos en los elementos esenciales del sistema y garantizar conclusiones específicas durante el tiempo disponible para la pasantía. Se estableció que la migración se centraría fundamentalmente en la modernización tecnológica del marco Angular, sin hacer cambios relevantes en la Interfaz gráfica del sistema.

Esta resolución se tomó porque el propósito principal del proyecto no era el modificación estética de la plataforma, sino el aumento de su estabilidad y eficiencia y su capacidad de mantenimiento. Así, se aseguró que los usuarios finales no experimentarán alteraciones drásticas en la manera de interactuar con el sistema, a diferencia de La estructura del front-end se reforzaba a nivel interno.

La revisión de las configuraciones del proyecto también formó parte del alcance, archivos de entorno y configuraciones requeridas para garantizar la compatibilidad con las ediciones más actuales de las herramientas de desarrollo empleadas por la compañía.

12.3. Gestión de dependencias y librerías externas

Una de las tareas más importantes durante la migración fue el análisis y Renovación de las dependencias externas empleadas en el proyecto. Algunas bibliotecas no ofrecían soporte inmediato para Angular 16, lo que necesitó análisis adicionales y modificaciones manuales.

La Tabla 5 presenta un resumen general del estado de las dependencias antes y después de la migración.

Tabla 5: Estado de dependencias antes y después de la migración

Dependencia	**Versión Inicial**	**Versión Final**	**Estado**
Angular Core	14.x	16.x	Actualizada
Angular Material	14.x	16.x	Actualizada
RxJS	7.5	7.8	Compatible
TypeScript	4.x	5.x	Actualizada
Librerías internas	Varias	Ajustadas	Refactorizadas

12.4. Proceso de actualización del framework Angular

La migración del framework se llevó a cabo de manera gradual, siguiendo los sugerencias oficiales de Angular. El procedimiento comprendió la realización de comandos particulares, la solución de conflictos que tengan que ver con dependencias y la inspección continua para asegurar el funcionamiento adecuado del sistema.

12.4.1. Comandos principales utilizados

```
ng update @angular/core
ng update @angular/cli
ng update @angular/material
```

Estos comandos permitieron automatizar gran parte del proceso, aunque fue necesaria la intervención manual en varios archivos de configuración.

12.5. Refactorización y mejora del código

La refactorización del código fue tratada como una actividad estratégica con el objetivo de optimizar la calidad interna del sistema sin modificar su funcionamiento exterior. Este proceso fue particularmente significativo teniendo en cuenta la antigüedad de ciertos módulos y el incremento gradual de deuda técnica con el paso del tiempo.

Con el objetivo de facilitar su lectura, se llevó a cabo la revisión de componentes complejos, la simplificación de estructuras condicionales largas y la reestructuración de la lógica interna de algunos servicios durante el proceso de refactorización. También se eliminaron partes de código que eran redundantes o que ya no se usaban y se hicieron adecuaciones a las convenciones de nomenclatura.

Estas medidas ayudaron a obtener un código más limpio, fácil de mantener y acorde con las mejores prácticas de la ingeniería del software en la actualidad, lo que simplificará el mantenimiento y la evolución del sistema en el futuro.

12.6. Pruebas funcionales y validación

Se llevaron a cabo pruebas funcionales con la finalidad de comprobar que el sistema continuara su adecuado funcionamiento después de la migración y reestructuración del código de programación. Se decidió llevar a cabo pruebas funcionales, considerando la magnitud del proyecto. Fundamentales que se enfocan en los flujos esenciales empleados por los usuarios.

La validación del acceso al sistema, la navegación entre módulos, el correcto almacenamiento de vistas y la comprobación del funcionamiento general de los elementos migrados. Cualquier irregularidad que se identificó en esta etapa fue examinada y corregida antes de

finalizar el proceso.

La realización de estas pruebas posibilitó corroborar que la migración no introdujo regresiones funcionales y que el sistema estaba en un estado estable para su aplicación en el ámbito de la producción.

