ReGeo ISSN: 2177-3246

IFDIGITAL: SISTEMA DE REALIDAD AUMENTADA PARA EL CAMPUS SANTANA DO LIVRAMENTO DE LA IFSUL

IFDIGITAL: AUGMENTED REALITY SYSTEM FOR THE SANTANA DO LIVRAMENTO CAMPUS OF IFSUL

IFDIGITAL: SISTEMA DE REALIDADE AUMENTADA PARA O CAMPUS SANTANA DO LIVRAMENTO DA IFSUL

https://doi.org/10.56238/revgeov17n8-172

Data de submissão: 19/07/2026 **Data de publicação:** 19/08/2026

Kevin Sebástian Severo Ortiz[1], Abner Gilead Araujo Guedes[2], Victor Machado Alves[3], Alfredo Parteli Gomes[4], Kauê Vargas Sitó[5]

RESUMEN

Este trabajo describe el desarrollo de IFDigital, un sistema de realidad aumentada diseñado para el campus Santana do Livramento de IFSul. La solución integra tecnologías como Unity, Vuforia, Laravel y C#, y combina una aplicación móvil con un sitio web administrativo para facilitar la orientación y acceso a información relevante para usuarios del campus. El sistema resultó efectivo en prototipos funcionales, cumpliendo con los objetivos establecidos.

Palabras clave: Realidad Aumentada. Unity. Vuforia. Laravel.

ABSTRACT

This paper describes the development of IFDigital, an augmented reality system designed for the Santana do Livramento campus of IFSul. The solution integrates technologies, such as Unity, Vuforia, Laravel, and C#, combining a mobile application with an administrative website to facilitate orientation and access to relevant campus information. The system proved effective in functional prototypes, achieving the set objectives.

Keywords: Augmented Reality. Unity. Vuforia. Laravel.

RESUMO

Este trabalho descreve o desenvolvimento do IFDigital, um sistema de realidade aumentada projetado para o campus Santana do Livramento da IFSul. A solução integra tecnologias como Unity, Vuforia, Laravel e C#, combinando uma aplicação móvel com um site

[1] Tecnólogo en Análisis y Desarrollo de Sistemas. Instituto Federal Sul-rio-grandense Campus Santana do Livramento (IFSul). E-mail: kevin.severo@estudiantes.utec.edu.uy
[2] Profesor Supervisor. Instituto Federal Sul-rio-grandense Campus Santana do Livramento (IFSul). E-mail: abnerguedes@ifsul.edu.br
[3] Profesor Supervisor. Instituto Federal Sul-rio-grandense Campus Santana do Livramento (IFSul). E-mail: victoralves@ifsul.edu.br
[4] Profesor Supervisor. Instituto Federal Sul-rio-grandense Campus Santana do Livramento (IFSul). E-mail: alfredogomes@ifsul.edu.br
[5] Profesor Supervisor. Instituto Federal Sul-rio-grandense Campus Santana do Livramento (IFSul). E-mail: kauesito@ifsul.edu.br

ISSN: 2177-3246

administrativo para facilitar orientação e acesso a informações relevantes do campus. O sistema se mostrou eficaz em protótipos funcionais, atingindo os objetivos estabelecidos.

Palavras-chave: Realidade Aumentada. Unity. Vuforia. Laravel.

1 INTRODUCCIÓN

La realidad aumentada se define como la extensión del mundo real mediante el uso de información relevante para los usuarios (Kirner & Kirner, 2011, pp. 10–25). La percepción de la realidad se basa en los cinco sentidos: vista, audición, olfato, tacto y gusto (Fundación Telefónica, 2011). Con la realidad aumentada, estos sentidos se potencian al presentar información en forma de textos, imágenes dinámicas, sonidos espaciales y sensaciones hápticas generadas en tiempo real por dispositivos tecnológicos (Kirner & Kirner, 2011, pp. 10–25). La incorporación de estos elementos virtuales a lo que es captado por el equipo informático da lugar a una realidad mixta, en la que lo real se complementa con datos informáticos (Rigueros Bello, 2017).

En la actualidad la realidad aumentada se ha aplicado a una multitud de áreas, como aplicaciones móviles enfocadas en el área turística que hacen uso del GPS y la cámara para reconocimiento de imágenes, para tener una idea de la localización del usuario y así proporcionarle información (Salazar Fierro *et al.*, 2018). También aplicándose por empresas del área automotriz para mostrarle al usuario desde cómo se vería su automóvil hasta ayudarlos a tener una mejor experiencia cuando estén leyendo el manual. Lo que viene haciendo que la realidad aumentada se convierta en una tendencia debido a su empleo en varias áreas (Rigueros Bello, 2017).

2 JUSTIFICACIÓN

En los últimos años, con las crecientes iniciativas de desarrollo tecnológico en la frontera como el Parque Tecnológico Regional Norte en Rivera (UTEC, 2020), además de las futuras mejoras en conectividad aérea con la declaración de binacionalidad del aeropuerto de Rivera, estando ahora habilitado para vuelos domésticos desde Brasil (Diario Norte, 2023). Esto atraerá más personas a la frontera, lo que aumentará el número de nuevos estudiantes y visitantes a instituciones tecnológicas como el IFSUL.

Hoy existe una carencia de herramientas digitales para orientar a estudiantes y visitantes en el campus Santana do Livramento del IFSUL, esto dificulta la localización de salones y laboratorios, también dificulta la obtención de información de los docentes. Lo que con el posible aumento de flujo de personas podría verse agravada en los próximos años, generando confusión y desmejorando la experiencia educativa.

Las soluciones tradicionales, como mapas físicos y señalizaciones estáticas no consiguen responder a un ambiente dinámico que cada semestre tiene nuevos horarios,

nuevas propuestas educativas, nuevos profesores. Debido a que estos métodos no tienen la capacidad de actualizarse en tiempo real.

Se propone desarrollar un software basado que haga uso de la realidad aumentada para brindar información a los estudiantes y visitantes ayudando así en su ubicación dentro de las instalaciones del campus, ofreciendo información sobre las carreras, los profesores y la organización de los laboratorios y salones. Haciendo frente a las limitaciones de las soluciones tradicionales.

3 OBJETIVOS

Objetivo general: Desarrollar un sistema de realidad aumentada para el campus Santana do Livramento del IFSUL, que ofrezca a través de interfaces interactivas información de ubicación de salones, horarios de clase e información de los docentes dirigido específicamente a estudiantes, visitantes, profesores y personal administrativo.

Objetivos específicos:

- Desarrollar una aplicación de realidad aumentada para dispositivos móviles, para que sea usada por los estudiantes y visitantes.
- Desarrollar un sitio web que permita al personal administrativo y a los profesores, gestionar y actualizar la información que sea exhibida en la aplicación de realidad aumentada.

4 TRABAJOS RELACIONADOS

En las siguientes secciones se abordarán algunos de los proyectos ya realizados en el área de la realidad aumentada aplicados en ambientes educativos. Obsérvese el uso de esta tecnología para brindarle información a los estudiantes y visitantes de los centros educativos.

4.1 APLICACIÓN MÓVIL DE REALIDAD AUMENTADA PARA LA UBICACIÓN DE LAS AULAS DE CLASE EN EL CAMPUS PORVENIR DE LA UNIVERSIDAD DE LA AMAZONIA

En “Aplicación móvil de realidad aumentada para la ubicación de las aulas de clase en el Campus Porvenir de la Universidad de la Amazonia”. Los autores realizaron una investigación sobre Sistemas con la aplicación de realidad aumentada, fueron levantadas las cuestiones sobre el hecho de que el Campus Porvenir de la Universidad de la Amazonia recibe un número de estudiantes nuevos en cada periodo, estos tenían cierta dificultad para

localizarse en el campus, entonces se planteó como objetivo el desarrollo de una aplicación móvil que hiciera el uso de la realidad aumentada para facilitar la ubicación de los nuevos estudiantes en el campus universitario (Sandoval & Carvajal-Beltrán, 2016).

4.2 SISTEMA DE REALIDAD AUMENTADA PARA LA UBICACIÓN DE ESPACIOS Y TIEMPOS EN UNIVERSIDADES

En “Sistema de realidad aumentada para la ubicación de espacios y tiempos en universidades” los autores realizaron una investigación sobre el desarrollo de herramientas de realidad aumentada para ayudar a la orientación de personas en áreas de ciertas dimensiones como pueden ser campus universitarios. Por lo que presentan desde un inicio como las técnicas de realidad aumentada pueden ayudar a estudiantes y a los trabajadores de la facultad a acceder a información del campus. Obtuvieron como resultado el desarrollo de un sistema que terminó solucionando los problemas de ubicación y localización dentro su campus, mediante el uso de la realidad aumentada facilitando la orientación de personas. El trabajo en sí fue considerado completado de forma satisfactoria. Llegaron a la conclusión que la tecnología de la realidad aumentada nos provee una nueva perspectiva de la forma de obtener información sobre entornos universitarios (Ortiz-Martínez *et al.*, 2023).

4.3 COMPARACIÓN ENTRE LOS TRABAJOS RELACIONADOS

Con el fin de posteriormente basar nuestras decisiones en aspectos claves como el lenguaje de programación, herramientas, base de datos y la arquitectura del sistema realizamos la siguiente Tabla 1 donde se hará un breve comparativa de ambos trabajos.

Se puede observar que ambos utilizar el motor de videojuego Unity y en consecuencia el lenguaje de programación C#, además de la presencia en ambos de un componente de aplicación móvil.

Table 1

Tabla comparativa entre los trabajos relacionados

Trabajo	Lenguaje	Otras herramientas	Tipo	Similitudes	Diferencias
Sandoval & Carvajal-Beltrán, (2016)	C#	Unity, Blender, Vuforia	Aplicación móvil	Ambos usan Unity y Vuforia para el desarrollo.	Este trabajo utiliza modelado 3D con Blender.
Ortiz-Martinez et al. (2023)	C#	SQLite, Firebase	Aplicación móvil y componente web	Guiar a usuarios en espacios físicos utilizando tecnología de AR.	Incluye un componente web.

Fuente: Elaborada por los autores.

5 METODOLOGÍA

Esta sección presenta la metodología usada en este trabajo, además de presentar las herramientas y tecnologías que fueron usadas.

5.1 METODOLOGÍA DE LA INVESTIGACIÓN

Se realizó una búsqueda de trabajos relacionados usando la herramienta de búsqueda Google Académico empleando las palabras/frases clave como “Realidad aumentada”,”Ubicación” ,”Dispositivos móviles”,”Aplicación Móvil”. Para el filtrado se usó un rango de 10 años, por lo que se buscó artículos del 2013 hasta el 2023. Posteriormente después de filtrar la búsqueda por las palabras clave y acotar el rango de años, se le dio prioridad a las que se aplicaban en entornos educativos realizándose por último una selección manual.

