



Universidad Católica de Salta
Facultad de Ingeniería
Carrera: Ingeniería en Informática

“Desarrollo de aplicación móvil para la autogestión de alumnos de la
Universidad Católica de Santa María”

Alumno: Pablo Sandoval

Profesor Guía: Fernando German Paredes Marchena

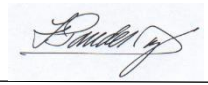
Año: 2024

Título:

Ingeniería en informática

Profesor Guía:

Nombre: Fernando German Paredes Marchena

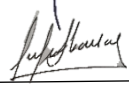
Firma:  _____

Codirectores de proyecto:

Nombre: Toshiro Nagata Bolívar

Firma:  _____

Nombre: Patricia Aballay

Firma:  _____

Tribunal Evaluador:

Nombre: _____

Firma: _____

Nombre: _____

Firma: _____

Nombre: _____

Firma: _____

Alumno:

Nombre: Pablo Sandoval

Firma:  _____

Fecha de Exposición: __/__/____

Agradecimientos:

Quiero agradecer a mi mamá, Silvia Faviana Brizuela, por guiarme, apoyarme y ayudarme a elegir esta carrera, que me permitió ser profesionalmente lo que soy hoy.

A mi papá, Marcelo Jorge Sandoval, por su apoyo y consejo a lo largo de los años.

A mi hermana, María Belén, por estar siempre ahí para mí.

A mis amigos y compañeros de ingeniería, gracias por creer en mí, por alentarme a seguir y ayudarme a no bajar los brazos.

A mis profesores, por instruirme a lo largo de estos años de carrera y darme las oportunidades de formarme.

A Myka, a Ximena, y a mis compañeros/amigos de Perú, por ayudarme y acompañarme durante mi intercambio en Arequipa y ser los “primeros testers” de mi proyecto.

A mi profesor guía, Fernando German Paredes Marchena, por otorgarme la oportunidad de llevar a cabo este proyecto y brindarme las herramientas necesarias.

Por último, a mis codirectores, Patricia Aballay y Toshiro Nagata, por su disponibilidad y ayuda a lo largo del proyecto.

Índice del Trabajo:

Índice de Tablas	6
Índice de Ilustraciones	7
Abstract:	8
Capítulo 1: “Introducción”	9
Capítulo 2: “Estado de la Cuestión”	10
Introducción:	10
Capítulo 3: “Definición del problema”	13
Introducción:	13
Objetivos del Proyecto:	14
Alcance del proyecto:	15
Restricciones del Proyecto:	16
Capítulo 4: “Marco teórico”	18
Capítulo 5: “Solución Propuesta”	27
Introducción:	27
Análisis de requerimientos:	27
Metodología a Implementar:	38
Incremento 1:	38
Incremento 2:	39
Incremento 3:	39
Incremento 4:	40
Incremento 5:	40
Incremento 6:	40
Estructura del proyecto:	41
Front-end:	41
Back-end:	60
Evaluación de Aplicaciones:	73
Evaluación de la Calidad del Software:	74
Planificación del proyecto:	75
Análisis de Costos:	77
Costos de Recursos Humanos:	77
Costos de Hardware:	78
Costos de Software:	78
Costos Totales:	79
Análisis de Factibilidad:	79
Factibilidad Económica:	79

Factibilidad Operativa:.....	79
Factibilidad Técnica:	80
Factibilidad Legal:	80
Capítulo 6: “Resultados”	81
Introducción:.....	81
Pruebas Realizadas:	81
Pruebas Funcionales:	81
Pruebas de Integración:	84
Pruebas de Usabilidad:.....	84
Resultados Esperados:	85
Resultados Obtenidos:.....	86
Comparativa:.....	92
Capítulo 7: “Conclusión”	93
Trabajos Futuros:.....	93
Bibliografía:	94
Anexo 1:	97
Anexo 2:	98

Índice de Tablas

Tabla 1: historia de usuario: “Inicio de Sesión”	28
Tabla 2: historia de usuario: “Acceso a cursos”	29
Tabla 3: historia de usuario: “Horario”	29
Tabla 4: historia de usuario: “Deuda”	29
Tabla 5: historia de usuario: “Avance Académico”	30
Tabla 6: historia de usuario: “Eventos”	30
Tabla 7: historia de usuario: “Recursos Auxiliares”	30
Tabla 8 : historia de usuario: “endpoint auth”	31
Tabla 9 historia de usuario: “endpoint debt”	31
Tabla 10 historia de usuario: “endpoint enrollment”	32
Tabla 11 historia de usuario: “endpoint enrollment/id”	32
Tabla 12 historia de usuario: “endpoint events”	33
Tabla 13 historia de usuario: “endpoint events/id”	33
Tabla 14 historia de usuario: “endpoint semesters”	34
Tabla 15 historia de usuario: “endpoint semesters/id”	34
Tabla 16 historia de usuario: “endpoint subjects”	35
Tabla 17 historia de usuario: “endpoint subjects per enrollment”	35
Tabla 18 historia de usuario: “endpoint subject/id”	36
Tabla 19 historia de usuario: “endpoint subjects remaining”	36
Tabla 20 historia de usuario: “endpoint get statistics”	37
Tabla 21 historia de usuario: “configuración docker”	37
Tabla 22 historia de usuario: “despliegue en render”	37
Tabla 23: “Costos de Recursos Humanos”	78
Tabla 24: “Costos de Hardware”	78
Tabla 25: “Costos Totales”	79
Tabla 26: “Tabla de resultados de la Evaluación de Usabilidad”	86

Índice de Ilustraciones

Ilustración 1: “Grafico comparativo de las diversas categorías de aplicaciones móviles”	10
Ilustración 2 "Repositorio de GitHub de la aplicación Móvil"	42
Ilustración 3 “Estadística de los mobile frameworks Cross-Platform usados por desarrolladores de software en el periodo 2019 a 2022”	42
Ilustración 4: “Representación gráfica del modelo Gitflow”	45
Ilustración 5: “Diseños preliminares de la aplicación realizados en Figma”	46
Ilustración 6: “Resultado final del horario del alumno usando Ionic2-Calendar”	47
Ilustración 7: “Nuevo estudiante de prueba sin avance de carrera presente.”	48
Ilustración 8: “Jerarquía de directorios del proyecto de aplicación móvil”	49
Ilustración 9 “Repositorio de GitHub de la API REST”	60
Ilustración 10 “Servicio Web desplegado en Render”	62
Ilustración 11 “Interfaz de Swagger de API REST”	65
Ilustración 12 "Interfaz de pgAdmin 4"	67
Ilustración 13 "Instancia de base de datos PostgreSQL en Render"	68
Ilustración 14: “Diagrama de la estructura de los contenedores gestionados”	69
Ilustración 15 "Prueba de endpoint login con Postman"	70
Ilustración 16: “Comunicación de los servicios para las pruebas de Integración”	73
Ilustración 17: “diagrama de Gantt del proyecto”	76
Ilustración 18: “diagrama de Gantt del proyecto – análisis, planificación e investigación”	76
Ilustración 19: “diagrama de Gantt del proyecto - desarrollo”	77
Ilustración 20: “diagrama de Gantt del proyecto - Evaluación”	77
Ilustración 21 "Pruebas funcionales con django test"	81
Ilustración 22 "Caso de Prueba de inicio de sesión exitoso"	82
Ilustración 23 "Caso de Prueba de inicio de sesión fallido"	82
Ilustración 24 "Documentación de casos de pruebas en Postman"	83
Ilustración 25 "Casos de pruebas de inicio de sesión en aplicación movil"	83
Ilustración 26 "Carga de datos de prueba para pruebas de integración con aplicación movil"	84
Ilustración 27: “Evaluación de Usabilidad por medio de Google Forms”	85
Ilustración 28: “Grafico de respuestas de la pregunta 1”	87
Ilustración 29: “Grafico de respuestas de la pregunta 2”	87
Ilustración 30: “Grafico de respuestas de la pregunta 3”	88
Ilustración 31: “Grafico de respuestas de la pregunta 4”	88
Ilustración 32: “Grafico de respuestas de la pregunta 5”	89
Ilustración 33: “Grafico de respuestas de la pregunta 6”	89
Ilustración 34: “Grafico de respuestas de la pregunta 7”	90
Ilustración 35: “Grafico de respuestas de la pregunta 8”	90
Ilustración 36: “Grafico de respuestas de la pregunta 9”	91
Ilustración 37: “Grafico de respuestas de la pregunta 10”	91

Abstract:

El siguiente proyecto de grado propone el análisis, diseño y desarrollo de una solución informática orientada a mejorar la experiencia académica y administrativa de los estudiantes de la Universidad Católica de Santa María (UCSM) en Arequipa, Perú. El enfoque se centra en el desarrollo de una aplicación móvil estudiantil que brinde acceso a información personal, selección de cursos, gestión de horarios, calificaciones, notificaciones de eventos, pagos y recursos educativos. Se optó por una metodología incremental para adaptarse a cambios y expansiones funcionales. La aplicación se construirá siguiendo las mejores prácticas de seguridad y escalabilidad, permitiendo una experiencia de usuario óptima en dispositivos móviles y de escritorio. Este proyecto busca presentar un software de calidad que sirva como una alternativa para mejorar la eficiencia y la accesibilidad de los datos en los servicios de la UCSM a través de soluciones tecnológicas innovadoras.

Capítulo 1: “Introducción”

En la última década, la aparición y adopción sustantiva de dispositivos móviles ha instaurado una herramienta integral en diversos contextos de actividad, facilitando la ejecución eficiente y ágil de tareas. Entre esos contextos, se ha destacado en el ámbito educativo las llamadas “aplicaciones de administración académica” o “aplicaciones de gestión estudiantil móvil” enfocadas en mejorar la experiencia de los estudiantes y optimizar la administración académica.