12.7. Documentación y cierre del proceso

Se documentó de manera minuciosa el procedimiento de, como la etapa final del proyecto. migración, que incluye las decisiones técnicas tomadas y las modificaciones efectuadas y las sugerencias para las próximas etapas de modernización del sistema.

Este material se preparó con la finalidad de funcionar como referencia para el Equipo de desarrollo de Webcloster SAS, para ayudar a que el proceso continúe. de actualización tecnológica, así como disminuir la curva de aprendizaje para los recién llegados a programadores que participen en el proyecto.

Se realizó por finalizar el proyecto de pasantía, alcanzando los objetivos establecidos y contribuyendo con valor técnico a la plataforma AcademiApp.

13. Resultados Obtenidos

La migración exitosa y la modernización parcial del elemento *front-end* de la plataforma AcademiApp se llevaron a cabo gracias al desarrollo del proyecto de pasantía, satisfaciendo así las metas establecidas y ofreciendo importantes mejoras en términos técnicos, estructurales y de compatibilidad tecnológica.

Los resultados conseguidos no se reflejaron sobre todo en un rediseño visual integral del sistema, sino en la modernización de su base tecnológica, la estabilización del ambiente de desarrollo y la adecuación de la aplicación para etapas posteriores de evolución. Los resultados principales obtenidos durante la realización del proyecto se detallan a continuación.

13.1. Migración exitosa del framework Angular

La renovación del framework Angular desde la versión 14 hasta la 16 fue uno de los logros más significativos del proyecto. Este proceso posibilitó que el front-end de AcademiApp se sincronizara con una versión más nueva y estable del marco, asegurando su compatibilidad con las herramientas y bibliotecas contemporáneas empleadas en el ecosistema Angular.

Aunque la transición de Angular 14 a 15 no conllevó cambios relevantes, la actualización a Angular 16 supuso modificaciones técnicas significativas, como el cambio de configuraciones internas, la actualización de dependencias y la corrección de advertencias vinculadas con funcionalidades ya obsoletas. Dado que En consecuencia, el proyecto pudo compilar y ejecutarse de manera adecuada con la nueva versión del framework, sin que la funcionalidad normal de la plataforma se viera afectada.

13.2. Actualización y mejora de la sección de noticias

Se hicieron modificaciones particulares en la sección de noticias (*News*), uno de los módulos más importantes del sistema, durante el proceso migratorio. Esta sección necesitó ajustes a nivel de código y de compatibilidad con las versiones más recientes del marco y sus dependencias.

La sección de noticias se volvió completamente funcional bajo Angular 16 gracias a estos cambios, eliminando las advertencias de obsolescencia y aumentando la estabilidad de los componentes implicados. Asimismo, se hicieron ligeros cambios visuales que ayudaron a una presentación de la información más coherente, conservando la experiencia del usuario previamente establecida. Estas modificaciones hicieron posible que el módulo siga siendo mantenible y ampliable en etapas posteriores de desarrollo.

La visualización de una noticia publicada después de finalizar el proceso de registro se puede observar en la Figura **??**, mientras que el formulario empleado para la creación de noticias dentro del sistema se exhibe en la Figura **??**.