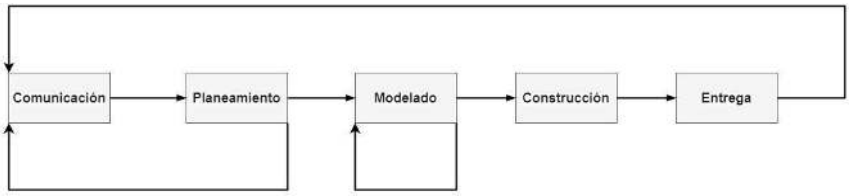
Esta búsqueda permitió identificar tecnologías y herramientas adecuadas para el desarrollo de una aplicación de realidad aumentada además de conocer experiencias y buenas prácticas en la integración de interfaces interactivas y sistemas de gestión de información.

5.2 METODOLOGÍA DE DESARROLLO

Para el desarrollo del sistema de realidad aumentada se hará uso del modelo incremental, también conocido como desarrollo incremental. Este modelo está basado en la idea de desarrollar una versión inicial del software e irla mejorando sucesivamente hasta llegar a un software que cubra las necesidades de los usuarios. Parece adecuado para este proyecto debido al uso de cierta cantidad de tecnologías, es difícil elaborar una solución completa con anterioridad previendo todos los posibles problemas (Pressman, 2011).

Figure 1

Flujo de proceso iterativo adaptado al proyecto Pressman (2021)



Fuente: Elaborada por los autores.

En la figura 1 está el flujo de trabajo que se optó para la realización de este proyecto, es está basado en el Flujo de proceso iterativo, se discute sobre las necesidades del proyecto, posteriormente se planea cómo implementarlas, se hace el modelado de los diagramas luego en construcción se hace la implementación con código, para posteriormente hacer una entrega lo que lleva a hacer devuelta una discusión sobre mejoras y defectos.

5.3 TECNOLOGÍAS

5.3.1 Unity

Unity desarrollado por Unity Technologies lanzado al mercado en 2005, es un motor de videojuegos 2D y 3D creado para brindar a los desarrolladores una serie de herramientas de desarrollo de videojuegos, pero desde aquel entonces ha sufrido muchos cambios en la actualidad posicionándose como uno de los principales motores de videojuegos (Zenva, 2023).

Unity posee varias características clave. Empezando por el soporte a gráficos 3D y 2D, permitiendo el desarrollo de proyectos en diferentes estilos, teniendo herramientas específicas para el 2D y el 3D, también posee una arquitectura fácil de entender Unity divide los proyectos en distintas escenas, cada escena conteniendo sus respectivos objetos, estos objetos tienen cierta capacidad de herencia ya es posible la composición padre-hijo de los objetos que están dentro de los escenarios. Otra característica destacable es el soporte a la compilación multiplataforma, la cual es esencial en cierto tipos de proyecto donde se busca que el software se ejecute en distintos sistemas operativos, Unity soportando actualmente la compilación para las plataformas Android,iOS,Windows,MacOS,Linux,PS y Xbox (Zenva, 2023).

Capacidades de desarrollar software de realidad virtual y aumentada, están presentes en Unity existiendo una gran cantidad de paquetes que facilitan el desarrollo de aplicaciones

de realidad aumentada para Android y iOS de forma simultánea, además de paquetes con compatibilidad con casi todos los cascos de realidad virtual que están disponibles en el mercado (Zenva, 2023).

5.3.2 Vuforia

Vuforia es un kit de desarrollo de software de realidad aumentada que le proporciona a los desarrolladores las herramientas necesarias para la creación de software de visión computacional en gran cantidad de plataformas y dispositivos, haciendo énfasis en la creación de experiencias de realidad aumentada con interacción realista con los objetos y el entorno del mundo real (Vuforia, n.d.).

Vuforia, usado en conjunto con Unity, nos permite la creación de experiencias de realidad aumentada multiplataforma debido a la posibilidad de compilar nuestro software para diversos sistemas operativos a través de Unity, lo cual no sería posible si usaramos en otros entornos de desarrollo como Android Studio por poner un ejemplo (Vuforia, n.d.).

En un estudio realizado en 2021 donde se hace una comparación entre diferentes kits de desarrollo de software de realidad aumentada donde se hace una comparación de las características de diversas características como la facilidad de uso, el reconocimiento de diferentes tipos de objetos,la creación de bases de datos. Vuforia salió muy bien posicionada destacándose sobre otras opciones disponibles en el mercado (Vakaliuk & Pochtoviuk, 2021).

5.3.3 Image Targets

Las image targets son imágenes objetivo que Vuforia puede detectar, mediante la cámara del dispositivo y después comparándolas con su base de datos donde están guardados los patrones posibles de las image targets (Vuforia, n.d.).

Se usan en las aplicaciones para desencadenar eventos como mostrar gráficos o mostrar información. Depende del software pero sus más comunes es en medios impresos, se imprime el image target ya sea en el empaque mismo de un producto o de forma separada permitiendo la interacción del usuario a través de la cámara de su dispositivo móvil (Vuforia, n.d.).

Para generar los image targets que serán posteriormente usados se usó la herramienta de Brosvision (Brosvision, n.d.).

5.4 TESTS DE SOFTWARE

Este trabajo utilizó un enfoque basado en el modelo V, este modelo describe la relación entre acciones que buscan una garantía de calidad con las actividades de construcción iniciales. Se realizaron pruebas con el fin de validar el código, los componentes y la arquitectura del sistema. Se buscó entregar un sistema con la menor cantidad de fallos posibles, que no afecten la experiencia de los usuarios finales (Pressman, 2011).

5.4.1 Tests Unitarios

Se realizaron tests unitarios usando las herramientas PHPUnit, NUnit. Con el fin de verificar el correcto funcionamiento de funciones, métodos y clases individuales. En el componente web al utilizar Laravel como framework se usó PHPUnit debido a que viene listo para usar en ese framework (Laravel, n.d.). En el componente de aplicación móvil al haber sido desarrollado usando C# se optó por NUnit que es un framework de testing compatible con todos los lenguajes .Net ofreciendo las herramientas necesaria para los tests unitarios (NUnit, n.d.).

5.4.2 Tests de Integración

Se realizaron los tests de integración, para probar la correcta integración de los diversos componentes del sistema, en contraste a los tests unitarios que se usan para verificar el correcto funcionamiento de las funciones, métodos y clases individuales, los tests de integración se usan para verificar la interacción de los componentes entre sí. Siendo este test fundamental en este trabajo debido a su estructura con un gran número de clases y controladores (García & López, 2015).

6 RESULTADOS

En las siguientes secciones se presentarán los resultados del trabajo, desde un enfoque práctico en el cual se mostrará paso a paso desde el ingreso de un profesor al sistema por parte de un administrador, el posterior ingreso a su perfil del profesor ingresado para editar su descripción de perfil, hasta finalmente la actualización de la base de datos mediante un consumo del API por parte de un usuario.

6.1 ESTRUCTURA DEL TRABAJO

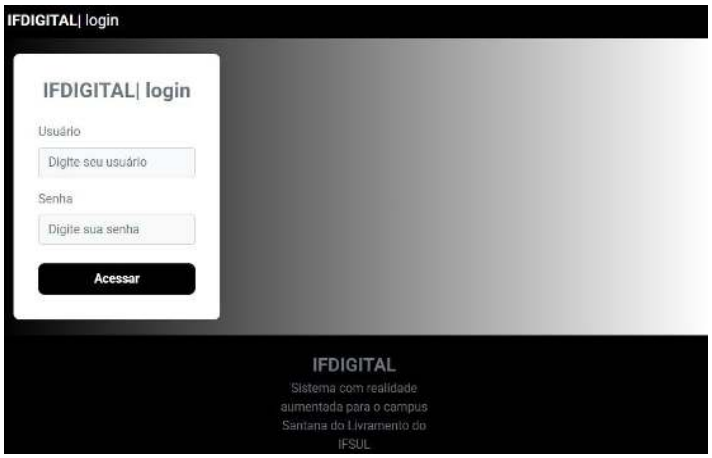
Debido a la arquitectura de este trabajo el cual se divide en un componente web, un componente aplicación, se ha decido separar las siguientes siguiendo ese criterio a fin de tener mayor claridad.

6.2 COMPONENTE WEB

Para ingresar ya sea al dashboard o al editor de perfil del componente web de IFDigital primero hay que pasar por el login. En la Figura 4 se puede observar la pantalla de login.

Figure 2

Página de login de IFDigital



Fuente: Elaborada por los autores.

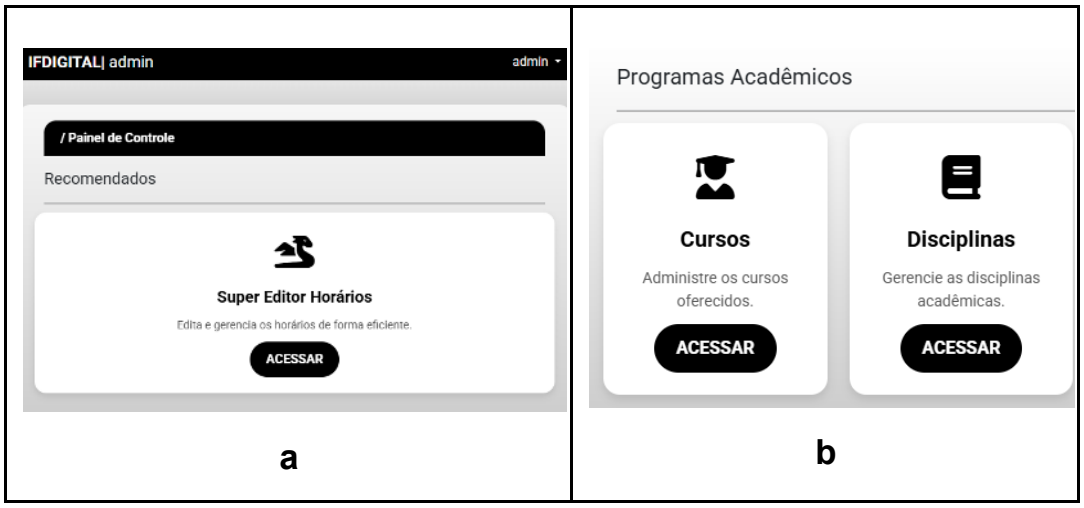
El personal administrativo y profesores de IFDigital pueden tener dos papeles diferentes en el sistema, ellos pueden o ser administradores los cuales poseen más privilegios al poder acceder a la totalidad del sistema o ser profesor que pueden únicamente acceder a la página de editor de perfil.

6.2.1 Dashboard

En este ejemplo el profesor al tener el papel de administrador ingresó al dashboard el cual se puede observar en la Figura 3a y 3b. Este posee el acceso a los CRUD de todas las tablas de la base de datos del sistema, así como también al “Super Editor Horarios” que permite realizar un filtrado más avanzado y observar de forma más intuitiva los horarios de las aulas por periodo académico y curso.

Figure 3

a. Página dashboard y b. Más opciones del dashboard



Fuente: Elaborada por los autores.