Entre las diversas instituciones que trajeron a su incorporación esta tecnología se incluye a la Universidad Católica de Santa María (UCSM) ubicada en la ciudad de Arequipa, Perú (*Universidad Católica de Santa María – UCSM*, n.d.). Dicha institución fue fundada el 6 de diciembre de 1961 y desde dicha fecha, han estado comprometidos con brindar una educación que supere las expectativas de sus estudiantes y satisfacer todos los intereses de la juventud del país, formando profesionales exitosos en el mundo de hoy (*Escuelas Profesionales - UCSM*, n.d.). A raíz de esto, fue que se llevó a cabo la creación e incorporación del proyecto “UCSM PWA” para que los distintos estudiantes de la universidad accedan al mismo para consultar acerca de su información académica. Sin embargo, cabe destacar que, si bien este tipo de aplicaciones combina características de sitios web y aplicaciones móviles para brindar una experiencia de usuario optimizada tanto en dispositivos de escritorio como móviles, no puede emplear todo el potencial de dichas tecnologías como es por ejemplo la experiencia nativa de aprovechar al máximo las características del dispositivo y acceder a funciones avanzadas.

Actualmente, la información presentada por las aplicaciones es recuperada y proporcionada a estas por medio de Interfaces de programación de aplicaciones disponibles en servicios en la nube o servidores locales. Esto si bien se asume como prioridad en el desarrollo de sistemas descentralizados, se debe contemplar en las mismas cualidades a menudo conocidas como atributos de calidad siendo estos: robustez, escalabilidad, rendimiento, usabilidad, mantenibilidad, seguridad entre otros (Dorfman & Thayer, 1990). A partir de esta premisa, los proyectos que buscan disponer de estas características siguen prácticas y enfoques que a menudo se basan en patrones y estándares de desarrollo marcados por los mismos profesionales u organismo referentes alrededor del mundo.

A partir de esto y reconociendo la creciente importancia de la tecnología en la educación moderna, se propone el desarrollo de una aplicación móvil estudiantil que complemente y mejore la experiencia educativa y administrativa de los estudiantes de la UCSM como así también la construcción de servicios que complemente su funcionalidad siguiendo estándares de calidad.

Capítulo 2: “Estado de la Cuestión”

Introducción:

A lo largo de la última década, el panorama tecnológico en el Perú ha experimentado transformaciones notables, especialmente en el ámbito de la tecnología móvil. Esta evolución se ha visto acompañada por la creciente adopción de aplicaciones móviles que, en numerosas ocasiones, han demostrado ser herramientas de gran utilidad en diversos segmentos de la sociedad. De manera particular, diversas instituciones educativas en el país han emprendido el camino del desarrollo e implementación de aplicaciones móviles con objetivos educativos específicos, buscando satisfacer las necesidades particulares de sus comunidades académicas.

Este proceso de integración de aplicaciones móviles se acentuó aún más con la aparición de la pandemia de COVID-19 a principios de 2020. La crisis sanitaria provocó una rápida transición hacia modalidades de enseñanza y trabajo a distancia, lo que aumentó la relevancia de las herramientas digitales, incluidas las aplicaciones móviles, para mantener la continuidad de las actividades habituales del mundo físico. La demanda de soluciones tecnológicas que permitieran una gestión eficiente de las actividades académicas se volvió imperativa en un contexto donde la interacción presencial se vio limitada, llegando a tener un aumento del 20% en su uso diario (*Q1 Data: The Impact of COVID-19 on Daily App Usage* - *CleverTap*, n.d.). Es por eso por lo que se denota como las aplicaciones móviles han emergido como una vía eficaz para la entrega de contenidos digitales que enriquecen diversos procesos como lo son la enseñanza y el aprendizaje, ampliando las posibilidades de interacción y participación de los estudiantes (Slade, 2020). Esta comparativa se puede observar detalladamente en la ilustración 1 planteada por Sensor Tower (Perez, 2022):

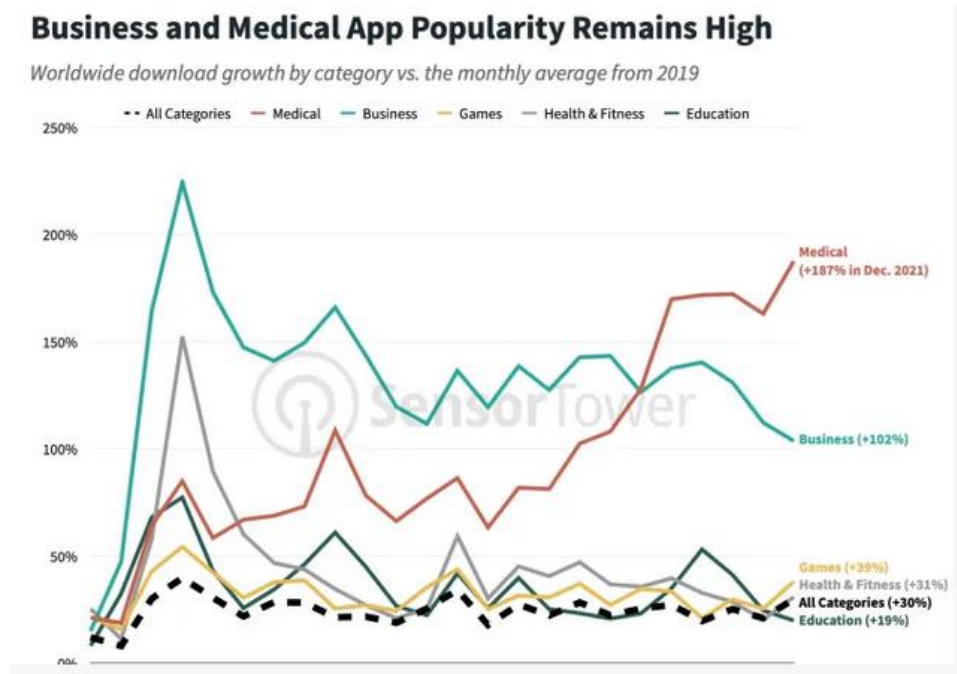


Ilustración 1: “Grafico comparativo de las diversas categorías de aplicaciones móviles”

En el contexto peruano, se ha dado un constante esfuerzo por avanzar en el ámbito de la tecnología móvil, visible en iniciativas como la implementación de servicios de banca móvil por parte de instituciones financieras líderes, como el Banco de Crédito del Perú (BCP) (*Banca Móvil BCP - Apps*

En Google Play, n.d.). A través de estas iniciativas, los usuarios pueden acceder a una gama de servicios financieros desde sus dispositivos móviles inteligentes, permitiéndoles realizar consultas, transacciones y pagos de manera ágil y cómoda, sin la necesidad de desplazarse a una agencia bancaria física.

Sin embargo, en el sector educativo, a pesar de los avances que se están generando en la incorporación de tecnologías móviles, el país se encuentra aún en una fase de transición. Un ejemplo ilustrativo de esta realidad se observa en los sitios web de varias universidades, donde los métodos tradicionales para el control y registro de calificaciones aún prevalecen. Esta situación contrasta con la tendencia actual de aprovechar aplicaciones móviles, lo cual puede ocasionar dificultades en el acceso a sistemas y en la realización de trámites académicos en línea.

Ejemplos como la Universidad Nacional de San Agustín (UNSA), reflejan un creciente interés en explorar la viabilidad de desarrollar aplicaciones móviles que optimicen la experiencia estudiantil al brindar un acceso eficiente a información académica esencial, como calificaciones y registros académicos. Otra institución (y la cual es objeto de estudio a lo largo del presente trabajo) es la Universidad Católica de Santa María (UCSM) que, en su afán por mantenerse a la vanguardia en términos tecnológicos y mejorar la experiencia de sus estudiantes, ha sido una de las instituciones que ha reconocido la creciente importancia de las aplicaciones móviles, especialmente en el contexto postpandemia. A medida que el mundo se enfrentó a los desafíos del distanciamiento social y la educación a distancia, la UCSM demostró una rápida capacidad de adaptación al lanzar proyectos innovadores que optimizaron la autogestión estudiantil como fue el aula virtual (*UCSM Inicia El Año Académico 2020 Utilizando Aula Virtual – Universidad Católica de Santa María, n.d.*).

Otro de los proyectos destacados, ya en el área multiplataforma, fue la implementación de una Progressive Web App (PWA) destinada a los estudiantes. Esta es una aplicación web que ofrece una experiencia similar a la de una aplicación móvil nativa, pero se puede acceder a través de un navegador web; por ende, no es necesario descargar e instalar una aplicación desde una tienda de aplicaciones, lo que simplifica el proceso de acceso para los estudiantes.

La PWA diseñada por la UCSM fue concebida para brindar un acceso eficiente a información académica, recursos y servicios relevantes para los estudiantes, en un formato optimizado para dispositivos móviles. Esta iniciativa no solo facilitó la autogestión estudiantil, sino que también ofreció una experiencia más integrada y cómoda en comparación con los métodos tradicionales.

Sin embargo, cabe destacar que las aplicaciones PWA presentan ciertas limitaciones a comparación de desarrollos nativos como es acceder a funcionalidades y características propias del dispositivo. Por dicho motivo, el presente año 2023, la UCSM lanzó la "UCSM App", una aplicación móvil dedicada exclusivamente a la comunidad estudiantil. Esta aplicación fue diseñada para abordar diversas necesidades estudiantiles, como la consulta de calificaciones, horarios de clases, comunicaciones institucionales y acceso a recursos académicos. La UCSM App permitió a los estudiantes gestionar su vida académica de manera más eficiente y conveniente, y sirvió como un ejemplo de cómo las aplicaciones móviles pueden mejorar la autogestión estudiantil.

Estos proyectos no solo reflejaron el compromiso de la UCSM con la innovación tecnológica, sino que también respondieron directamente a la necesidad de proporcionar soluciones eficaces en un contexto en el que la demanda de autogestión estudiantil se intensificó desde la pandemia. La universidad demostró su capacidad para adaptarse rápidamente a las circunstancias cambiantes y brindar a los estudiantes herramientas que facilitaron su involucramiento y éxito académico.