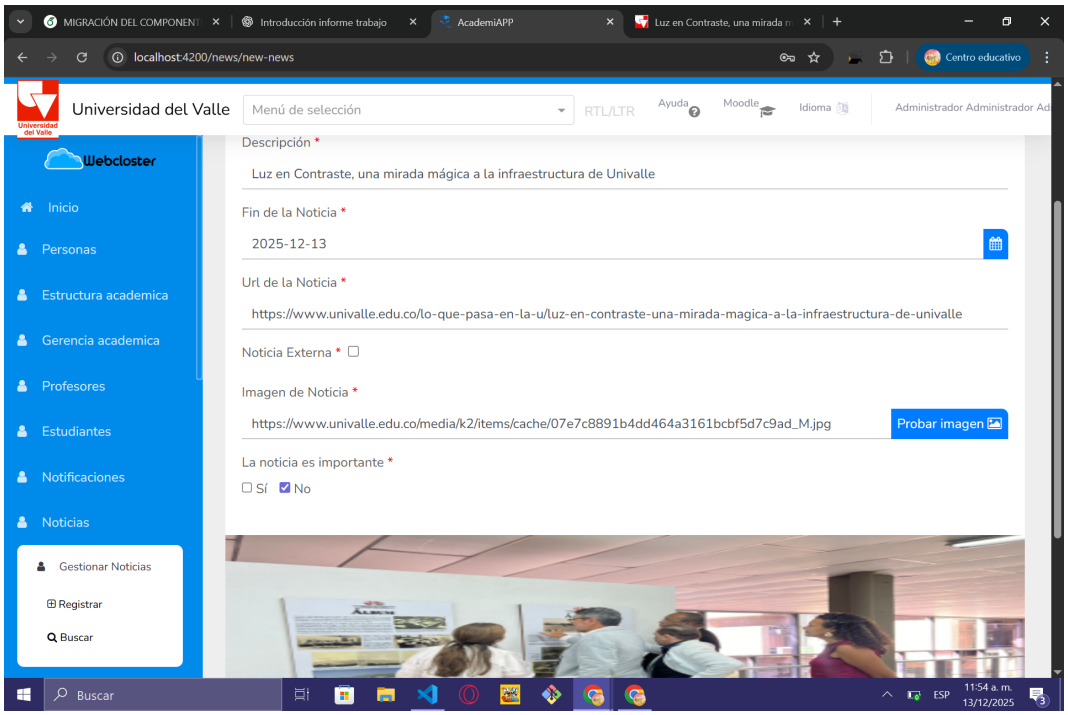

Figura 13: Formulario de creación de noticias en AcademiApp

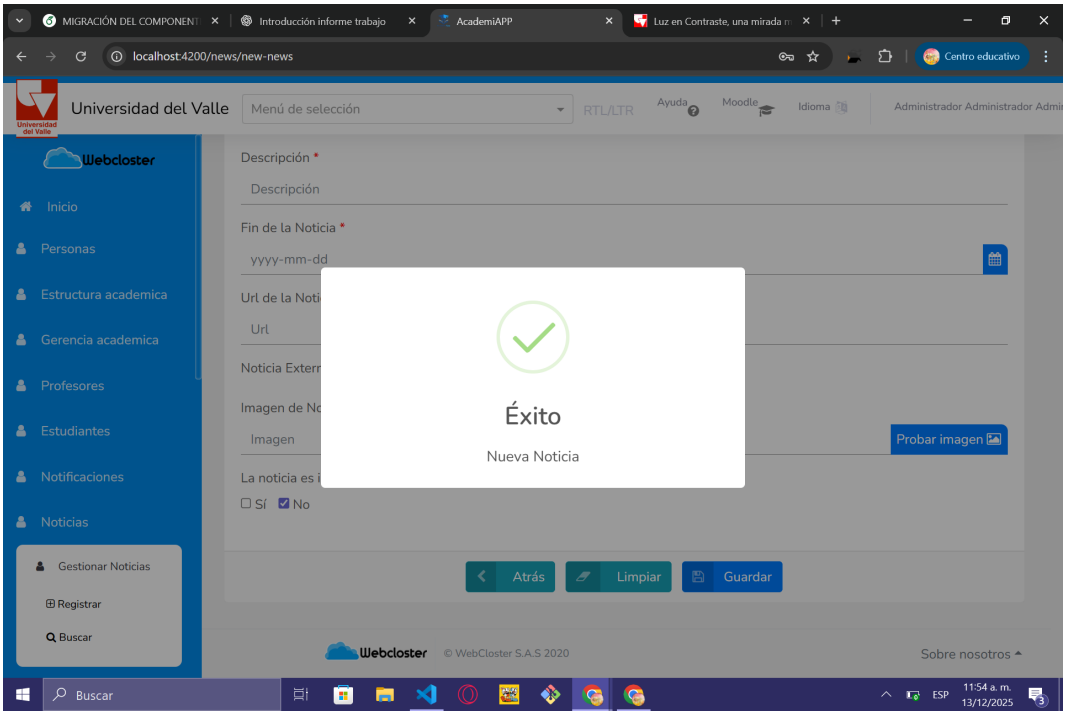

Figura 14: Visualización de una noticia publicada en AcademiApp

13.3. Ajustes visuales y funcionales en el Dashboard

Después de que se actualizó el marco, se notaron pequeñas alteraciones visuales en el *Dashboard* de la aplicación, sobre todo relacionadas con modificaciones en los estilos predeterminados y en la forma de algunos elementos de Angular Material.

Aunque estas modificaciones no supusieron un cambio importante en la interfaz de usuario, sí significaron un avance en cuanto a la coherencia visual y a la adaptación del diseño a las normas vigentes del marco de trabajo. El tablero de control continuó funcionando como originalmente, asegurándose de que el proceso de migración no perjudicara a los usuarios finales.

Como consecuencia, se logró una vista principal más estable y en línea con las versiones recientes de Angular, sin que fuera necesario un rediseño total.

En la Figura Figura 15 se observa el Dashboard de AcademiApp después de la migración, donde se reflejan los ajustes visuales realizados y la correcta integración del módulo de noticias actualizado.