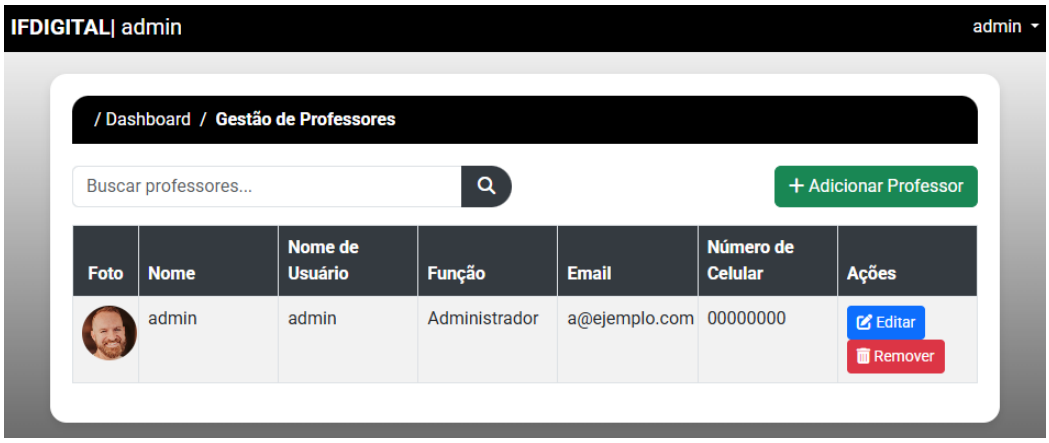
Además en las Figuras 3a y 3b, se observan las tarjetas las cuales llevan a los administradores a los distintos CRUDs para lograr una administración completa del sistema. Siguiendo con el ejemplo, supongamos que el administrador decide ingresar al administrador de profesores.

6.2.2 Administrador de Profesores

El administrador de profesores cuenta con una lista de profesores, un buscador de profesores, un botón para agregar más profesores, si se desea ya sea editar o eliminar un profesor se lo debe buscar en la lista de profesores. En la Figura 4 se puede observar la página en la cual se administra a los profesores.

Figure 4

Página para gestionar profesores

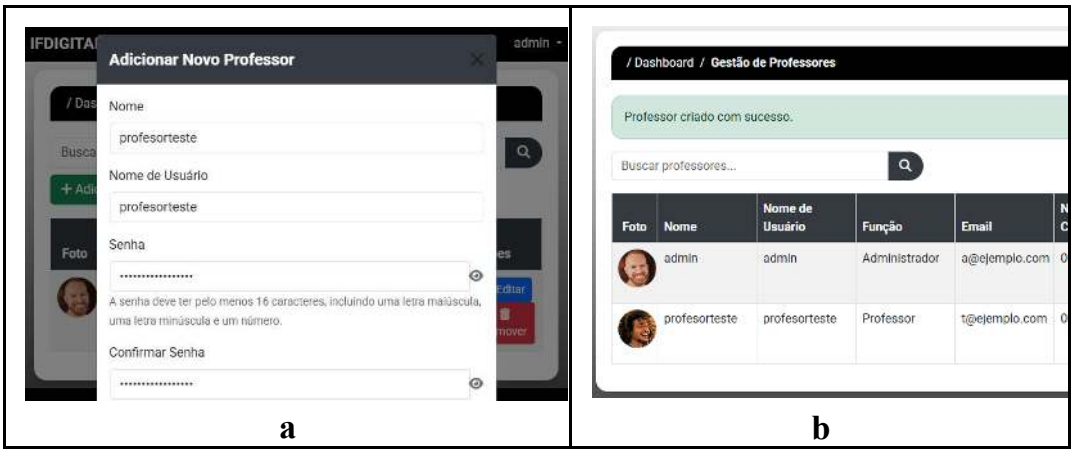


Fuente: Elaborada por los autores.

Al hacer click en el botón verde, se puede agregar un profesor, en este ejemplo agregaremos un profesor. Podemos ver el formulario en la Figura 5a y en la Figura 5b, la confirmación visual que el profesor fue ingresado con suceso.

Figure 5

a. Formulario para agregar un profesor y b. Feedback al usuario que el profesor fue ingresado con suceso



Fuente: Elaborada por los autores.

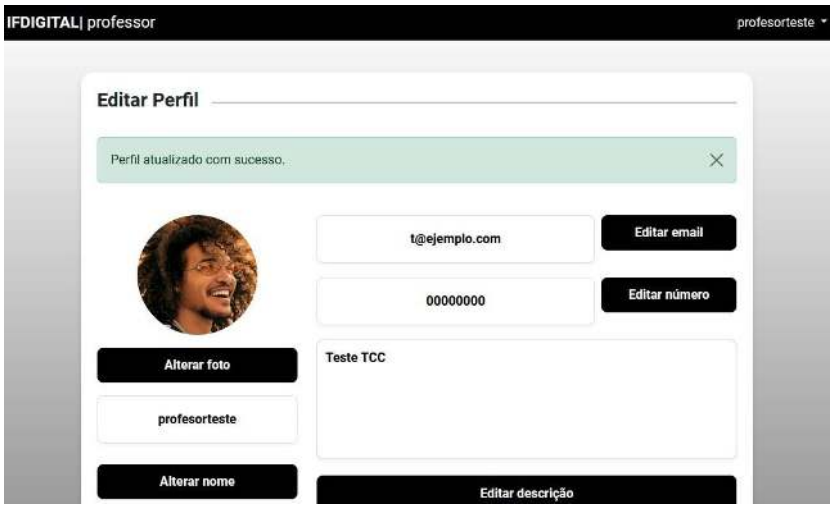
Una vez ingresados el profesor, vamos a suponer que el quiera actualizar sus datos, por lo que él con su usuario y contraseña haría login y sería llevado la página de editor de perfil.

6.2.3 Editor de Perfiles

El editor de perfiles permite la edición completa de todos los datos de un profesor, menos el cambio de papeles, ya que solo un usuario administrador puede cambiar el papel de los otros. En la Figura 6 se observa la página editor de perfil.

Figure 6

Editor de perfiles



Fuente: Elaborada por los autores.

En este se editó la descripción del profesor previamente ingresado en el paso anterior, se cambió su descripción vacía por “Teste TCC” el cual estará visible para los usuario del componente aplicación.

6.3 COMPONENTE APLICACIÓN

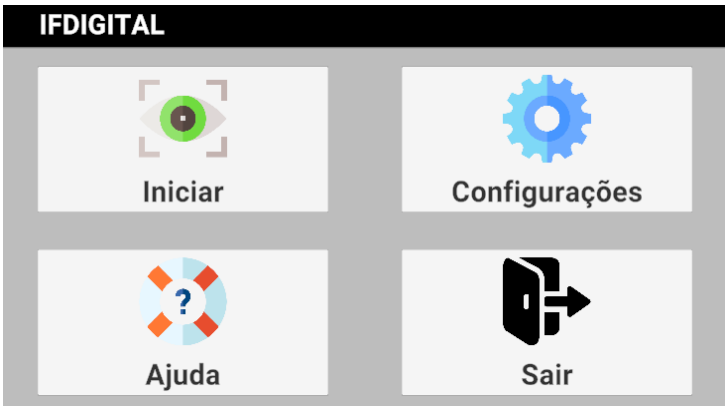
En el componente aplicación no sería necesario ningún tipo de login, por eso se tomaron algunas precauciones como el uso de una api key para evitar accesos indeseados a la información de las consultas, además de que solo se puedan hacer consultas de lectura. Una limitación de consultas por minuto también fue implementada.

6.3.1 Pantalla Principal

El menú principal fue proyectada para que sea simple el cual muestra la posibilidad de tocar 4 botones, uno para ingresar en la experiencia de realidad aumentada, uno para ingresar a los ajustes de la aplicación, el otro para ingresar a la ayuda de la aplicación que le ofrecerá una explicación de cómo usar la aplicación a los estudiantes o visitantes y por último un botón de salir. En la siguiente Figura 9 se observa lo explicado anteriormente.

Figure 7

Menú principal de la aplicación



Fuente: Elaborada por los autores.

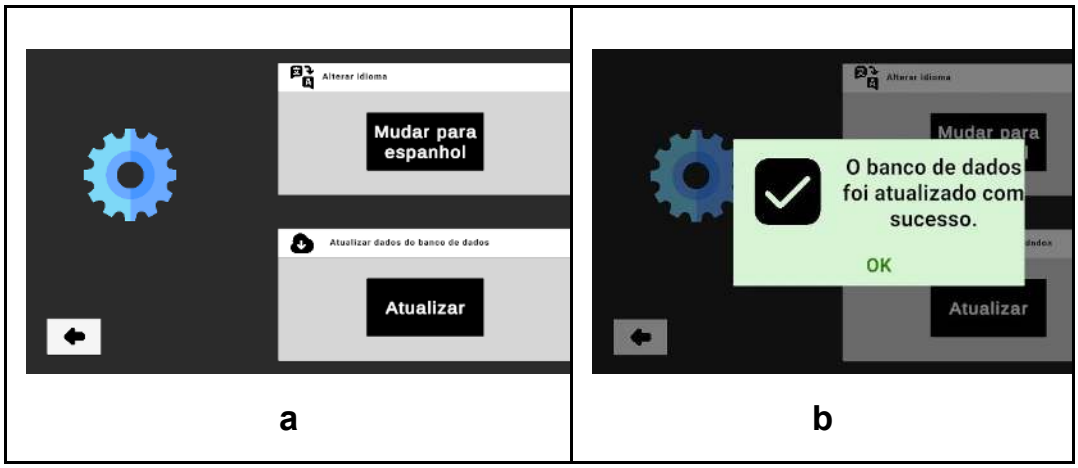
En este ejemplo se procederá al menú de ajustes para actualizar la base de datos SQLite de la aplicación con los datos que se consuman de la API en la cual estarán los datos del profesor que se ha ingresado en las secciones anteriores, así como su información actualizada con la descripción “Teste TCC”.

6.3.2 Pantalla de Ajustes

La pantalla de ajustes ofrece al estudiante la posibilidad de cambiar el idioma de la aplicación así como también el de actualizar la base de datos de la aplicación. En la Figura 8a se observa el menú de ajustes, en la Figura 8b se observa el feedback al usuario de que el consumo de los datos del API así como también la actualización de los datos de la base SQLite fue realizado correctamente.

Figure 8

a. Menú de ajustes y b. Confirmación visual para el usuario



Fuente: Elaborada por los autores.

Después de actualizar con suceso los datos de la aplicación en este ejemplo procederemos a la experiencia de realidad aumentada para mostrar la información del profesor.

6.3.4 Image Targets

Para la identificación de cada salón se generaron 64 image targets con la herramienta anteriormente mencionada en la sección de tecnologías, posteriormente se le añadió un marco y un espacio en blanco ya que según la documentación de Vuforia mejora la detección. En la figura 9a se observa uno de los 64 image targets.

Figure 9

a. Image target y b. Patrones del image target



Fuente: Elaborada por los autores.