Es importante resaltar que ambas aplicaciones, tanto la PWA como la UCSM App, consumen servicios a través de peticiones HTTP, los cuales son creados por el área de Informática de la universidad y el ERP (área que se encarga de desarrollar soluciones informáticas integrales para la universidad). Además, en términos de seguridad, tanto la base de datos como los servicios implementados son sujetos a rigurosas medidas de seguridad, con el objetivo de proteger la confidencialidad y la integridad de los

datos estudiantiles. Entre estas, la institución ha adoptado soluciones de seguridad como Palo Alto Networks (*Leader in Cybersecurity Protection & Software for the Modern Enterprises - Palo Alto Networks*, n.d.), para salvaguardar la infraestructura y los sistemas contra amenazas cibernéticas. Estas soluciones ofrecen una capa adicional de seguridad, garantizando que los datos de los estudiantes estén protegidos en todo momento y que los servicios brindados mantengan los más altos estándares de seguridad informática.

Capítulo 3: “Definición del problema”

Introducción:

En los últimos años, el panorama educativo a nivel mundial ha experimentado una transformación sin precedentes, impulsada en gran medida por dos factores interrelacionados: la pandemia de COVID-19 y los avances tecnológicos. En este contexto, las instituciones educativas, incluida la Universidad Católica de Santa María (UCSM) en Arequipa, Perú, se han visto obligadas a adaptarse a una nueva realidad que ha redefinido la enseñanza y la administración académica.

La pandemia de COVID-19, que comenzó a principios de 2020, impuso restricciones significativas a la interacción presencial, lo que llevó a un cambio masivo hacia la educación a distancia y el trabajo remoto. La UCSM, como muchas otras universidades, enfrentó el desafío de mantener la continuidad de las actividades académicas y administrativas en un entorno virtual. Fue necesario encontrar soluciones tecnológicas eficaces para asegurar la conectividad, la gestión académica y la administración eficiente en un mundo en rápida transformación.

En este contexto, las soluciones informáticas, como el Sistema de Planificación de Recursos Empresariales (ERP), desempeñaron un papel fundamental en la capacidad de la UCSM para adaptarse a las nuevas circunstancias y evolucionar para satisfacer las necesidades emergentes. El ERP, que ya estaba en uso antes de la pandemia, se convirtió en una columna vertebral digital esencial para la gestión de la universidad.

Durante la pandemia, el ERP se sometió a una profunda revisión y expansión para abordar las demandas cambiantes. Se digitalizaron y automatizaron aún más procesos académicos y administrativos, permitiendo a la UCSM operar de manera eficiente en un entorno virtual. La gestión de recursos, desde la planificación de cursos hasta la asignación de aulas virtuales, se optimizó a través del ERP. La administración de registros académicos y financieros se volvió más ágil y transparente, lo que facilitó la toma de decisiones informadas en tiempos inciertos.

Además, se implementaron sistemas de registro de proveedores y otros procesos administrativos que mejoraron la eficiencia y la transparencia en la gestión de recursos y servicios. Estas soluciones permitieron a la UCSM mantener el suministro de recursos críticos y servicios esenciales, incluso en un entorno virtual.

Sin embargo, a medida que avanzaba la pandemia y se consolidaba la educación a distancia, surgió la necesidad de brindar a los estudiantes y al personal académico un acceso más conveniente a la información y los servicios. Una de estas soluciones fue el desarrollo y lanzamiento de la Progressive Web App (PWA) orientada a la autogestión académica como parte de su enfoque para mejorar la experiencia estudiantil y la administración académica, facilitando el acceso a la información y a la autonomía de los estudiantes.

No obstante, es importante destacar que, si bien las PWAs ofrecen numerosas ventajas, como la accesibilidad desde navegadores web sin necesidad de instalación, presentan limitaciones en comparación con las aplicaciones móviles nativas. Estas limitaciones incluyen un acceso más restringido a funciones avanzadas del dispositivo, como sensores de hardware, cámaras y sistemas de geolocalización, lo que puede limitar algunas capacidades interactivas y experiencias personalizadas. Además, aunque las PWAs ofrecen un buen rendimiento en la mayoría de los casos, las aplicaciones móviles nativas suelen ser más rápidas y receptivas, especialmente en dispositivos más antiguos o con recursos limitados. Las notificaciones push también pueden ser más limitadas en PWAs en comparación con aplicaciones nativas, lo que puede afectar la capacidad de una institución educativa para comunicarse de manera efectiva con sus estudiantes. Además, las PWAs pueden tener restricciones en la ejecución en segundo plano, lo que significa que algunas funciones, como la actualización de

contenido en tiempo real, pueden ser menos eficientes en comparación con aplicaciones nativas. Finalmente, las aplicaciones nativas a menudo ofrecen un mayor control sobre el almacenamiento de datos y la capacidad de funcionar sin conexión a Internet, lo que puede ser fundamental en entornos educativos. Como resultado de estas limitaciones y reconociendo la creciente importancia de la tecnología en la educación moderna, se inició, en el presente año 2023, el proyecto de la UCSM App, una aplicación móvil nativa desarrollada con Angular Ionic, que culminó con la presentación de un MVP (Producto Mínimo Viable) en la App Store. Esta iniciativa busca superar las restricciones inherentes a las PWAs y proporcionar a estudiantes y personal académico una experiencia más enriquecedora y completa en términos de funcionalidad y rendimiento.

El proyecto en cuestión se caracteriza por la integración de dos servicios web previamente desarrollados con tecnologías PHP, cada uno de los cuales es administrado por diferentes áreas de la Universidad. Esta iniciativa busca mejorar y optimizar la experiencia académica de los estudiantes, específicamente a través de la creación de un sistema de autogestión estudiantil que cumpla con los estándares de seguridad y accesibilidad requeridos.

El primer servicio web, desarrollado por el área de informática de la universidad, se distingue por su acceso a servidores de bases de datos que almacenan información detallada de los alumnos. Este acceso a datos precisos resulta esencial para implementar funcionalidades específicas en el aplicativo móvil.

Por otro lado, el segundo servicio web fue creado por el propio equipo de desarrollo del ERP de la universidad y tiene como finalidad cubrir ciertas funcionalidades fundamentales, como la autenticación de usuarios y la recuperación de datos, teniendo diferentes niveles de acceso a la base de datos gestionada por el área de informática. Ambos servicios se basan en el protocolo de comunicación HTTP (Hypertext Transfer Protocol) para facilitar la comunicación y el intercambio de datos. En esencia, estos servicios actúan como la "interfaz" crucial entre los datos almacenados y el aplicativo móvil, permitiendo la interacción efectiva entre los estudiantes y la información académica.

La necesidad de este proyecto se sustenta en la retroalimentación proporcionada por los alumnos de la Facultad de Ingeniería de Sistemas de la universidad durante el inicio del semestre impar del año 2023. Los estudiantes identificaron aspectos clave a mejorar, tales como la necesidad de una interfaz más dinámica y una navegación más sencilla y comprensible para los usuarios. Además, expresaron su interés en estar informados sobre eventos y actividades que se lleven a cabo en la universidad.

Además de la retroalimentación de los usuarios, el equipo de desarrollo reconoció la oportunidad de consolidar los servicios existentes en un único servicio integral que ofrezca todas las funcionalidades actuales. Sin embargo, este nuevo servicio debe destacarse por su calidad y rendimiento, cumpliendo con atributos de calidad definidos que aseguren su eficacia y eficiencia en su funcionamiento.

Basado en todos estos aspectos y consideraciones, el objetivo principal de este proyecto es la creación de un sistema de autogestión estudiantil que cumpla con rigurosos estándares de seguridad y accesibilidad. Este sistema, diseñado para su implementación en la Universidad Católica de Santa María, tiene como propósito principal permitir que los estudiantes gestionen y accedan a diversos aspectos relacionados con su experiencia académica a través de dispositivos móviles. Con un enfoque centrado en la mejora de la experiencia del usuario y la optimización de la calidad del servicio, este proyecto se convierte en un hito significativo en la modernización y eficiencia de la gestión académica en la universidad.

Objetivos del Proyecto:

El objetivo primordial de este proyecto es desarrollar e implementar una aplicación móvil estudiantil de vanguardia para la UCSM. Esta aplicación proporcionará una plataforma integral para que los estudiantes accedan y gestionen su información académica y administrativa de manera eficiente a través de dispositivos móviles.

Con el propósito de esclarecer de manera detallada los objetivos que este proyecto se propone cumplir durante su desarrollo, a continuación, se enunciarán de manera explícita:

- **Diseñar Mockups:** El primer objetivo consiste en la creación de "Mockups". En el contexto del diseño y desarrollo de aplicaciones y sitios web, los "Mockups" son representaciones visuales estáticas y de baja fidelidad de la interfaz de usuario que se espera obtener como producto final para la Universidad Católica de Santa María. Estos Mockups servirán como una representación gráfica preliminar de la futura interfaz de usuario de la aplicación, permitiendo una visualización temprana de su diseño y funcionalidad.
- **Diseñar la Arquitectura:** El segundo objetivo se enfoca en el diseño de la arquitectura que se seguirá para la construcción de los servicios que serán consumidos por la aplicación. Este paso crucial implica la planificación detallada de la estructura de la aplicación, incluyendo la definición de componentes, flujos de datos y la lógica subyacente. Una arquitectura bien diseñada proporcionará una base sólida para el desarrollo eficiente y escalable del proyecto.
- **Desarrollar un API REST:** El tercer objetivo se refiere al desarrollo de un API REST que tenga la capacidad de acceder a la base de datos del sistema de matrículas de la Universidad Católica de Santa María y devolver información de manera segura, eficiente y transparente. Para lograrlo, se seguirán los estándares de REST Security de OWASP como referencia para su implementación. Este API servirá como el puente de comunicación entre la aplicación móvil y la fuente de datos, garantizando la confidencialidad y la integridad de la información.
- **Desarrollar una Aplicación Móvil:** El cuarto objetivo se concentra en el desarrollo de un aplicativo móvil diseñado para la autogestión estudiantil. Esta aplicación permitirá a los estudiantes llevar un seguimiento de diversos aspectos académicos, aprovechando la accesibilidad y movilidad que ofrece una aplicación nativa en el dispositivo. La aplicación móvil será diseñada para ser intuitiva y fácil de usar, garantizando una experiencia de usuario óptima.
- **Evaluar los Beneficios de la Alternativa Desarrollada:** Este objetivo se centra en la evaluación de la usabilidad y las ventajas que ofrece la nueva solución propuesta en comparación con el sistema actualmente disponible en la universidad. El análisis se realizará utilizando diversas técnicas y métricas para destacar la eficiencia, seguridad y beneficios que la alternativa desarrollada aporta, sin enfocarse en la sustitución del sistema existente.