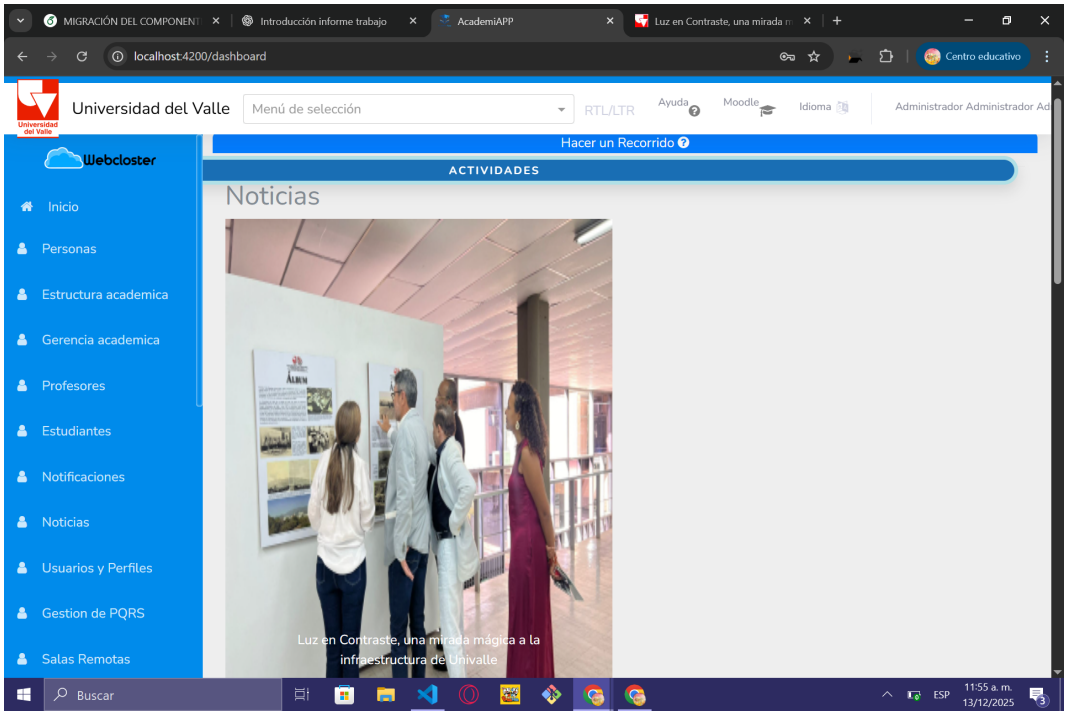

Figura 15: Dashboard de AcademiApp tras la migración a Angular 16

13.4. Actualización de componentes basados en calendario

La modernización de las partes del sistema que empleaban elementos de calendario, especialmente las que se fundamentaban en bibliotecas externas como *Angular Calendar*, fue otro hallazgo significativo del proyecto. Dado que estas bibliotecas tenían funciones obsoletas o incompatibles con Angular 16, era necesario revisarlas y modificarlas.

Se hicieron las modificaciones necesarias en el código y se actualizaron esas dependencias durante la migración para garantizar que funcionaran correctamente. En consecuencia, las vistas que incluyen calendarios pudieron funcionar sin errores de compilación ni advertencias de depreciación, lo que permitió una mayor estabilidad del sistema en general.

Esta actualización posibilitó que el proyecto quedara también listo para optimizaciones futuras en los módulos que administran programación académica y eventos.

13.5. Reducción de advertencias y elementos deprecados

Se consiguió disminuir de manera significativa el número de advertencias producidas a lo largo de la compilación del proyecto gracias a la migración y refactorización. La utilización de APIs obsoletas, configuraciones anticuadas y dependencias sin soporte fue el tema de muchas de estas alertas. Para las versiones más actuales de Angular.

La supresión de estos componentes propició un ambiente de desarrollo más limpio y controlado, lo que favoreció la identificación de fallos reales y optimizó la experiencia del grupo encargado del desarrollo. Este resultado es particularmente significativo para la viabilidad del proyecto en el largo plazo.

13.6. Mejora en la mantenibilidad y preparación para futuras fases

Para concluir, la optimización de la mantenibilidad general del front-end de AcademiApp fue uno de los hallazgos más relevantes del proyecto. La actualización del marco y la refactorización de componentes priorizados lograron que se tuviera una base de código más clara, ordenada y coherente con las buenas prácticas de ingeniería de software.