Y en la Figura 9b se puede observar de la forma que ve Vuforia los image targets, detectando sus patrones, para poder diferenciarlos de otros.

6.3.5 Realidad Aumentada

Al volver al menú y tocar el botón iniciar se inicia la experiencia de realidad aumentada, para que la información aparezca se debe apuntar a un image target de los cual se explicaron en la sección de tecnologías. En la Figura 10a se puede observar los horarios de un salón y en la Figura 10b los datos del profesor.

Figure 10

a. Horarios de un salón y b. Información del profesor



Fuente: Elaborada por los autores.

Los horarios de un salón son presentados en una lista, cada fila de la lista tiene un botón asociado a ella, al tocar el botón aparece los datos del profesor que dictan la disciplina, Como se puede observar en la descripcion del profesor dice “Teste TCC” por lo que este ejemplo de funcionamiento se da por terminado.

6.4 TESTS

Se realizaron una serie de tests para probar el funcionamiento del sistema, sobre todo en la parte web, donde el cambio de los datos se ven reflejados en el componente de la aplicación.

6.4.1 Tests de Integración

En la Tabla 2 están los tests de integración que tuvieron como objetivo verificar el correcto control de acceso en las rutas del componente web que fue construido con Laravel.

Table 2

Tests de integración para verificar el control de acceso

Nombre del test	Papel evaluado	Rutas evaluadas
administrador_puede_acceder _a_rutas_protegidas	Administrador	Rutas protegidas
professor_no_puede_acceder_ a_rutas_protegidas_para_admi nistrador	Profesor	Rutas protegidas
invitado_no_puede_acceder_a _rutas_protegidas	Invitado	Rutas protegidas
professor_puede_acceder_a_r utas_permitidas	Profesor	Rutas permitidas para el profesor
administrador_puede_acceder	Administrador	Rutas permitidas para el
_a_rutas_de_professor		profesor
invitado_no_puede_acceder_a _rutas_de_professor	Invitado	Rutas permitidas para el profesor

Fuente: Elaborada por los autores.

Los códigos completos de los tests de integración se encuentran en el Apéndice. Así como el código de los otros tests realizados.

7 CONCLUSIONES

El proyecto IFDigital logró desarrollar un sistema de realidad aumentada para facilitar la interacción y orientación dentro del campus Santana do Livramento de IFSul. Se usaron tecnologías como Unity, C#, Vuforia y Laravel, donde se implementó una solución funcional que cumplió con los objetivos establecidos, demostrando su efectividad en prototipos. Este trabajo es un puntapié inicial en la incorporación de herramientas de realidad aumentada en el campus así como el futuro desarrollo de las mismas, sentando una base para futuros proyectos.

A pesar de los desafíos enfrentados durante la integración de múltiples tecnologías, el enfoque iterativo e incremental demostró ser correcto. Con la esperada disminución en los costos de dispositivos de realidad aumentada en un futuro, este proyecto visibiliza el potencial de estas tecnologías y su relevancia en ambientes educativos. Además, se identificaron diversas oportunidades de mejora que podrán ser abordadas en versiones futuras del sistema.

7.1 DESARROLLO FUTURO

Aunque el presente sistema ha cumplidos sus requisitos, ofreciendo una experiencia de realidad aumentada además de la posibilidad de administrar todos los datos que se

muestran en ella. Existe un margen de desarrollo bastante amplio, por lo que presentaremos algunas propuestas a los estudiantes que planeen en futuro desarrollar una versión más completa de IFDigital.

- Ampliación de funciones de accesibilidad. Permitiendo que sea accesible a todos los usuarios.
- Agregar mas funcionalidad intectivas. Desarrollando más interfaces en la realidad aumentada.
- Mejorar la interfaz gráfica del sistema. Mejorar el sitio web, así como también la aplicación de realidad aumentada con interfaces gráficas estéticamente mejores.
- Compatibilidad con gafas de realidad mixta.

REFERENCES

Bello, C. R. (2017). La realidad aumentada: lo que debemos conocer. Tecnología Investigación y Academia, 5(2), 257–261.

Brosvision. (s.f.). AR Marker Generator. Recuperado de https://www.brosvision.com/ar-marker-generator

Diario Norte. (2023, 14 de agosto). Hoy se suscribirá el documento que establece la binacionalidad del aeropuerto de Rivera. Recuperado el 4 de diciembre de 2023, de https://diarionorte.com.uy/sociedad/hoy-se-suscribira-el-documento-que-establece-la-binacionalidad-del-aeropuerto-de-rivera-68722.html

Kirner, C., & Kirner, T. G. (2011). Evolução e tendências da Realidade Virtual e da Realidade Aumentada. En Realidade Virtual e Aumentada: Aplicações e Tendências (Capítulo 1, pp. 10–25).

Ortiz-Martínez, E.-F., Benavides-Zúñiga, D.-A., Amador-Donado, S., Marceles-Villalba, K., & Camacho-Ojeda, M.-C. (2023). Sistema de realidad aumentada para la ubicación de espacios y tiempos en universidades. Revista Facultad de Ingeniería, 32(64), e15538. https://doi.org/10.19053/01211129.v32.n64.2023.15538

Rigueros Bello, C. (2017). La realidad aumentada: lo que debemos conocer. Tecnología Investigación y Academia, 5(2), 257–261. Recuperado de https://revistas.udistrital.edu.co/index.php/tia/article/view/11278

Salazar Fierro, F. A., Pineda Manosalvas, C. A., Arciniega Hidrobo, S. R., & Cervantes Rodríguez, N. N. (2018). Comparativa técnica de herramientas para realidad aumentada: Wikitude, Vuforia y ARToolKit. Axioma, 12(8). https://doi.org/10.26621/XV19.2018.12.A08.PUCESI.2550.6684

Sandoval, C. A., & Carvajal-Beltrán, L. V. (2016). Aplicación móvil de realidad aumentada para la ubicación de las aulas de clase en el Campus Porvenir de la Universidad de la Amazonia - Mobile augmented reality application for the location of classrooms at the Campus Porvenir of the University of the Amazon. Revista Científica, 3(26), 81. https://doi.org/10.14483/23448350.11095

Telefónica, F. (2011). Realidad Aumentada: una nueva lente para ver el mundo. Fundación Telefónica.

UTEC. (2020, 20 de noviembre). Llamado a empresas interesadas en instalarse en el nuevo Parque Tecnológico Regional Norte en Rivera. Recuperado el 4 de diciembre de 2023 de https://utec.edu.uy/es/noticia/llamado-a-empresas-interesadas-en-instalarse-en-el-nuevo-parque-tecnologico-regional-norte-en-rivera/

APÉNDICE 1. PLANEAMIENTO

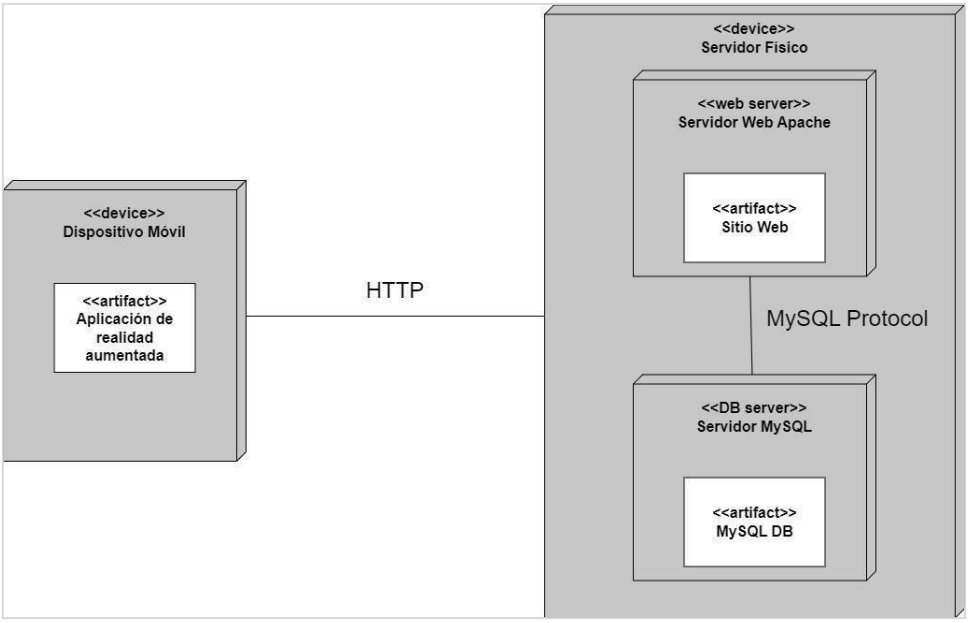
El desarrollo del sistema informativo de realidad aumentada se centró en 3 componentes principales, la aplicación móvil con la cual los usuarios interactúan con la realidad aumentada que hace uso de Unity y Vuforia lo cual permite el reconocer Image Targets para posteriormente mostrar la información relevante, así como almacenar la información localmente en una base de datos SQLite para permitir su uso sin conexión.

El servidor web actuará como un intermediario entre los administradores y la aplicación móvil, usando Laravel como framework de desarrollo backend. Los administradores mediante la interfaz web basada en Laravel administrarán los datos que posteriormente mediante el uso de una API serán consumidos por la aplicación móvil.

El tercer componente sería la comunicación del componente aplicación móvil y el componente servidor web mediante el API, la aplicación móvil no contara con ningún sistema de login con la idea que los usuarios simplemente la ejecuten sin la necesidad de estar creando usuarios debido a esto el API será de solo lectura y se hará uso de una api key en la aplicación móvil para validar las consultas.

Figura 11

Diagrama de despliegue del sistema



Fuente: Autores.

La Figura 13 es un diagrama de despliegue que nos ayuda a visualizar la

arquitectura del sistema que hemos planeado, tenemos 2 nodos, el nodo dispositivo móvil donde estará la aplicación de realidad aumentada y el nodo servidor físico donde estara alojados el servidor Apache y la base de datos MySQL.

REQUISITOS

Se usarán requisitos basados en el artículo relacionado “Sistema de realidad aumentada para la ubicación de espacios y tiempos en universidades”, teniendo en cuenta que estamos en un ambiente similar, un campus universitario con salones, profesores, materias, carreras, etc. En próxima interacciones se podrán agregar los requisitos que se crean pertinentes para que el sistema cumpla su objetivo.