Estos objetivos, debidamente establecidos, guiarán el desarrollo del proyecto, asegurando que se alcancen los resultados deseados de manera eficiente y efectiva. Cada uno de estos pasos contribuirá al éxito de la implementación del sistema de autogestión estudiantil en la Universidad Católica de Santa María.

Alcance del proyecto:

El alcance del trabajo a desarrollar se encuentra limitado por las siguientes actividades. En primer lugar, el objetivo principal del proyecto propuesto es definir y desarrollar una solución informática integral que haga un uso adecuado de los recursos disponibles, tanto en términos de seguridad como de servicios. Esta solución se concibe como un elemento fundamental para satisfacer las necesidades específicas de la Universidad y mejorar su eficiencia operativa.

En las etapas de definición y diseño del sistema, se hará hincapié en la comprensión de las medidas y servicios de seguridad que dispone la universidad, así como del sistema que actualmente está en funcionamiento. Este conocimiento es esencial para garantizar la protección de los datos y la integridad de la infraestructura durante todo el proceso de desarrollo y operación.

Además, se llevará a cabo un análisis teórico y fundamentado sobre las razones por las cuales seguir los estándares de OWASP puede garantizar un sistema seguro y robusto. Se explorarán las mejores prácticas

de seguridad recomendadas por OWASP con el objetivo de garantizar que el sistema resultante sea seguro y robusto. Esto abarcará desde la identificación de vulnerabilidades potenciales hasta la implementación de medidas preventivas y correctivas.

Asimismo, se buscará seguir estrategias escalables y de rendimiento en el desarrollo, propias de los frameworks tanto del aplicativo móvil como el Back End. Estas estrategias permitirán que el sistema sea capaz de gestionar un crecimiento sostenible en el número de usuarios y datos sin comprometer su funcionamiento. Se dará especial atención a la calidad del software en futuras etapas para garantizar su continuidad y mejora a lo largo del tiempo.

Este enfoque meticuloso y riguroso, basado en las actividades mencionadas y conectado mediante diversos puntos, establece las bases para la implementación exitosa de una solución informática que cumple con contempla aspectos de seguridad y calidad para la Universidad en cuestión.

Para este proyecto, se desarrollarán tres versiones del prototipo, cada una con un propósito específico:

1. Versión con la API oficial de la universidad: Esta versión conformada con el apk (instalador de la aplicación compilada para plataformas Android) y haciendo consumo del servicio actual desarrollado en PHP, se utilizará para realizar pruebas de usabilidad, aprovechando los datos y servicios proporcionados por la universidad.
2. Versión con la API y base de datos desplegados en Render: Esta versión permitirá probar el servicio web desarrollado en un entorno de nube, utilizando Render para hospedar tanto la API como la base de datos, ofreciendo una solución más cercana a un entorno de producción. Y que la misma aplicación instalada del apk consuma de este.
3. Versión local con API armada en Docker Compose: Para facilitar el desarrollo y pruebas locales, esta versión utilizará Docker Compose para configurar un entorno con dos contenedores: uno para la base de datos PostgreSQL y otro para el servicio web.

Estas versiones permitirán evaluar el desempeño y la funcionalidad del sistema en diferentes entornos, asegurando que se identifiquen y aborden posibles problemas antes de una implementación final.

Restricciones del Proyecto:

El proyecto se encuentra sujeto a ciertas restricciones que deben ser tenidas en cuenta durante su ejecución:

- **Gestión de los recursos financieros en base a un presupuesto limitado:** Uno de los principales desafíos reside en el presupuesto asignado para este proyecto, el cual es limitado y, por lo tanto, debe ser gestionado con la máxima eficiencia posible. Cada gasto debe ser cuidadosamente evaluado para garantizar que se utilicen los recursos de manera óptima. Para la construcción de un producto viable mínimo (o MVP en sus siglas del inglés) se optarán por emplear tecnologías open source o tecnologías con planes gratuitos para mitigar costos. Estos serán detallados en la sección de análisis de costos.
- **Restricción de Tiempo:** El proyecto está sujeto a plazos definidos que deben ser cumplidos rigurosamente. Estos plazos se establecen con el propósito de asegurar la disponibilidad oportuna de la aplicación para los usuarios finales. Es fundamental mantener un cronograma detallado y garantizar que cada fase del proyecto se complete dentro de los plazos estipulados.
- **Acceso a Datos de Estudiantes:** Una de las limitaciones críticas del proyecto fue la imposibilidad de acceder a los datos reales de los estudiantes. Esta restricción se debe a la necesidad de proteger la privacidad y cumplir con las normativas de protección de datos. Como resultado, el desarrollo del servicio se llevó a cabo sin la posibilidad de realizar pruebas con

información auténtica, lo que limitó la capacidad para validar completamente las funcionalidades del sistema en un entorno real.

- **Acceso a la Base de Datos Institucional:** Otra restricción significativa fue la falta de acceso a la base de datos institucional. Esta limitación impidió comprender la estructura y las relaciones entre las diversas entidades del sistema actual. Dado que el acceso a esta información es crítico para el diseño de un sistema que refleje adecuadamente las necesidades de la universidad, se optó por crear una estructura propia. Esta alternativa permitió emular un sistema aceptable para el desarrollo y las pruebas, aunque con las limitaciones inherentes de no basarse en datos y estructuras reales.
- **Cumplimiento Normativo:** El proyecto está sujeto al cumplimiento de regulaciones y normativas, en particular en lo que respecta a la privacidad de datos y cualquier otra normativa aplicable. Esto implica que se deben implementar medidas de seguridad y privacidad robustas para proteger la información de los estudiantes y cumplir con las leyes vigentes. Esto se aborda en detalle en la sección de factibilidad legal.
- **Restricción de Servidores Universitarios:** No es posible utilizar los servidores de la universidad para desplegar el servicio. Esta limitación obliga a buscar alternativas externas para el hosting y despliegue, lo que puede influir en la latencia, disponibilidad y costos operativos del servicio. Se debe garantizar que la solución escogida cumpla con los requerimientos de rendimiento y seguridad necesarios.
- **Restricción de Publicación en Tiendas de Aplicaciones:** No se puede subir la aplicación a las tiendas de Android y iOS debido a las etapas de evaluación requeridas y a los costos asociados. Esto limita el acceso de los usuarios finales a la aplicación, restringiendo su distribución a métodos alternativos que no impliquen los procesos formales de publicación en estas plataformas.
- **Restricción de Pruebas en iOS:** No se pueden realizar pruebas en dispositivos iOS debido a la falta de acceso a dichos dispositivos. Esto limita las pruebas a simulaciones en entornos de desarrollo y dispositivos Android, lo que puede afectar la identificación de problemas específicos de la plataforma iOS y la optimización de la experiencia de usuario en dicha plataforma.

Para llevar a cabo este proyecto de manera exitosa, se requieren recursos específicos que incluyen:

- **Infraestructura Tecnológica:** La infraestructura tecnológica, que abarca servidores, bases de datos y sistemas de respaldo, es esencial para almacenar y gestionar los datos de los estudiantes de manera segura y eficiente. Esta infraestructura debe estar dimensionada adecuadamente para soportar la carga de usuarios y garantizar la disponibilidad de la aplicación.
- **Acceso a Datos Académicos:** Colaboración con los departamentos académicos de la Universidad Católica de Santa María (UCSM) para obtener acceso a los datos académicos necesarios. Esta colaboración es fundamental para garantizar la precisión y actualización de la información que se mostrará en la aplicación.
- **Presupuesto:** Se requerirán de diversos recursos para financiar todas las etapas del proyecto, incluyendo desarrollo, pruebas y lanzamiento de la aplicación. Entonces, es esencial desarrollar un presupuesto que exprese todos los costos involucrados para la ejecución exitosa del proyecto.

La consideración y gestión efectiva de estas restricciones y recursos son cruciales para asegurar el éxito del proyecto de desarrollo de la aplicación de autogestión estudiantil en la UCSM.

Capítulo 4: “Marco teórico”

En la presente sección, se proporcionará una explicación detallada de los diversos conceptos, metodologías, herramientas y tecnologías que serán empleados a lo largo de este proyecto. Estos elementos desempeñarán un papel fundamental en la planificación, desarrollo e implementación de la solución de autogestión estudiantil que se busca crear para la Universidad Católica de Santa María (UCSM). Cada uno de estos componentes desempeñará un papel crucial en la consecución de los objetivos generales y específicos del proyecto, garantizando que se sigan prácticas de calidad, seguridad y accesibilidad en el diseño y la ejecución de la aplicación móvil y los servicios web asociados:

Django:

Es un framework de desarrollo web de alto nivel, gratuito y de código abierto, desarrollado en el lenguaje de programación de alto nivel Python. El mismo fue desarrollado por Adrián Holovaty, Simón Wilson y Jacob Kaplan-Moss en 2003 (Holovaty & Kaplan-Moss, 2008).

Django es un framework de desarrollo web ampliamente reconocido que destaca por sus diversas cualidades, que lo convierten en una herramienta valiosa para construir aplicaciones web de alta calidad. Estos son destacados en su documentación según (Django Software Foundation, 2022)

Uno de los principios fundamentales de Django es el concepto de "Don't Repeat Yourself" (DRY), que promueve la eficiencia al minimizar la duplicación de código. Gracias a DRY, los desarrolladores pueden escribir menos código y, al mismo tiempo, mantener una base de código más limpia y fácil de mantener.

El enfoque de Django en el patrón de diseño MTV (Modelo-Plantilla-Vista) proporciona una estructura organizativa clara para las aplicaciones web. Esto separa la lógica de la aplicación en tres componentes distintos, lo que facilita la comprensión y la colaboración en el desarrollo de proyectos.

Una de las principales ventajas de Django es su robusta colección de características "baterías incluidas". Esto significa que Django proporciona un conjunto completo de herramientas y módulos listos para usar para realizar tareas comunes, como autenticación de usuarios, administración de bases de datos y generación de formularios. Esto acelera el desarrollo y ahorra tiempo a los desarrolladores.