La plataforma se ha quedado lista para etapas de modernización futuras, como la introducción de nuevas funcionalidades, la optimización del rendimiento o la puesta en marcha de pruebas automatizadas, gracias a estas modificaciones. Así, el proyecto no solamente atendió requerimientos inmediatos, sino que también brindó a la compañía Webcloster SAS un valor estratégico a largo plazo.

13.7. Recomendaciones

Con base en los resultados obtenidos durante la modernización y migración del componente *front-end* de la plataforma AcademiApp, así como en el proyecto de pasantía, a continuación se presentan una serie de sugerencias que pueden ser útiles para las siguientes etapas de mantenimiento y desarrollo del sistema.

Se aconseja, ante todo, seguir actualizando de manera gradual los otros módulos de la plataforma que no se trataron en esta fase de la pasantía. A pesar de que los elementos fundamentales se mantuvieron operativos bajo Angular 16, todavía hay partes del sistema que podrían beneficiarse de procesos parecidos de modernización y refactorización, con el propósito de disminuir la deuda técnica acumulada y optimizar la calidad general del código.

También se recomienda poner en marcha una serie de pruebas automatizadas, unitarias e integradas, que hagan posible la identificación temprana de fallos ante posibles mejoras o actualizaciones del marco. Incluir estas pruebas haría más fácil el mantenimiento del sistema y disminuiría la posibilidad de incluir errores en producción, particularmente en un proyecto que está en continua evolución como AcademiApp.

Otra sugerencia relevante es llevar a cabo un monitoreo regular de las bibliotecas y entidades externas empleadas en el proyecto. Es crucial examinar regularmente el estado de estas dependencias para prevenir la utilización de librerías obsoletas o sin soporte durante mucho tiempo, ya que esto podría causar problemas de compatibilidad en versiones posteriores. Esto es especialmente importante porque el ecosistema Angular está en constante evolución.