REQUISITOS FUNCIONALES

Table 4

Requisitos funcionales

ID	Descripción del requisito
RF01	La aplicación de realidad aumentada mediante el uso de Image Targets será capaz de distinguir todos los salones del campus.
RF02	Los usuarios mediante el aplicativo de realidad aumentada en sus dispositivos móviles, deben tener la posibilidad de acceder a información relevante de los profesores.
RF03	Los usuarios mediante el aplicativo de realidad aumentada en sus dispositivos móviles, deberán tener la posibilidad de acceder a los horarios de los profesores.
RF04	Los usuarios mediante el aplicativo de realidad aumentada en sus dispositivos móviles, deben tener la posibilidad de acceder a información de las carreras que se dictan en el campus Santana do Livramento.
RF05	Los usuarios mediante el aplicativo de realidad aumentada en sus dispositivos móviles, deberán tener la posibilidad de acceder a los horarios de las asignaturas que se dictan en los salones.
RF06	Los administradores acceden al sitio web mediante un login.
RF07	Los administradores mediante el sitio web podrán registrar, modificar y eliminar salones.
RF08	Los administradores mediante el sitio web podrán registrar, modificar y eliminar profesores.
RF09	Los administradores mediante el sitio web podrán registrar, modificar y eliminar materias.
RF10	Los administradores mediante el sitio web podrán registrar, modificar y eliminar carreras.
RF11	Los administradores mediante el sitio web podrán registrar, modificar y eliminar los horarios de los salones, profesores y materias.

Fuente: Elaborada por los autores.

REQUISITOS NO FUNCIONALES

Table 5

Requisitos no funcionales

ID	Descripción del requisito
RNF01	La aplicación de realidad aumentada mediante el uso de Image Targets será capaz de distinguir todos los salones del campus.
RNF02	El aplicativo de realidad aumentada deberá ser intuitivo, procurando que la interfaz sea simple y fácil de entender para los usuarios.
RNF03	El aplicativo de realidad aumentada deberá ser fiable y estable para su uso por largo periodos de tiempo.
RNF04	Los usuarios mediante el aplicativo de realidad aumentada en sus dispositivos móviles, deben tener la posibilidad de acceder a información relevante de los profesores.

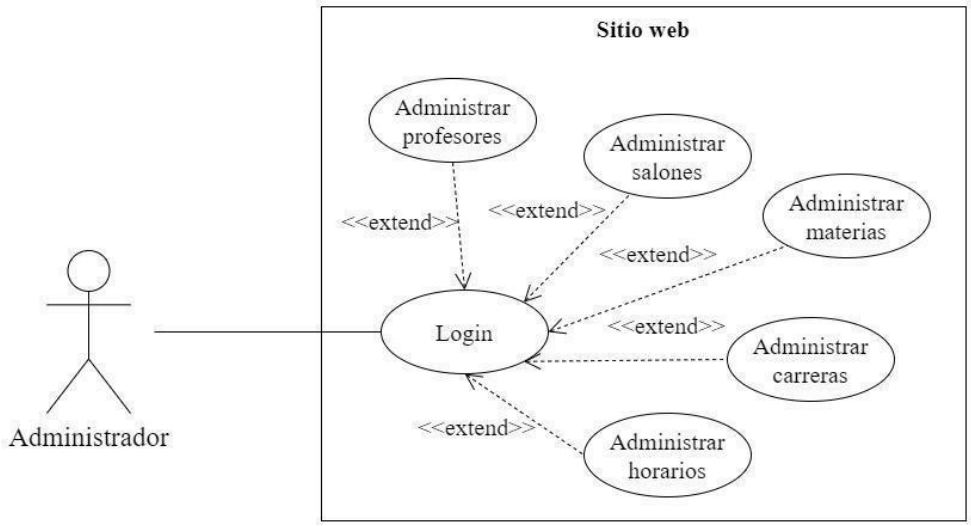
Fuente: Elaborada por los autores.

DIAGRAMA DE CASOS DE USO

En base a los requisitos funcionales y los no funcionales, además de los contextos teniendo en cuenta la interacción que tendrán los administradores y los usuarios con el sistema, se optó por hacer 2 diagramas de casos de uso que representen sus interacciones. La figura 14 es un diagrama de casos de uso la cual representan las interacciones de los administradores con el sitio web de administración, en la cual también están los requisitos funcionales RF06, RF07, RF08, RF09, RF10 y RF11.

Figure 12

Diagrama de casos de uso

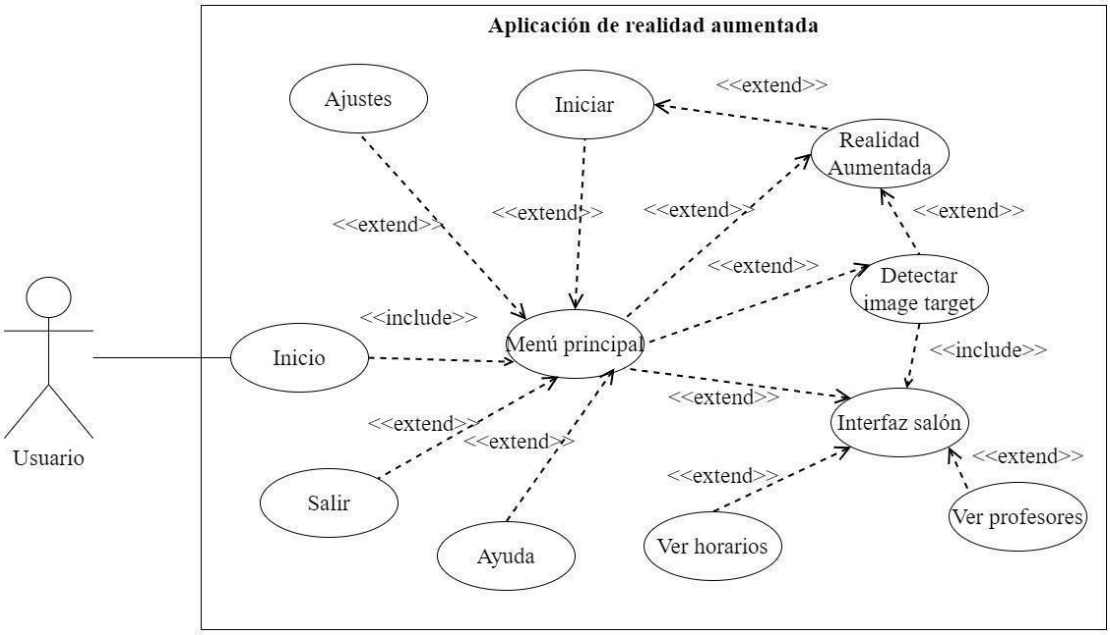


Fuente: Elaborada por los autores.

La figura 15 es un diagrama de casos de uso la cual representan las interacciones de los usuarios con la aplicación de realidad aumentada, en la cual también están los requisitos funcionales RF01, RF02, RF03, RF04 y RF05.

Figure 12

Diagrama de casos de uso de la interacción de los usuarios con la aplicación móvil



Fuente: Elaborada por los autores.

FINALIZACIÓN DEL PLANEAMIENTO

Se finalizó el planeamiento habiendo definido la arquitectura del sistema, los requisitos funcionales y no funcionales. También como las tecnologías y herramientas que serán usadas. Mediante los diagramas de casos de uso se definió de cierta forma las interfaces gráficas del sistema.

IMPLEMENTACIÓN DEL SISTEMA

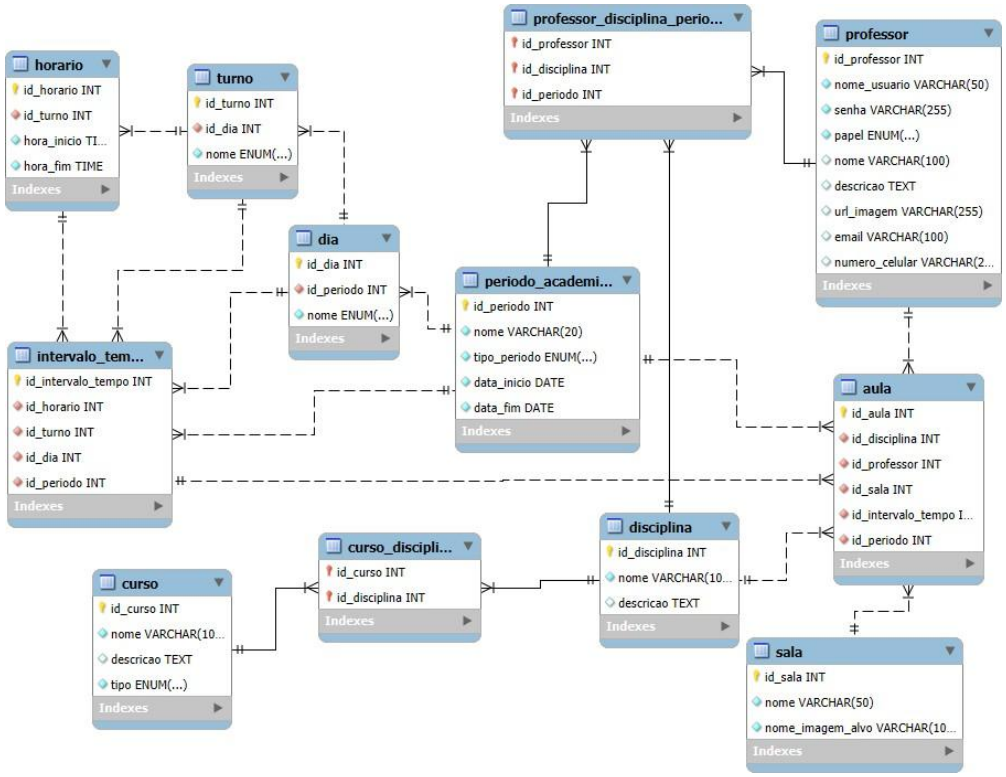
A modo de ejemplificar mostrando un uso más práctico, en las siguientes secciones mediante el uso de diagramas entidad-relación, diagramas de secuencia y capturas de pantalla del sistema. Se explica como funciona el sistema desde que un administrador ingresa los datos de un profesor hasta que el usuario consume esos datos a través del API.

BASE DE DATOS MYSQL

En la siguiente Figura 16 se muestra como estará organizada la información del lado del servidor.

Figure 13

Diagrama ER de la base de datos del sitio web



Fuente: Elaborada por los autores.

Se guardará toda la información que posteriormente se usará para ser consumida por el API. La base de datos consta de 12 tablas donde serán almacenados los datos ingresados por los administradores a través del sitio web de administración.