Este también es conocido por su enfoque en la seguridad. Viene con protecciones integradas contra amenazas comunes, como ataques de inyección SQL y ataques CSRF (Cross-Site Request Forgery). Esto permite a los desarrolladores crear aplicaciones web robustas y seguras desde el principio.

Django Rest Framework:

Django REST Framework (DRF) es una potente herramienta para desarrollar aplicaciones web y APIs REST en Django. Ofrece a los desarrolladores las herramientas necesarias para crear APIs REST de forma eficiente, pero es importante destacar que DRF puede manejar niveles de madurez de API menos desarrollados.

Lo que distingue a Django REST Framework es que no impone ningún diseño de nivel de hipermedia específico, lo que significa que no dicta cómo debe estructurarse un API con HATEOAS (Hypermedia as the Engine of Application State). En su lugar, brinda a los desarrolladores la flexibilidad para diseñar sus APIs según sus necesidades específicas. (Stack Overflow Contributors, 2019)

DRF permite a los desarrolladores implementar fácilmente APIs RESTful con características avanzadas, como autenticación, autorización, serialización de datos y enrutamiento. También ofrece un sólido conjunto de herramientas para trabajar con vistas basadas en clases y funciones, lo que facilita la creación de APIs bien organizadas y de alto rendimiento.

PostgreSQL:

PostgreSQL es un sistema de gestión de bases de datos relacional y de objeto de código abierto, conocido por su robustez, escalabilidad y cumplimiento con los estándares SQL. Fue creado por el profesor Michael Stonebraker y su equipo en la Universidad de California, Berkeley, como parte del proyecto POSTGRES en 1986. Desde entonces, ha evolucionado y se ha mantenido gracias a una comunidad activa de desarrolladores y colaboradores de todo el mundo. Una ventaja significativa es su capacidad para gestionar grandes volúmenes de datos y realizar consultas complejas de manera eficiente. PostgreSQL incluye características avanzadas como replicación, índices avanzados (incluyendo índices parciales y expresiones), y soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que garantiza la integridad y confiabilidad de los datos. (The PostgreSQL Global Development Group, n.d.)

Angular:

Angular es un framework de desarrollo de aplicaciones web front-end altamente potente y versátil. Su objetivo principal es simplificar y agilizar la creación de aplicaciones web robustas, dinámicas y de alta calidad. El mismo fue desarrollado y es respaldado por Google. (Murray et al., 2020)

Angular ofrece una serie de ventajas significativas en la creación de aplicaciones web modernas y robustas como es por ejemplo su arquitectura que, estando basada en componentes, facilita la construcción de aplicaciones modulares y reutilizables, lo que resulta en un desarrollo más eficiente y mantenible.

Otra de las características distintivas de Angular es su uso de Typescript, un superconjunto tipado de JavaScript. Esta elección proporciona una verificación de tipos estática, lo que mejora significativamente la seguridad y la confiabilidad del código. Además, Typescript brinda a los desarrolladores funciones de autocompletado y ayuda en tiempo real, lo que agiliza el proceso de desarrollo.

Angular también introduce el concepto de inyección de dependencias, lo que simplifica la gestión de dependencias en una aplicación. Esto facilita la creación de pruebas unitarias efectivas y permite un mayor modularidad en el código, lo que a su vez mejora la capacidad de mantenimiento.

En cuanto a la navegación y el enrutamiento, Angular ofrece un enrutador integrado que simplifica la creación de rutas y la navegación entre vistas y componentes. Esto es esencial para construir aplicaciones web de una sola página (SPA) y garantiza una experiencia de usuario fluida.

Angular se beneficia de una comunidad activa respaldada por Google. Esto se traduce en una abundante documentación, recursos en línea y una amplia variedad de módulos de terceros disponibles para extender la funcionalidad de la aplicación.

Por último, en el proceso de desarrollo, Angular se apoya en herramientas poderosas como Angular CLI, que simplifica la creación, prueba y construcción de aplicaciones. Además, el framework tiene un fuerte enfoque en la seguridad, con medidas incorporadas para proteger contra amenazas comunes como ataques de script entre sitios (XSS) y falsificación de solicitudes entre sitios (CSRF).

Ionic:

Ionic es un framework de desarrollo de aplicaciones móviles híbridas que ofrece numerosas ventajas para la creación de aplicaciones multiplataforma de alto rendimiento. Este framework se destaca por su enfoque en la construcción de aplicaciones móviles utilizando tecnologías web estándar como HTML, CSS y JavaScript (*Ionic Docs - Ionic Documentation*, n.d.).

Una de las ventajas clave de Ionic es su capacidad para desarrollar aplicaciones que se ejecutan tanto en dispositivos iOS como Android, lo que ahorra tiempo y recursos al eliminar la necesidad de crear

aplicaciones separadas para cada plataforma. Esto se logra mediante el uso de una única base de código compartida.

Ionic también se beneficia de Angular, un popular framework de desarrollo front-end. Esta integración permite a los desarrolladores aprovechar las ventajas de Angular al construir aplicaciones móviles, incluido el enlace de datos bidireccional y la inyección de dependencias. Además, Ionic utiliza Typescript como lenguaje de programación, lo que brinda a los desarrolladores un sistema de tipos estáticos y herramientas de desarrollo eficientes.

La interfaz de usuario en Ionic se basa en componentes predefinidos que siguen las pautas de diseño nativas de iOS y Android, lo que garantiza una apariencia y una experiencia de usuario consistentes en ambas plataformas. Además, Ionic ofrece un conjunto de herramientas de diseño y personalización para adaptar la interfaz de usuario según las necesidades específicas de cada aplicación.

Otra ventaja significativa de Ionic es su enfoque en la rapidez de desarrollo. Con el uso de Ionic CLI, los desarrolladores pueden crear rápidamente proyectos, generar componentes y previsualizar aplicaciones en tiempo real en dispositivos móviles. Esto acelera el proceso de desarrollo y permite una iteración más rápida.

Capacitor:

Capacitor es una plataforma de desarrollo de código abierto diseñada para facilitar la creación de aplicaciones móviles multiplataforma utilizando tecnologías web estándar, como HTML, CSS y JavaScript. Esta herramienta permite a los desarrolladores ampliar su alcance y crear aplicaciones móviles para iOS, Android y la web de manera eficiente y efectiva (*Capacitor - Cross-Platform Native Runtime for Web Apps* | *Capacitor Documentation*, n.d.).

Una de las ventajas más destacadas de Capacitor es su estrecha integración con los populares frameworks Angular e Ionic. Esto permite a los desarrolladores aprovechar al máximo las características y capacidades de estas tecnologías en la creación de aplicaciones móviles nativas o progresivas.

En el contexto de Angular, Capacitor ofrece una integración fluida. Los desarrolladores pueden utilizar Angular para construir la parte web de sus aplicaciones y luego, con la ayuda de Capacitor, envolver esta aplicación Angular en una aplicación nativa para dispositivos móviles. Esto permite utilizar un solo conjunto de habilidades y código base para crear aplicaciones tanto web como móviles, lo que ahorra tiempo y recursos significativos.

Cuando se combina con el framework Ionic, Capacitor brinda una experiencia de desarrollo aún más completa. Ionic proporciona componentes y herramientas de diseño listas para usar que ayudan a crear interfaces de usuario atractivas y consistentes en dispositivos móviles. Capacitor y Ionic trabajan juntos de manera armoniosa, lo que significa que los desarrolladores pueden crear aplicaciones móviles con una apariencia nativa utilizando las potentes herramientas de diseño de Ionic y aprovechar las funcionalidades nativas a través de Capacitor.

Docker:

Docker es una plataforma de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software. Fue desarrollada por Solomon Hykes y lanzada por la empresa Docker, Inc. en 2013. Los contenedores de Docker permiten empaquetar una aplicación con todas sus dependencias, lo que asegura que se ejecute de manera uniforme y consistente en cualquier entorno. (*Docker: Accelerated Container Application Development*, n.d.)

Una de las principales ventajas de Docker es su capacidad para facilitar la portabilidad y escalabilidad de aplicaciones. Los contenedores son ligeros y rápidos de desplegar, lo que permite a los desarrolladores y equipos de operaciones trabajar más eficientemente. Además, Docker optimiza el uso

de recursos del sistema al compartir el kernel del sistema operativo entre los contenedores, lo que reduce la sobrecarga en comparación con las máquinas virtuales tradicionales.

Docker Compose:

Docker Compose es una herramienta de Docker que permite definir y gestionar aplicaciones multi-contenedor. Fue desarrollada como parte del ecosistema Docker y lanzada en 2014. Docker Compose facilita la orquestación de contenedores, permitiendo a los desarrolladores y equipos de operaciones definir y ejecutar aplicaciones complejas de manera simple y eficiente mediante archivos de configuración.

Una de las principales ventajas de Docker Compose es su capacidad para simplificar la configuración y el despliegue de entornos de desarrollo y producción. Mediante un solo archivo YAML, conocido como `docker-compose.yml`, se pueden definir múltiples servicios, redes y volúmenes que una aplicación requiere para funcionar. Esto incluye especificar las imágenes de Docker, las variables de entorno, los puertos expuestos y las dependencias entre servicios. (*Docker Compose Overview* | *Docker Docs*, n.d.)

Render:

“Render es una plataforma de software basada en la nube muy popular que se encarga de compilar, desplegar código y proporcionar recursos para alojar aplicaciones y servicios” (*Cloud Application Hosting for Developers* | *Render*, n.d.). Con Render, se puede desplegar rápidamente un prototipo funcional de una aplicación para usuarios potenciales.

Figma:

Figma es una herramienta de diseño de interfaces y prototipado basada en la web, que permite a los diseñadores crear, colaborar y compartir diseños de manera eficiente. Ofrece una amplia gama de características, como diseño vectorial, creación de prototipos interactivos, bibliotecas de componentes reutilizables y la posibilidad de integrarse con otras herramientas y plataformas (*Figma: La Herramienta de Diseño de Interfaz Colaborativo*, n.d.). Para el presente proyecto, ha sido empleado para el diseño de las pantallas de la aplicación móvil (Detallado en la sección de Diseño).