Desde la perspectiva de la experiencia y el diseño del usuario, se aconseja considerar en fases futuras la opción de llevar a cabo mejoras visuales más significativas en la interfaz del sistema. Una revisión del diseño podría ayudar a que la experiencia de los usuarios sea más moderna y consistente, en particular utilizando las posibilidades actuales de Angular Material, aunque no se priorizó un rediseño estético para este proyecto.

Además, se recomienda robustecer la documentación técnica del proyecto, añadiendo instrucciones precisas sobre cómo debe estar estructurado el código, las configuraciones principales y los procedimientos de despliegue. La actualización de la documentación haría más fácil que nuevos desarrolladores se unieran al equipo y ayudaría a entender más rápido cómo funciona la plataforma internamente.

Por último, se aconseja continuar utilizando herramientas de colaboración como WebDesk y GitLab, así como metodologías ágiles para administrar el proyecto. Estas prácticas fueron efectivas a lo largo de las pasantías, posibilitando un seguimiento nítido de las labores, una comunicación más fluida entre los integrantes del equipo y una supervisión apropiada de las modificaciones hechas al código fuente.

El uso de estas sugerencias permitirá que AcademiApp siga progresando de una forma organizada, sostenible y en consonancia con las mejores prácticas del desarrollo de software, lo cual reforzará su valor operativo y tecnológico para la compañía Webcloster SAS.

14. Conclusiones

El desarrollo de este trabajo de grado, realizado durante la pasantía en la empresa Webcloster SAS, permitió cumplir de manera satisfactoria los objetivos planteados. La migración y actualización del *front-end* de la plataforma AcademiApp representó un paso importante para la modernización de su base tecnológica, garantizando su correcto funcionamiento y su permanencia en un entorno productivo.

Desde el punto de vista técnico, la actualización del framework Angular de la versión 14 a la 16 facilitó la eliminación de dependencias obsoletas y la reducción de advertencias durante la compilación, lo que se tradujo en un código más limpio y fácil de mantener. Este proceso dejó la plataforma mejor preparada para afrontar futuras actualizaciones y redujo de manera significativa la deuda técnica acumulada.

En cuanto al rendimiento y la estabilidad, los módulos intervenidos, especialmente el área de noticias y los componentes basados en calendario, mostraron un comportamiento más estable y compatible con las versiones actuales del framework, sin generar impactos negativos en el uso cotidiano de la plataforma por parte de los usuarios.

A nivel de experiencia de usuario, las mejoras visuales realizadas en el *Dashboard* y otros módulos permitieron mantener una interfaz coherente con los lineamientos actuales de Angular Material. Estos ajustes, aunque sutiles, contribuyeron a una navegación más clara y ordenada, conservando la funcionalidad original del sistema.

Finalmente, la modernización tecnológica realizada deja a AcademiApp en una posición favorable para su crecimiento futuro, permitiendo la incorporación de nuevos módulos y la adaptación a cambios tecnológicos según las necesidades de las instituciones educativas. En conjunto, este proyecto no solo fortaleció la plataforma, sino que también representó una experiencia formativa significativa, al permitir aplicar los conocimientos adquiridos durante la carrera en un contexto laboral real y fortalecer las competencias técnicas y profesionales del estudiante.

Referencias

[1] M. Andreessen, "Software Is Eating the World," *The Wall Street Journal*, 2022. Accessed: Jun. 05, 2025. [Online]. Available: https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

[2] L. A. Acosta, F. A. Becerra, and D. Jaramillo, "Sistema de Información Estratégica para la gestión universitaria en la Universidad de Otavalo (Ecuador)," *Nature*, no. 388, pp. 539–547, 1997. Accessed: Jun. 05, 2025. [Online]. Available: https://www.researchgate.net/publication/339540030$_Sistema_de_informacion_estrategica_para_la_gestion_u$