FLUJO DE GESTIÓN DE DATOS: INGRESO DE UN PROFESOR

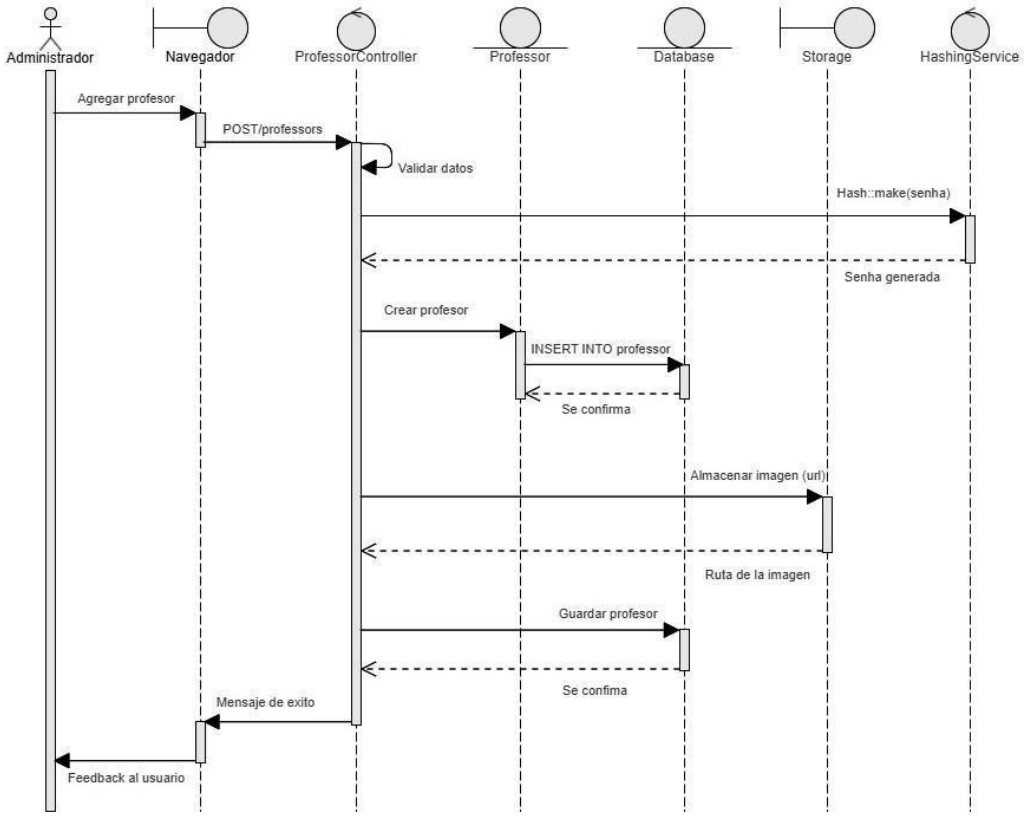
Después de hacer login, los administradores acceden a un dashboard donde estarán disponibles los CRUDs para cada tabla de la base de datos. Al ingresar al CRUD de profesor podrán administrar los profesores. En la siguiente figura, la Figura 17 se podrá observar un diagrama de secuencia. En el diagrama se mostrará el proceso completo de registro de un profesor en el sistema a través del sitio de administración. Primero un administrador completará y enviará un formulario de “Agregar Profesor” desde el navegador. Seguido a eso se realizara una validación de los datos del lado del servidor de la siguiente forma: se

realiza una solicitud POST al endpoint /professors, enviando el formulario al controlador ProfessorController, este controlador validará los datos recibidos para asegurar que estén correctos y completos.

Luego de la validación el controlador hará uso de un servicio de hashing para cifrar la contraseña antes de su almacenamiento. Esto se hace para garantizar cierta seguridad de las credenciales de los profesores.

Figure 14

Diagrama de secuencia



Fuente: Elaborada por los autores.

Luego una nueva instancia del modelo Professor se crea utilizando los datos proporcionados, el modelo entonces interactuara con la base de datos MySQL para insertar los datos del profesor. Después que se inserten los datos la base confirmará la operación.

Existirá también un manejo de la imagen del profesor en caso de que se proporcione una foto de perfil en el formulario, dependiendo si se proporcionó o no la imagen el controlador podrá omitir este paso. Luego el controlador almacena el registro completo del profesor y en caso de ser necesario la ruta de la foto de perfil en la base de datos.

Por último al completar el proceso el controlador redirige al navegador a la página /professorescrud con el mensaje de éxito en caso de haber sido exitoso.

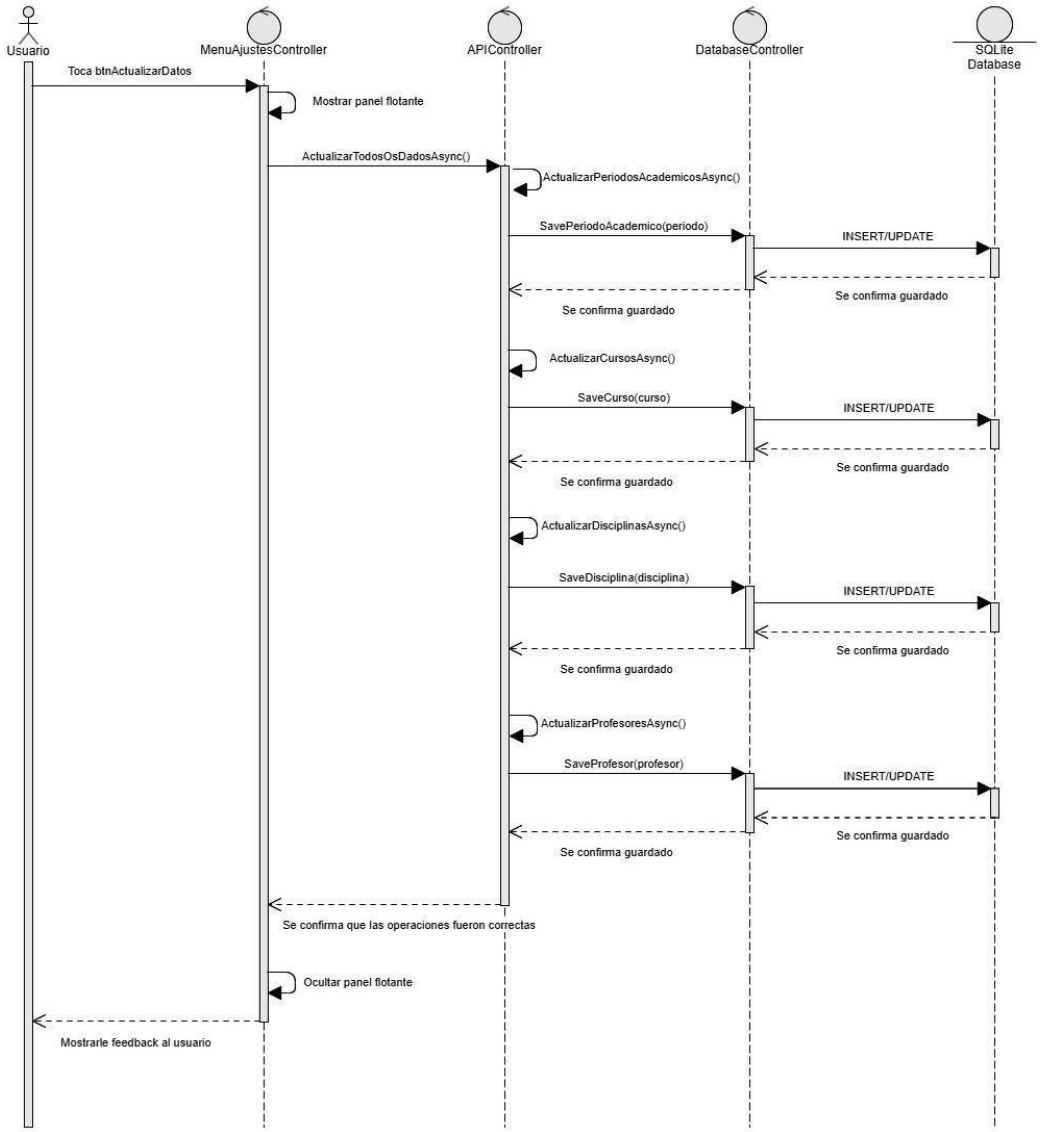
El administrador tendrá una confirmación visual de que el profesor fue agregado correctamente.

FLUJO DE CONSUMO DE INFORMACIÓN

En la sección anterior se observó mediante un diagrama de secuencia del ingreso de un profesor, en esta sección en cambio se centrará en explicar el diagrama de secuencia que ilustra cómo se consume la información del API con la aplicación móvil.

Figure 15

Diagrama de secuencia



Fuente: Elaborada por los autores.

En la Figura 16 tenemos el diagrama de secuencia en el cual se muestra el flujo completo de cómo se actualiza la base de datos SQLite de la aplicación móvil mediante el consumo del API.

Primero el usuario ingresara el menú de ajustes a partir del menú principal, luego el usuario hará click en el botón “Actualizar datos”, lo que activará la funcionalidad en el controlador de ajustes MenuAjustesController.

Habrá una retroalimentación al usuario mediante un panel flotante indicando el inicio del proceso de actualización de datos. El controlador de ajustes invocará el método ActualizarTodosOsDadosAsync() en el controlador de la API llamado APIController, que es el encargado de coordinar la actualización de todas las entidades de la aplicación móvil. Seguido se hará una actualización de las entidades, después de este procesamiento se le devolverá al MenuAjustesController un mensaje indicando si la operación fue exitosa o no.

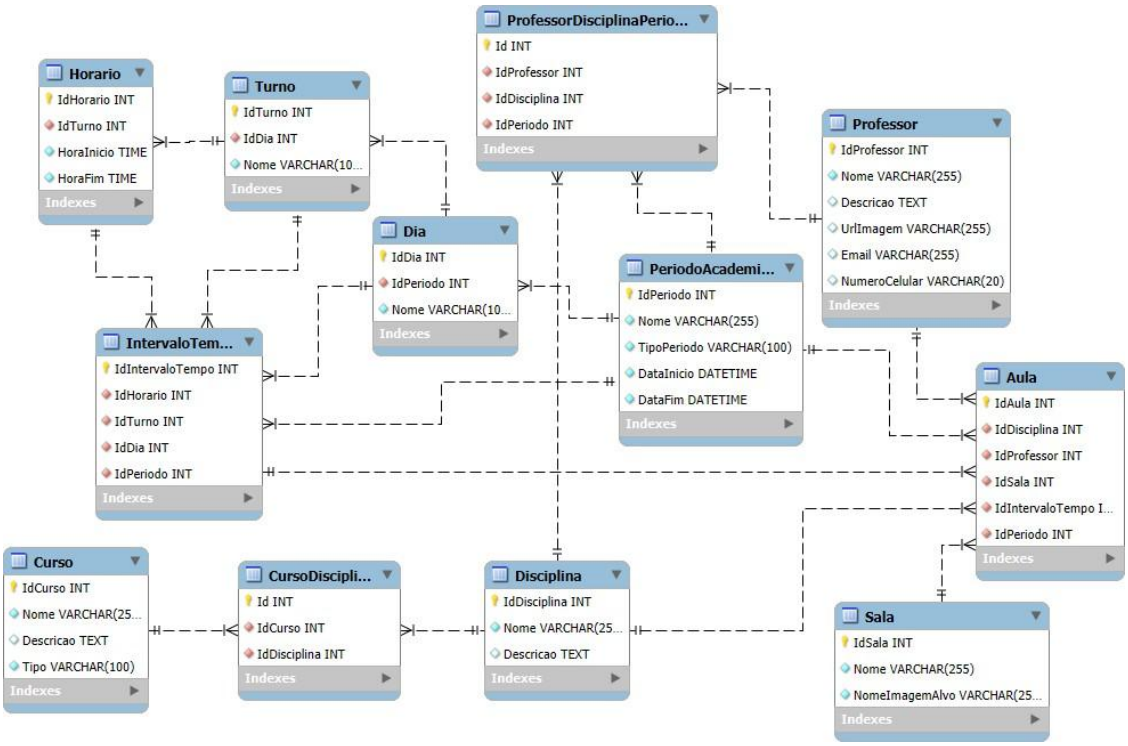
Se termina con retroalimentación visual al usuario, el controlador de ajustes ocultara el panel flotante y dependiendo del resultado mostrará un panel flotante de éxito o un panel flotante de error, para que el usuario sepa si fue actualizada o no la base de datos de la aplicación.