PrimeFlex:

PrimeFlex es una biblioteca de estilos de CSS de código abierto que se utiliza en el desarrollo web para aplicaciones y sitios web receptivos. Esta herramienta ofrece una amplia gama de utilidades de diseño y componentes de interfaz de usuario predefinidos que facilitan la creación de interfaces de usuario modernas y atractivas. (*PrimeFlex*, n.d.)

Una de las principales características de PrimeFlex es su enfoque en la facilidad de uso y la flexibilidad. Los desarrolladores web pueden aprovechar las clases de estilo CSS proporcionadas por PrimeFlex para diseñar páginas web de manera eficiente y coherente. Estas clases abarcan desde sistemas de rejilla y alineación de elementos hasta estilos de tipografía y espaciado entre elementos.

PrimeFlex es altamente compatible con diferentes frameworks y bibliotecas de JavaScript, lo que lo hace versátil y adecuado para una amplia variedad de proyectos web. Se integra fácilmente con frameworks populares como Angular, React y Vue.js, lo que permite a los desarrolladores crear aplicaciones web receptivas y atractivas utilizando tecnologías familiares.

Además de su flexibilidad y compatibilidad, PrimeFlex se destaca por su comunidad activa y su mantenimiento constante. Esto significa que los desarrolladores pueden contar con actualizaciones regulares y un soporte sólido para mantener sus proyectos web actualizados y funcionando sin problemas.

Desarrollo Incremental:

El desarrollo incremental es un enfoque que implica diseñar una implementación inicial, presentarla al usuario para recibir comentarios y luego mejorarla en varias versiones hasta llegar a un sistema completo y adecuado. A diferencia de los modelos de desarrollo tradicionales en cascada, donde las actividades de especificación, desarrollo y validación se realizan de manera separada y lineal, el desarrollo incremental entrelaza estas actividades y permite una retroalimentación constante durante todo el proceso.(Sommerville et al., 2011)

Este enfoque se basa en la idea de que rara vez se puede prever y resolver completamente un problema de software desde el principio. En cambio, se avanza paso a paso hacia una solución, corrigiendo errores a lo largo del camino. El desarrollo incremental resulta más económico y flexible, ya que facilita la adaptación a los cambios en los requisitos del cliente a medida que se desarrolla el software.

Cada iteración o versión del sistema agrega nuevas funcionalidades, comenzando con las más importantes o urgentes. Esto permite que el cliente evalúe el progreso del proyecto en una etapa temprana y, si es necesario, realice ajustes solo en el incremento actual sin afectar el trabajo previo.

OWASP:

La Fundación Open Worldwide Application Security Project (OWASP) es una organización sin fines de lucro que se dedica a mejorar la seguridad del software. Su enfoque abarca diversos aspectos. Por un lado, promueve proyectos de Software de Código Abierto Liderados por la Comunidad, impulsando soluciones y herramientas para fortalecer la seguridad del software. Además, cuenta con una amplia red de Capítulos Locales, que supera los 250 en todo el mundo, fomentando la colaboración y el intercambio de conocimientos sobre seguridad de aplicaciones a nivel local. (*About the OWASP Foundation | OWASP Foundation*, n.d.)

En esta organización, se encuentra una gran Comunidad de Miembros comprometidos con la seguridad de las aplicaciones. Asimismo, OWASP organiza Conferencias Educativas y de Formación de Clase Mundial, que son líderes en la industria de la seguridad. Esta organización tiene como visión la premisa de "No más software inseguro" y su misión es ser la comunidad abierta global que impulsa la creación de software seguro a través de la educación, las herramientas y la colaboración.

OWASP TOP 10:

El OWASP Top 10 es un documento de concienciación estándar y de amplio reconocimiento en el ámbito de la seguridad de aplicaciones web. Este informe representa el consenso de la comunidad de seguridad cibernética y desarrolladores sobre los riesgos de seguridad más críticos que enfrentan las aplicaciones web en la actualidad.(*OWASP Top Ten | OWASP Foundation*, 2023)

Su principal objetivo es brindar una guía esencial para desarrolladores y profesionales de seguridad, ayudándoles a comprender y abordar los riesgos comunes y críticos que pueden afectar a las aplicaciones web. Además, actúa como una referencia globalmente aceptada que promueve las mejores prácticas en el desarrollo de aplicaciones web más seguras.

El OWASP Top 10 es ampliamente reconocido como el primer paso hacia una programación más segura. Las empresas de todo el mundo adoptan este documento como parte fundamental de sus procesos de desarrollo de software. Al hacerlo, inician un proceso para garantizar que sus aplicaciones web minimicen los riesgos de seguridad identificados en este informe.

Utilizar el OWASP Top 10 es una estrategia efectiva para transformar la cultura de desarrollo de software dentro de una organización. Ayuda a crear una mentalidad centrada en la seguridad, donde se valora y prioriza la protección de las aplicaciones web desde las etapas iniciales del ciclo de desarrollo.

De la misma, para el presente proyecto se abordarán como referentes dos derivados del mismo:

- **OWASP API Security:** El Proyecto de Seguridad se enfoca en la protección de las Interfaces de Programación de Aplicaciones (API), que desempeñan un papel crucial en las aplicaciones web, móviles y SaaS modernas. Las API exponen la lógica de la aplicación y datos sensibles, lo que las convierte en objetivos para los atacantes. Este proyecto, reconocido a nivel mundial, se centra en comprender y mitigar las vulnerabilidades únicas de las API. (OWASP API Security Project | OWASP Foundation, 2023)

El Top 10 de Seguridad de las API 2023 es una parte esencial de este proyecto y destaca los riesgos más críticos para las API:

- **Autorización de Nivel de Objeto Roto:** Problemas de control de acceso a nivel de objeto.
- **Autenticación Rota:** Implementación incorrecta de la autenticación.
- **Autorización de Nivel de Propiedad de Objeto Roto:** Falta de validación de autorización a nivel de propiedad del objeto.
- **Consumo No Restringido de Recursos:** Uso excesivo de recursos, lo que puede llevar a la denegación de servicio.
- **Autorización de Nivel de Función Rota:** Políticas de control de acceso complejas que conducen a fallas de autorización.
- **Acceso No Restringido a Flujos de Negocios Sensibles:** Exposición de flujos de negocios sin restricciones.
- **Falsificación de Solicitudes del Lado del Servidor (SSRF):** Explotación de fallas SSRF al obtener recursos remotos sin validación.
- **Configuración de Seguridad Incorrecta:** Configuraciones complejas que pueden contener fallas de seguridad.
- **Gestión Incorrecta de Inventario:** Exposición de múltiples puntos finales sin documentación adecuada.
- **Consumo Inseguro de APIs:** Confiar demasiado en datos de terceros y adoptar estándares de seguridad débiles.

Este proyecto es esencial para la comunidad de desarrollo, ya que ayuda a comprender y abordar las amenazas a la seguridad de las API, promoviendo un desarrollo más seguro y una cultura de código más seguro en las organizaciones. La licencia Creative Commons Attribution-ShareAlike 4.0 permite su uso gratuito y la adaptación con atribución adecuada.

- **OWASP Mobile Top 10 2023:** Se enfoca en identificar y abordar las principales amenazas de seguridad en aplicaciones móviles. Estas amenazas son críticas debido a la creciente dependencia de las aplicaciones móviles en la vida diaria de las personas y su acceso a información sensible. Este proyecto proporciona una guía valiosa para desarrolladores, profesionales de seguridad y propietarios de aplicaciones móviles para comprender y mitigar estas amenazas. (OWASP Mobile Top 10 | OWASP Foundation, 2023)

Manual para medios y/o plataformas digitales:

Este manual proporcionado por la Universidad Católica de Santa María (UCSM), se presenta como un conjunto de directrices y pautas que especifican el uso adecuado de elementos visuales, como imágenes, iconos y paletas de colores, en el contexto de proyectos digitales y medios en línea. Este manual tiene como objetivo principal proporcionar una guía precisa y detallada sobre cómo utilizar estos componentes visuales de manera coherente y representativa de la identidad y valores de la institución educativa. Además, busca asegurar que la comunicación visual de la UCSM en entornos digitales sea consistente, profesional y alineada con sus estándares de calidad, promoviendo así una experiencia uniforme y reconocible para los usuarios y audiencias en línea.

Kebab-case:

Es una convención de nomenclatura en la que las palabras se separan utilizando un guion (-) como característica distintiva. En esta convención, todas las letras suelen ser minúsculas, y se utiliza un guion para separar palabras en lugar de espacios, mayúsculas u otros caracteres. Esta convención es una de las sugerencias a emplear (con camelCase) en la documentación oficial de Angular con el fin de que el código sea más coherente y fácil de mantener cuando trabajas con el framework. (*Angular - Angular Coding Style Guide*, n.d.)

Lazy Loading:

La carga diferida o carga perezosa en español, se refiere a una técnica que consiste en cargar módulos o componentes de una aplicación de manera asincrónica y bajo demanda, en lugar de cargar todo el código de la aplicación al inicio. Esto significa que los módulos o componentes se cargan solo cuando el usuario navega a una parte específica de la aplicación que los necesita, lo que puede mejorar significativamente el rendimiento y la velocidad de carga de la aplicación. (*Angular - Lazy-Loading Feature Modules*, n.d.)

Standalone Strategy:

Las últimas versiones de Angular, cuenta con la inclusión de los llamados componentes “Standalone” los cuales son unidades autónomas de la arquitectura de Angular que encapsulan la lógica de presentación y la funcionalidad asociada en una única entidad reutilizable. Estos componentes están diseñados para ser independientes y modularizados, lo que facilita su inclusión en diferentes partes de una aplicación sin generar dependencias no deseadas. (*Angular - Getting Started with Standalone Components*, n.d.)

La estrategia que se busca incorporar es de componentes “limpios” y reutilizables empleando tanto standalone, como estilo y plantilla unificado, creando un único archivo con el componente en cuestión que pueda ser instanciado las veces necesarias para los diversos contextos que la requieran.

Patrones de Diseño:

Un patrón de diseño es una solución general y reutilizable para un problema común que surge durante el diseño y desarrollo de software. Estos patrones representan las mejores prácticas codificadas y las experiencias acumuladas por los desarrolladores a lo largo del tiempo. Proporcionan un enfoque estructurado para resolver problemas recurrentes de manera eficiente y efectiva.