[2] B. V. Deemer, P. Pete, G. Benefield, and C. Larman, "Básica de Scrum (The Scrum Primer)," Scrum Training Institute, 2009. Accessed: Jun. 05, 2025. [Online]. Available: https://assets.scrumtraininginstitute.com/downloads/1/scrumprimer.pdf

[3] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2016. Accessed: Jun. 05, 2025. [Online]. Available: https://dl.acm.org/doi/10.5555/3013530

[4] R. S. Pressman, *Ingeniería de Software: Un enfoque práctico*, 7th ed. McGraw-Hill, 2010. Accessed: Jun. 05, 2025. [Online]. Available: https://books.google.com/books?id=1w88MwEACAAJ

[5] R. Stair and G. Reynolds, *Principles of Information Systems*, 13th ed. Cengage Learning, 2019. Accessed: Jun. 05, 2025. [Online]. Available: https://www.cengage.com/c/principles-of-information-systems-13e-stair/9781337553496/

[6] Google Angular Team, "Angular Documentation," 2024. Accessed: Jun. 05, 2025. [Online]. Available: https://angular.io/docs

[7] M. Fowler, *Refactoring: Improving the Design of Existing Code*, 2nd ed. Addison-Wesley, 2019. Accessed: Jun. 05, 2025. [Online]. Available: https://martinfowler.com/books/refactoring.html

[8] W. Cunningham, "The Technical Debt Metaphor," in *OOPSLA*, 1992. Accessed: Jun. 05, 2025. [Online]. Available: http://c2.com/doc/oopsla92.html

[9] K. Schwaber and J. Sutherland, "The Scrum Guide," Scrum.org, 2020. Accessed: Jun. 05, 2025. [Online]. Available: https://scrumguides.org

[10] N. Ford, R. Parsons, and M. Kemerer, *Building Evolutionary Architectures*. O'Reilly Media, 2017. Accessed: Jun. 05, 2025. [Online]. Available: https://www.oreilly.com/library/view/building-evolutionary-architectures/9781491986363/

[11] S. Valentim, E. Alves, and R. Rodrigues, "Long-Term Evaluation of Technical Debt in Open-Source Software," *arXiv*, 2020. Accessed: Jun. 05, 2025. [Online]. Available: https://arxiv.org/abs/2007.13422

[12] A. Besker, M. Sejersen, and J. Greene, "Experiences on Managing Technical Debt with Code Smells and AntiPatterns," *arXiv*, 2021. Accessed: Jun. 05, 2025. [Online]. Available: https://arxiv.org/abs/2103.05412

[13] Atlassian, "What is Technical Debt?" Accessed: Jun. 05, 2025. [Online]. Available: https://www.atlassian.com/es/agile/software-development/technical-debt

[14] IBM, "Technical Debt Explained." Accessed: Jun. 05, 2025. [Online]. Available: https://www.ibm.com/es-es/think/topics/technical-debt

[15] Techopedia, "Technical Debt Definition." Accessed: Jun. 05, 2025. [Online]. Available: https://www.techopedia.com/es/definicion/deuda-tecnica

[16] S. Garcia, "Informe Trabajo de Grado I – Facultad de Ingeniería," Universidad del Valle, 2025. (Documento interno del estudiante). turn1file0

[17] S. Garcia, "Reporte Técnico de Migración de Angular 15 a 16," Webcloster SAS, 2025. (Documento interno del proyecto). turn1file1

[18] OpenAI, "ChatGPT: Large Language Model for text generation and academic writing support," San Francisco, CA, USA, 2025. [Online]. Available: https://chat.openai.com turn1file1

[19] Western Governors University, "Breaking Down the Web Application Deployment Process," 2023. [Online]. Available: https://www.wgu.edu/blog/breaking-down-web-application-deployment-process2311.html turn1file1

[20] G. McGraw, "Software Security," *IEEE Security & Privacy*, vol. 2, no. 2, pp. 80–83, Mar. 2004. doi: 10.1109/MSECP.2004.1281254.