BASE DE DATOS SQLITE

En la Figura 17 está el diagrama entidad relación de la base de datos SQlite que se encuentra en la aplicación móvil. Esta tiene similitudes con la de que se encuentra en el sitio web, ya que la idea es copiar los datos mediante las consultas del API, para que estén disponible sin conexión a internet.

Figure 16

Diagrama ER de la base de datos de la aplicación



Fuente: Elaborada por los autores.

Se pueden observar algunas diferencias como por ejemplo que en la tabla Profesores no se almacenan ningún dato relacionado con las contraseñas, esto se debe por motivos de seguridad y la ausencia de login de la aplicación, la base de datos SQLite es más bien una copia de los datos proporcionados por el API.

Apéndice 2.1 AccessControlTest.php

Figure 17

Test de integración

```php
/** @test */
public function administrador_puede_acceder_a_rutas_protegidas()
{
    $admin = Professor::factory()->administrador()->create([
        'nome_usuario' => 'admin_user',
        'senha' => bcrypt('password123'),
    ]);

    $rutasProtegidas = [
        '/dashboard',
        '/professorescrud',
        '/cursoscrud',
        '/periodosacademicoscrud',
        '/disciplinascrud',
        '/salascrud',
        '/diascrud',
        '/turnoscrud',
        '/horarioscrud',
        '/intervaloscrud',
        '/cursosdisciplinascrud',
        '/professoresdisciplinasperiodoscrud',
        '/aulascrud',
        '/supereditorhorarioscrud',
    ];

    foreach ($rutasProtegidas as $ruta) {
        $response = $this->actingAs($admin)->get($ruta);
        $response->assertStatus(200);
    }
}
```

Fuente: Elaborada por los autores.

Figure 18

Test de integración

```php
/** @test */
public function professor_no_puede_acceder_a_rutas_protegidas_para_administrador()
{
    $professor = Professor::factory()->create([
        'nome_usuario' => 'professor_user',
        'senha' => bcrypt('password123'),
    ]);

    $rutasProtegidas = [
        '/dashboard',
        '/professorescrud',
        '/cursoscrud',
        '/periodosacademicoscrud',
        '/disciplinascrud',
        '/salascrud',
        '/diascrud',
        '/turnoscrud',
        '/horarioscrud',
        '/intervaloscrud',
        '/cursosdisciplinascrud',
        '/professoresdisciplinasperiodoscrud',
        '/aulascrud',
        '/supereditorhorarioscrud',
    ];

    foreach ($rutasProtegidas as $ruta) {
        $response = $this->actingAs($professor)->get($ruta);
        $response->assertStatus(403); // Forbidden
    }
}
```

Fuente: Elaborada por los autores.

Figure 19

Test de integración

```php
/** @test */
public function invitado_no_puede_acceder_a_rutas_protegidas()
{
    $rutasProtegidas = [
        '/dashboard',
        '/professorescrud',
        '/cursoscrud',
        '/periodosacademicoscrud',
        '/disciplinascrud',
        '/salascrud',
        '/diascrud',
        '/turnoscrud',
        '/horarioscrud',
        '/intervaloscrud',
        '/cursosdisciplinascrud',
        '/professoresdisciplinasperiodoscrud',
        '/aulascrud',
        '/supereditorhorarioscrud',
    ];

    foreach ($rutasProtegidas as $ruta) {
        $response = $this->get($ruta);
        $response->assertRedirect('/login'); // Asumiendo que los invitados son
redirigidos al login
    }
}
```

Fuente: Elaborada por los autores.

Figure 20

Test de integración

```
    /** @test */
    public function invitado_no_puede_acceder_a_rutas_protegidas()
    {
        $rutasProtegidas = [
            '/dashboard',
            '/professorescrud',
            '/cursoscrud',
            '/periodosacademicoscrud',
            '/disciplinascrud',
            '/salascrud',
            '/diascrud',
            '/turnoscrud',
            '/horarioscrud',
            '/intervaloscrud',
            '/cursosdisciplinascrud',
            '/professoresdisciplinasperiodoscrud',
            '/aulascrud',
            '/supereditorhorarioscrud',
        ];

        foreach ($rutasProtegidas as $ruta) {
            $response = $this->get($ruta);
            $response->assertRedirect('/login'); // Asumiendo que los invitados son
redirigidos al login
        }
    }
```

Fuente: Elaborada por los autores.

Figure 21

Test de integración

```
    /** @test */
    public function professor_puede_acceder_a_rutas_permitidas()
    {
        $professor = Professor::factory()->create([
            'nome_usuario' => 'professor_user',
            'senha' => bcrypt('password123'),
        ]);

        $rutasPermitidas = [
            '/editarperfil',
            // Agrega aquí otras rutas que los profesores puedan acceder
        ];

        foreach ($rutasPermitidas as $ruta) {
            $response = $this->actingAs($professor)->get($ruta);
            $response->assertStatus(200);
        }
    }
```

Fuente: Elaborada por los autores.

ISSN: 2177-3246

Figure 22

Test de integración

```php
    /** @test */
    public function administrador_puede_acceder_a_rutas_de_professor()
    {
        $admin = Professor::factory()->administrador()->create([
            'nome_usuario' => 'admin_user',
            'senha' => bcrypt('password123'),
        ]);

        $rutasProfesor = [
            '/editarperfil',
            // Agrega aquí otras rutas que los profesores puedan acceder
        ];

        foreach ($rutasProfesor as $ruta) {
            $response = $this->actingAs($admin)->get($ruta);
            $response->assertStatus(200);
        }
    }
```

Fuente: Elaborada por los autores.

Figure 23

Test de integración

```php
    /** @test */
    public function invitado_no_puede_acceder_a_rutas_de_professor()
    {
        $rutasProfesor = [
            '/editarperfil',
            // Agrega aquí otras rutas que los profesores puedan acceder
        ];

        foreach ($rutasProfesor as $ruta) {
            $response = $this->get($ruta);
            $response->assertRedirect('/login'); // Asumiendo que los invitados son
redirigidos al login
        }
    }
```

Fuente: Elaborada por los autores.

Apéndice 2.2 EditarPerfilControllerTest.php

Figure 24

Test de integración

```php
/** @test */
    public function usuarios_no_autenticados_no_pueden_acceder_a_editar_perfil()
    {
        $response = $this->get(route('editarperfil'));
        $response->assertRedirect(route('login'));

        $response = $this->post(route('editarperfil.atualizarPerfil'), []);
        $response->assertRedirect(route('login'));
    }
```

Fuente: Elaborada por los autores.

Figure 25

Test de integración

```php
    /** @test */
    public function usuario_autenticado_puede_ver_pagina_editar_perfil()
    {
        $user = Professor::factory()->create();
        $this->actingAs($user);

        $response = $this->get(route('editarperfil'));
        $response->assertStatus(200);
        $response->assertViewIs('editarperfil');
        $response->assertViewHas('professor', $user);
    }
```

Fuente: Elaborada por los autores.

Figure 26

Test de integración

```php
/** @test */
public function usuario_puede_actualizar_nombre()
{
    $user = Professor::factory()->create();
    $this->actingAs($user);

    $data = [
        'nome' => 'Novo Nome',
    ];

    $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

    $response->assertRedirect(route('editarperfil'));
    $response->assertSessionHas('success', 'Perfil atualizado com sucesso.');

    $this->assertDatabaseHas('professor', [
        'id_professor' => $user->id_professor,
        'nome' => 'Novo Nome',
    ]);
}
```

Fuente: Elaborada por los autores.

Figure 27

Test de integración

```php
/** @test */
public function usuario_puede_actualizar_correo_electronico()
{
    $user = Professor::factory()->create();
    $this->actingAs($user);

    $data = [
        'email' => 'novoemail@example.com',
    ];

    $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

    $response->assertRedirect(route('editarperfil'));
    $response->assertSessionHas('success', 'Perfil atualizado com sucesso.');

    $this->assertDatabaseHas('professor', [
        'id_professor' => $user->id_professor,
        'email' => 'novoemail@example.com',
    ]);
}
```

Fuente: Elaborada por los autores.

Figure 28

Test de integración

```php
/** @test */
public function usuario_no_puede_actualizar_correo_electronico_a_correo_existente()
{
    $user = Professor::factory()->create();
    $otherUser = Professor::factory()->create(['email' => 'existingemail@example.com']);
    $this->actingAs($user);

    $data = [
        'email' => 'existingemail@example.com',
    ];

    $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

    $response->assertSessionHasErrors('email');
    $this->assertEquals($user->email, Professor::find($user->id_professor)->email);
}
```

Fuente: Elaborada por los autores.

Figure 29

Test de integración

```php
/** @test */
public function usuario_puede_actualizar_numero_de_telefono()
{
    $user = Professor::factory()->create();
    $this->actingAs($user);

    $data = [
        'numero_celular' => '1234567890',
    ];

    $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

    $response->assertRedirect(route('editarperfil'));
    $response->assertSessionHas('success', 'Perfil atualizado com sucesso.');

    $this->assertDatabaseHas('professor', [
        'id_professor' => $user->id_professor,
        'numero_celular' => '1234567890',
    ]);
}
```

Fuente: Elaborada por los autores.

Figure 30

Test de integración

```php
/** @test */
public function usuario_puede_actualizar_descripcion()
{
    $user = Professor::factory()->create();
    $this->actingAs($user);

    $data = [
        'descricao' => 'Nova descrição do professor.',
    ];

    $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

    $response->assertRedirect(route('editarperfil'));
    $response->assertSessionHas('success', 'Perfil atualizado com sucesso.');

    $this->assertDatabaseHas('professor', [
        'id_professor' => $user->id_professor,
        'descricao' => 'Nova descrição do professor.',
    ]);
}
```

Fuente: Elaborada por los autores.

ISSN: 2177-3246

Figure 31

Test de integración

```php
/** @test */
public function usuario_puede_actualizar_foto_de_perfil()
{
    Storage::fake('public');

    $user = Professor::factory()->create([
        'url_imagem' => null,
    ]);
    $this->actingAs($user);

    $file = UploadedFile::fake()->image('avatar.jpg');

    $data = [
        'url_imagem' => $file,
    ];

    $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

    $response->assertRedirect(route('editarperfil'));
    $response->assertSessionHas('success', 'Perfil atualizado com sucesso.');

    $user = $user->fresh();

    // Verificar que el archivo fue almacenado
    Storage::disk('public')->assertExists($user->url_imagem);

    $this->assertNotNull($user->url_imagem);
}
```

Fuente: Elaborada por los autores.