Los patrones de diseño no son código concreto o bibliotecas, sino más bien descripciones de alto nivel que abordan cómo organizar y estructurar el código para resolver un problema específico de manera elegante y mantenible. Son como planos que guían la construcción de una solución informática. (*¿Qué Es Un Patrón de Diseño?*, n.d.)

Node Package Manager (npm):

Node Package Manager (npm) es una herramienta fundamental en el ecosistema de desarrollo de Node.js que facilita la gestión de dependencias y la distribución de paquetes de software. Como gestor de paquetes predeterminado para Node.js, npm permite a los desarrolladores instalar, compartir y gestionar fácilmente las bibliotecas y herramientas que necesitan para sus proyectos.

La funcionalidad principal de npm radica en su capacidad para descargar e instalar paquetes de software desde el repositorio npm, que es una colección global de módulos y librerías de código abierto. Los desarrolladores pueden acceder a este vasto repositorio para encontrar y utilizar una amplia gama de paquetes que van desde frameworks y bibliotecas populares hasta utilidades y herramientas de desarrollo especializadas. (*About Npm | Npm Docs*, n.d.)

PNPM:

pnpm es un gestor de paquetes de código abierto para Node.js, diseñado para optimizar la instalación y gestión de dependencias de proyectos. A diferencia de otros gestores de paquetes como npm y Yarn, pnpm utiliza un enfoque de almacenamiento en caché y enlace simbólico para reducir el espacio de almacenamiento y mejorar el rendimiento durante el proceso de instalación. Esto permite una instalación más rápida de dependencias y una menor huella de almacenamiento en disco, lo que es especialmente beneficioso en el contexto de una aplicación móvil, donde la optimización del rendimiento y la eficiencia de los recursos son críticos para una experiencia de usuario fluida. (*Fast, Disk Space Efficient Package Manager* | Pnpm, n.d.)

Git:

Git es un sistema de control de versiones distribuido diseñado para manejar todo, desde pequeños hasta muy grandes proyectos con velocidad y eficiencia. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux y se ha convertido en la herramienta estándar para el control de versiones en la industria del software. El mismo permite, a cada desarrollador, tener una copia completa del historial del proyecto. Esto significa que se pueden realizar operaciones de control de versiones localmente, sin necesidad de estar conectados a un servidor central. Cada clon del repositorio es un repositorio completo con historial completo y control total de las versiones. Git es un proyecto de código abierto, lo que significa que su código fuente está disponible para que cualquiera lo use, modifique y distribuya. (*Git*, n.d.)

GitHub:

GitHub es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Fundada en 2008, GitHub permite a los desarrolladores almacenar y gestionar su código, así como seguir y controlar los cambios realizados en él. La plataforma se ha convertido en un estándar de facto para el desarrollo de software gracias a su capacidad para facilitar la colaboración y su integración con una amplia variedad de herramientas y servicios. Aunque GitHub facilita el desarrollo de software de código abierto y alberga millones de proyectos de código abierto, la plataforma en sí no es de código abierto. Sin embargo, GitHub proporciona acceso gratuito a sus principales características para proyectos públicos, lo que lo convierte en un recurso valioso para la comunidad de software libre. (*GitHub*, n.d.)

Gitflow:

Gitflow es un modelo de flujo de trabajo para el control de versiones con Git que se centra en la gestión estructurada y escalable del desarrollo de software en proyectos colaborativos. Este modelo, ideado por Vincent Driessen, proporciona una serie de directrices y convenciones para organizar y coordinar el trabajo de equipos de desarrollo, especialmente en proyectos de código abierto y empresas con múltiples desarrolladores. (*Flujo de Trabajo de Gitflow* | Atlassian Git Tutorial, n.d.)

Pipes:

En Angular, los pipes son una herramienta que permite transformar la visualización de datos en las plantillas de manera eficiente y declarativa. Los pipes actúan como funciones que se aplican a los datos directamente en las vistas, facilitando su presentación en el formato deseado sin necesidad de manipular los datos originales en el componente. Este se emplea para dar un formato a los datos de uso común como fechas, números, textos y monedas (*Pipes • Overview • Angular*, n.d.).

ISO 25010:

La ISO 25010 es un estándar internacional que define un modelo de calidad para evaluar la calidad del software. Este modelo se centra en varias características clave, como la funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Cada una de estas características se desglosa en sub-características que ayudan a medir y mejorar la calidad del software, proporcionando un marco

estructurado para la evaluación y asegurando que el software cumpla con los requisitos y expectativas del usuario final.(iso25000, n.d.)

Android Studio:

Android Studio es un entorno de desarrollo integrado (IDE) oficial para la plataforma Android, desarrollado por Google. Proporciona herramientas completas para el desarrollo de aplicaciones móviles, incluyendo un editor de código, un emulador para pruebas, herramientas de depuración y un sistema de gestión de proyectos basado en Gradle. Android Studio permite a los desarrolladores diseñar, compilar y depurar aplicaciones de manera eficiente, facilitando el proceso de desarrollo gracias a su integración con las herramientas de Android y sus características avanzadas de edición y depuración (*Download Android Studio & App Tools - Android Developers*, n.d.).

PG Admin:

pgAdmin 4 es una aplicación avanzada de administración y desarrollo para bases de datos PostgreSQL, diseñada para proporcionar una interfaz gráfica completa y fácil de usar. Esta herramienta permite a los desarrolladores y administradores realizar tareas complejas de gestión de bases de datos de manera eficiente, incluyendo la creación, modificación y eliminación de objetos de bases de datos, la ejecución de consultas SQL, y la visualización y manipulación de datos.

Entre sus características principales, se destaca por proporcionar una interfaz intuitiva que facilita la navegación a través de bases de datos y esquemas, soportando la administración de roles y permisos para un control detallado del acceso a los datos. Además, proporciona un editor de SQL robusto con funcionalidades avanzadas como resaltado de sintaxis y autocompletado, lo que mejora significativamente la experiencia del usuario en la ejecución y análisis de consultas. También incluye herramientas integradas para realizar copias de seguridad y restauración, lo que garantiza la integridad y disponibilidad de los datos (*Dpage/Pgadmin4 - Docker Image | Docker Hub*, n.d.).

Dependabot:

Dependabot es una herramienta integrada en GitHub que ayuda a los desarrolladores a mantener sus dependencias actualizadas y seguras. Funciona monitoreando los archivos de configuración de dependencias en un repositorio y generando automáticamente solicitudes de extracción cuando detecta actualizaciones disponibles o vulnerabilidades de seguridad (*Dependabot*, n.d.).

Entre sus funcionalidades principales, Dependabot ofrece la actualización automática de dependencias, revisando periódicamente las mismas y sugiriendo actualizaciones cuando hay nuevas versiones disponibles. Esto facilita la gestión continua de las dependencias del proyecto. Además, proporciona alertas de seguridad, escaneando las dependencias en busca de vulnerabilidades conocidas y generando alertas para que los desarrolladores puedan abordar estos problemas rápidamente, minimizando así los riesgos de seguridad.

Postman:

Postman es una herramienta ampliamente utilizada por desarrolladores y profesionales de TI para el desarrollo, prueba, documentación y monitoreo de APIs. Su principal función es facilitar la interacción con APIs RESTful y otros servicios web, permitiendo a los usuarios enviar solicitudes HTTP de manera sencilla y obtener respuestas (*Postman Documentation Overview | Postman Learning Center*, n.d.).

Capítulo 5: “Solución Propuesta”

Introducción:

En el presente apartado se destacará los diversos aspectos considerados a la hora de llevar a cabo el desarrollo del proyecto en cuestión; contemplando las actividades y metodología elegida, las herramientas empleadas, el desarrollo de cada una de las etapas como así también análisis de costos y factibilidad para evaluar el impacto que tenga la ejecución de este proyecto en la institución.

Análisis de requerimientos:

Para el corriente apartado, se buscó plantear una primera instancia detallar los requerimientos del proyecto tanto Funcionales como No Funcionales. Los mismos se detallarán a continuación:

Requerimientos Funcionales:

- **Inicio de Sesión:** La aplicación debe permitir a los estudiantes iniciar sesión utilizando sus credenciales institucionales (nombre de usuario y contraseña).
- **Cierre de Sesión:** La aplicación debe proporcionar una opción para que los estudiantes puedan cerrar sesión de manera segura.
- **Consulta de Carreras Matriculadas:** La aplicación debe permitir a los estudiantes ver una lista de todas las carreras en las que están matriculados.
- **Consulta de Cursos Actuales y Pasados:** La aplicación debe permitir a los estudiantes acceder a una lista de todos los cursos que están cursando en el semestre actual, así como los cursos cursados en semestres anteriores.
- **Consulta de Notas:** La aplicación debe permitir a los estudiantes ver sus notas y evaluaciones de todos los cursos en los que están o han estado inscritos.
- **Historial Académico:** La aplicación debe proporcionar un historial académico completo, mostrando las notas obtenidas en todos los cursos cursados.
- **Consulta de Comisiones:** La aplicación debe permitir a los estudiantes ver a qué comisión están inscritos para cada curso.
- **Información de Aulas y Horarios:** La aplicación debe proporcionar detalles sobre las aulas asignadas y los horarios de las clases para cada curso.
- **Consulta de Deuda:** La aplicación debe permitir a los estudiantes ver su estado de cuenta y cualquier deuda pendiente con la institución educativa.
- **Acceso a la Biblioteca Virtual:** La aplicación debe proporcionar enlaces directos a la biblioteca virtual de la universidad.
- **Acceso a la Plataforma de Aulas Virtuales:** La aplicación debe ofrecer acceso a la plataforma de aulas virtuales.
- **Acceso al Sistema de Matrículas:** La aplicación debe permitir a los estudiantes acceder al sistema de matrículas para realizar inscripciones y consultas relacionadas.
- **Consulta de Eventos:** La aplicación debe permitir a los estudiantes ver una lista de eventos relevantes que ocurran en la universidad, incluyendo conferencias, talleres, y actividades extracurriculares.