Figure 32

Test de integración

```
    /** @test */
    public function usuario_puede_actualizar_contrasena()
    {
        $user = Professor::factory()->create([
            'senha' => Hash::make('SenhaAnterior123'),
        ]);
        $this->actingAs($user);

        $data = [
            'senha_anterior' => 'SenhaAnterior123',
            'nova_senha' => 'NovaSenhaSegura123',
            'nova_senha_confirmation' => 'NovaSenhaSegura123',
        ];

        $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

        $response->assertRedirect(route('editarperfil'));
        $response->assertSessionHas('success', 'Perfil atualizado com sucesso.');

        $user = $user->fresh();

        $this->assertTrue(Hash::check('NovaSenhaSegura123', $user->senha));
    }
```

Fuente: Elaborada por los autores.

Figure 33

Test de integración

```
    /** @test */
    public function
usuario_no_puede_actualizar_contrasena_con_contrasena_actual_incorrecta()
    {
        $user = Professor::factory()->create([
            'senha' => Hash::make('SenhaAnterior123'),
        ]);
        $this->actingAs($user);

        $data = [
            'senha_anterior' => 'SenhaErrada123',
            'nova_senha' => 'NovaSenhaSegura123',
            'nova_senha_confirmation' => 'NovaSenhaSegura123',
        ];

        $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

        $response->assertSessionHasErrors('senha_anterior');

        $user = $user->fresh();

        $this->assertTrue(Hash::check('SenhaAnterior123', $user->senha));
    }
```

Fuente: Elaborada por los autores.

Figure 34

Test de integración

```php
/** @test */
public function usuario_no_puede_actualizar_contrasena_con_nueva_contrasena_invalida()
{
    $user = Professor::factory()->create([
        'senha' => Hash::make('SenhaAnterior123'),
    ]);
    $this->actingAs($user);

    $data = [
        'senha_anterior' => 'SenhaAnterior123',
        'nova_senha' => 'short',
        'nova_senha_confirmation' => 'short',
    ];

    $response = $this->post(route('editarperfil.atualizarPerfil'), $data);

    $response->assertSessionHasErrors('nova_senha');

    $user = $user->fresh();

    $this->assertTrue(Hash::check('SenhaAnterior123', $user->senha));
}
```

Fuente: Elaborada por los autores.

Apéndice 2.3 ProfessorControllerTest.php

Figure 35

Test de integración

```php
/** @test */
public function usuarios_no_autenticados_no_pueden_acceder_rutas_profesor()
{
    $professor = Professor::factory()->create();

    $this->get(route('professorescrud'))
        ->assertRedirect(route('login'));

    $this->post(route('professors.store'), [])
        ->assertRedirect(route('login'));

    $this->put(route('professors.update', $professor->id_professor), [])
        ->assertRedirect(route('login'));

    $this->delete(route('professors.destroy', $professor->id_professor))
        ->assertRedirect(route('login'));
}
```

Fuente: Elaborada por los autores.

Figure 36

Test de integración

```php
/** @test */
public function usuarios_no_admin_no_pueden_acceder_rutas_profesor()
{
    $user = Professor::factory()->create(['papel' => 'professor']);
    $this->actingAs($user);

    $professor = Professor::factory()->create();

    $this->get(route('professorescrud'))
        ->assertStatus(403);

    $this->post(route('professors.store'), [])
        ->assertStatus(403);

    $this->put(route('professors.update', $professor->id_professor), [])
        ->assertStatus(403);

    $this->delete(route('professors.destroy', $professor->id_professor))
        ->assertStatus(403);
}
```

Fuente: Elaborada por los autores.

Figure 37

Test de integración

```php
/** @test */
public function administrador_puede_acceder_indice()
{
    $user = Professor::factory()->create(['papel' => 'administrador']);
    $this->actingAs($user);

    $response = $this->get(route('professorescrud'));
    $response->assertStatus(200);
    $response->assertViewIs('professorescrud');
}
```

Fuente: Elaborada por los autores.

Figure 38

Test de integración

```
    /** @test */
    public function indice_muestra_profesores()
    {
        $user = Professor::factory()->create(['papel' => 'administrador']);
        $this->actingAs($user);

        $professors = Professor::factory()->count(15)->create();

        $response = $this->get(route('professorescrud'));
        $response->assertStatus(200);
        $response->assertViewIs('professorescrud');
        $response->assertViewHas('professors');
    }
```

Fuente: Elaborada por los autores.

Figure 39

Test de integración

```
    /** @test */
    public function indice_busca_profesores()
    {
        $user = Professor::factory()->create(['papel' => 'administrador']);
        $this->actingAs($user);

        $professor1 = Professor::factory()->create(['nome' => 'John Doe']);
        $professor2 = Professor::factory()->create(['nome' => 'Jane Smith']);

        $response = $this->get(route('professorescrud', ['search' => 'John']));
        $response->assertStatus(200);
        $response->assertViewIs('professorescrud');
        $response->assertViewHas('professors');
        $professors = $response->viewData('professors');

        $this->assertTrue($professors->contains($professor1));
        $this->assertFalse($professors->contains($professor2));
    }
```

Fuente: Elaborada por los autores.

Figure 40

Test de integración

```php
/** @test */
public function administrador_puede_guardar_profesor()
{
    Storage::fake('public');

    $user = Professor::factory()->create(['papel' => 'administrador']);
    $this->actingAs($user);

    $password = $this->generateValidPassword();

    $data = [
        'nome' => 'Test Professor',
        'nome_usuario' => 'testprofessor',
        'senha' => $password,
        'senha_confirmation' => $password,
        'papel' => 'professor',
        'email' => 'testprofessor@example.com',
        'numero_celular' => '123456789',
        'descricao' => 'Test description',
        'url_imagem' => UploadedFile::fake()->image('avatar.jpg'),
    ];

    $response = $this->post(route('professors.store'), $data);

    $response->assertRedirect(route('professorescrud'));
    $response->assertSessionHas('success', 'Professor criado com sucesso.');

    $this->assertDatabaseHas('professor', [
        'nome_usuario' => 'testprofessor',
        'nome' => 'Test Professor',
        'email' => 'testprofessor@example.com',
    ]);

    // Verificar que la imagen fue almacenada
    $professor = Professor::where('nome_usuario', 'testprofessor')->first();
    Storage::disk('public')->assertExists($professor->url_imagem);
}
```

Fuente: Elaborada por los autores.

Figure 41

Test de integración

```php
/** @test */
public function guardar_profesor_errores_de_validacion()
{
    $user = Professor::factory()->create(['papel' => 'administrador']);
    $this->actingAs($user);

    $data = [
        // Faltan campos requeridos y datos inválidos
        'nome_usuario' => 'testprofessor',
        'senha' => 'short',
        'senha_confirmation' => 'short',
        'papel' => 'invalidrole',
        'email' => 'invalidemail',
        'numero_celular' => '123456789',
        'descricao' => 'Test description',
        'url_imagem' => 'not-an-image',
    ];

    $response = $this->post(route('professors.store'), $data);

    $response->assertSessionHasErrors([
        'nome',
        'senha',
        'papel',
        'email',
        'url_imagem',
    ]);

    // Verificar que no se creó el profesor
    $this->assertDatabaseMissing('professor', ['nome_usuario' => 'testprofessor']);
}
```

Fuente: Elaborada por los autores.

Figure 42

Test de integración

```php
/** @test */
public function administrador_puede_actualizar_profesor()
{
    Storage::fake('public');

    $user = Professor::factory()->create(['papel' => 'administrador']);
    $this->actingAs($user);

    // Crear un profesor con una imagen existente
    $professor = Professor::factory()->create([
        'url_imagem' => 'uploads/original.jpg',
    ]);

    // Crear la imagen existente
    Storage::disk('public')->put('uploads/original.jpg', 'original content');

    // Datos para actualizar sin incluir una nueva imagen
    $data = [
        'nome' => 'Updated Name',
        'nome_usuario' => 'updateduser',
        'papel' => 'administrador',
        'email' => 'updated@example.com',
        'numero_celular' => '987654321',
        'descricao' => 'Updated description',
        // No enviamos 'url_imagem'
    ];

    $response = $this->put(route('professors.update', $professor->id_professor), $data);

    $response->assertRedirect(route('professorescrud'));
    $response->assertSessionHas('success', 'Professor atualizado com sucesso.');

    // Verificar en la base de datos
    $this->assertDatabaseHas('professor', [
        'id_professor' => $professor->id_professor,
        'nome' => 'Updated Name',
        'nome_usuario' => 'updateduser',
        'email' => 'updated@example.com',
    ]);

    // Verificar que la imagen antigua aún existe
    Storage::disk('public')->assertExists('uploads/original.jpg');
}
```

Fuente: Elaborada por los autores.

Figure 43

Test de integración

```php
/** @test */
public function administrador_puede_actualizar_profesor_con_contraseña()
{
    $user = Professor::factory()->create(['papel' => 'administrador']);
    $this->actingAs($user);

    $professor = Professor::factory()->create([
        'nome_usuario' => 'originaluser',
    ]);

    $newPassword = $this->generateValidPassword();

    $data = [
        'nome' => 'Updated Name',
        'nome_usuario' => 'updateduser',
        'senha' => $newPassword,
        'senha_confirmation' => $newPassword,
        'papel' => 'administrador',
        'email' => 'updated@example.com',
        'numero_celular' => '987654321',
        'descricao' => 'Updated description',
    ];

    $response = $this->put(route('professors.update', $professor->id_professor), $data);

    $response->assertRedirect(route('professorescrud'));
    $response->assertSessionHas('success', 'Professor atualizado com sucesso.');

    $updatedProfessor = Professor::find($professor->id_professor);

    $this->assertTrue(Hash::check($newPassword, $updatedProfessor->senha));
}
```

Fuente: Elaborada por los autores.

Figure 44

Test de integración

```
    /** @test */
    public function administrador_puede_eliminar_profesor()
    {
        Storage::fake('public');

        $user = Professor::factory()->create(['papel' => 'administrador']);
        $this->actingAs($user);

        // Crear un profesor con imagen
        $professor = Professor::factory()->create([
            'url_imagem' => 'uploads/avatar.jpg',
        ]);

        // Crear imagen existente
        Storage::disk('public')->put('uploads/avatar.jpg', 'avatar content');

        $response = $this->delete(route('professors.destroy', $professor->id_professor));

        $response->assertRedirect(route('professorescrud'));
        $response->assertSessionHas('success', 'Professor removido com sucesso.');

        // Verificar que el profesor fue eliminado
        $this->assertDatabaseMissing('professor', ['id_professor' => $professor-
>id_professor]);

        // Verificar que la imagen fue eliminada
        Storage::disk('public')->assertMissing('uploads/avatar.jpg');
    }
```

Fuente: Elaborada por los autores.