Requerimientos No Funcionales:

- **Compatibilidad Multiplataforma:** La aplicación resultante debe ser compatible con una variedad de dispositivos y sistemas operativos móviles, incluyendo iOS y Android. Esta restricción implica un desafío adicional en términos de desarrollo y pruebas, ya que se deben garantizar experiencias de usuario consistentes en todas las plataformas.
- **Seguridad de la API:** La API debe ser diseñada y desarrollada con un enfoque riguroso en la seguridad. Debe incorporar medidas de autenticación y autorización sólidas para garantizar que solo los usuarios autorizados tengan acceso a los datos y funciones pertinentes. Además, se debe implementar un control de acceso basado en roles para gestionar los permisos de los usuarios de manera efectiva.
- **Escalabilidad:** La API debe ser escalable para manejar un crecimiento potencial en la cantidad de usuarios y el tráfico de datos. Esto implica la implementación de estrategias de escalabilidad, como la asignación eficiente de recursos y la administración de la carga, para garantizar un rendimiento óptimo incluso en condiciones de alta demanda.
- **Compatibilidad Futura:** Dado que la tecnología evoluciona con el tiempo, la API debe ser diseñada de manera que permita la incorporación de nuevas características y tecnologías en el futuro sin interrupciones significativas en el servicio. Debe ser flexible y adaptable para cumplir con los requisitos cambiantes de la universidad y las necesidades de los usuarios.
- **Documentación Exhaustiva:** Es esencial proporcionar documentación detallada y exhaustiva de la API, incluyendo descripciones de los endpoints, métodos de autenticación, ejemplos de uso y posibles errores. Una documentación clara facilitará la integración de la API con otras aplicaciones y servicios.

A partir de dichos requerimientos, se procede a detallar las historias de usuario que guiarán el desarrollo del proyecto.

Por un lado, las historias de usuario del proyecto de la aplicación móvil:

<i>Prioridad</i>	<i>Como</i>	<i>Quiero</i>	<i>Para</i>	<i>Criterio de Aceptación</i>	<i>Incremento</i>	<i>Nº</i>
1	Estudiante	Acceder y gestionar información personal a través de la app móvil	Tener control sobre mi información	• Ingresar con número de identificación y contraseña validos • Almacenamiento seguro y acceso en cualquier momento • Visualizar información personal del alumno actualizada.	1	1

Tabla 1: historia de usuario: "Inicio de Sesión"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
1	Estudiante	Buscar y seleccionar cursos eficientemente desde la app móvil	Facilitar la selección de cursos	- Búsqueda por nombre o código de curso - Categorización de cursos por semestre - Acceso a información detallada sobre cada curso	2	2

Tabla 2: historia de usuario: "Acceso a cursos"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
1	Estudiante	Acceder y visualizar horarios de clases de forma clara en la app móvil	Mantenerme organizado con mis horarios	- Acceso integral a horarios académicos - Interfaz de usuario intuitiva - Sincronización con cursos seleccionados	3	3

Tabla 3: historia de usuario: "Horario"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
1	Estudiante	Consultar el estado de Deuda como los pagos correspondientes.	Tener un seguimiento del estado actual de deuda de la matrícula.	- Visualizar Saldo a Pagar. - Listado de Cuotas pagadas del presente año	3	4

Tabla 4: historia de usuario: "Deuda"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
2	Estudiante	Consultar calificaciones, registros y progreso académico en la app móvil	Tener un seguimiento de mi desempeño académico	• Consulta de calificaciones de exámenes, trabajos, etc. • Acceso a registros académicos actualizados • Visualización de progreso en programa de estudios	4	5

Tabla 5: historia de usuario: "Avance Académico"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
2	Estudiante	Recibir notificaciones y acceder a información sobre eventos universitarios	Mantenerme informado sobre eventos relevantes	• Notificaciones personalizadas sobre eventos • Detalles claros sobre fecha, hora, lugar y descripción de eventos • Acceso a una variedad de eventos universitarios	5	6

Tabla 6: historia de usuario: "Eventos"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
3	Estudiante	Acceder a recursos educativos y herramientas de aprendizaje en la app móvil	Mejorar mi experiencia de aprendizaje	• Acceso a bibliotecas virtuales con amplia colección • Materiales de estudio actualizados • Participación en aulas virtuales • Interfaz fácil de usar y recursos multimedia disponibles	6	7

Tabla 7: historia de usuario: "Recursos Auxiliares"

Finalmente, las historias de usuario del proyecto del back-end:

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
1	Estudiante	Iniciar sesión en el sistema	Acceder a mis datos académicos y financieros de manera segura	- El endpoint api/login acepta una petición POST con el documento de identificación y la contraseña. - Se genera y devuelve un token de sesión válido al usuario. - Se recibe un mensaje de error si las credenciales son incorrectas. - El token expire pasada las 24 hrs.	1	8

Tabla 8 : historia de usuario: "endpoint auth"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
1	Estudiante	Consultar mi deuda académica	Saber cuánto debo y planificar mis pagos	- El endpoint api/debts/get-debt devuelve la deuda del curso especificado y sus cuotas asociadas. - Se requiere un token de sesión válido en el header de la petición. - La respuesta contiene detalles de la deuda y las cuotas.	3	9

Tabla 9 historia de usuario: "endpoint debt"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
2	Estudiante	Consultar los cursos en los que estoy inscrito	Verificar mi carga académica y planificar mi semestre	- El endpoint api/enrollments devuelve una lista de todos los cursos en los que el estudiante está inscrito. - Se requiere un token de sesión válido en el header de la petición. - La respuesta incluye detalles de cada curso inscrito.	2	10

Tabla 10 historia de usuario: "endpoint enrollment"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
2	Estudiante	Obtener información detallada de un curso específico	Conocer más sobre un curso particular que estoy tomando	- El endpoint api/enrollments /id devuelve información detallada del curso especificado. - Se requiere un token de sesión válido en el header de la petición. - La respuesta contiene detalles completos del curso.	2	11

Tabla 11 historia de usuario: "endpoint enrollment/id"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
3	Estudiante	Ver una lista de eventos universitarios	Estar al tanto de las actividades y eventos de la universidad	• El endpoint api/events devuelve una lista completa de eventos. • Se requiere un token de sesión válido en el header de la petición. • La respuesta contiene detalles de cada evento.	5	12

Tabla 12 historia de usuario: "endpoint events"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
3	Estudiante	Obtener información detallada de un evento específico	Decidir si quiero asistir a ese evento	• El endpoint api/events/i devuelve información detallada del evento especificado. • Se requiere un token de sesión válido en el header de la petición. • La respuesta incluye detalles completos del evento.	5	13

Tabla 13 historia de usuario: "endpoint events/id"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
3	Estudiante	Ver una lista de todos los semestres disponibles	Conocer la estructura temporal de mi plan de estudios	- El endpoint api/semesters devuelve una lista de todos los semestres. - Se requiere un token de sesión válido en el header de la petición. - La respuesta contiene detalles de cada semestre.	4	14

Tabla 14 historia de usuario: "endpoint semesters"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
3	Estudiante	Obtener información detallada de un semestre específico	Verificar las asignaturas y actividades de ese semestre	- El endpoint api/semesters/<string:id> devuelve información detallada del semestre especificado. - Se requiere un token de sesión válido en el header de la petición. - La respuesta incluye detalles completos del semestre.	4	15

Tabla 15 historia de usuario: "endpoint semesters/id"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	de Incremento	N°
1	Estudiante	Ver mi perfil completo	Revisar mis datos personales y académicos	- El endpoint <code>api/students/id</code> devuelve el perfil completo del estudiante. - Se requiere un token de sesión válido en el header de la petición. - La respuesta incluye todos los datos personales y académicos del estudiante.	1	16

Tabla 16 historia de usuario: "endpoint subjects"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	N°
2	Estudiante	Ver las materias en las que estoy inscrito	Conocer mis asignaturas actuales	- El endpoint <code>api/subject-registrations/?enrollment_id=<uuid:id></code> devuelve una lista de materias inscritas. - Se requiere un token de sesión válido en el header de la petición. - La respuesta contiene detalles de cada materia inscrita.	4	17

Tabla 17 historia de usuario: "endpoint subjects per enrollment"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	de Incremento	Nº
2	Estudiante	Obtener información detallada de una materia específica	Conocer más sobre esa materia en particular	El endpoint <code>api/subject-registrations/<string:id></code> devuelve información detallada de la materia especificada. Se requiere un token de sesión válido en el header de la petición. La respuesta incluye detalles completos de la materia.	4	18

Tabla 18 historia de usuario: "endpoint subject/id"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
3	Estudiante	Ver las materias restantes por cursar	Planificar mi carga académica futura	- El endpoint <code>api/subject-registrations/get-remaining-subjects/?enrollment_id=<uuid:id></code> devuelve una lista de materias restantes. - Se requiere un token de sesión válido en el header de la petición. - La respuesta contiene detalles de cada materia restante.	4	19

Tabla 19 historia de usuario: "endpoint subjects remaining"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
3	Estudiante	Ver estadísticas de mis materias aprobadas y restantes	Tener una visión general de mi progreso académico	El endpoint api/subject-registrations/get-statistics/?enrollment_id=<uuid:id> devuelve estadísticas de materias aprobadas y restantes. Se requiere un token de sesión válido en el header de la petición. La respuesta incluye detalles y estadísticas precisas de las materias.	6	20

Tabla 20 historia de usuario: "endpoint get statistics"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
1	Desarrollador	Crear un archivo Docker Compose para el back-end	Facilitar la configuración y el despliegue del servicio en diferentes entornos	El archivo docker-compose.yml incluye la configuración para la API, una base de datos PostgreSQL y pgAdmin. La configuración permite que todos los servicios se inicien y se comuniquen correctamente.	1	21

Tabla 21 historia de usuario: "configuración docker"

Prioridad	Como	Quiero	Para	Criterio de Aceptación	Incremento	Nº
2	Desarrollador	Desplegar la aplicación en Render para pruebas	Validar las funcionalidades en un entorno de producción	La aplicación se despliega correctamente en Render. Se puede acceder a todos los endpoints y realizar pruebas funcionales de la API en el entorno de Render.	1	22

Tabla 22 historia de usuario: "despliegue en render"

Metodología a Implementar:

En el marco de este proyecto, se ha reconocido la necesidad de crear una aplicación de gestión académica destinada a estudiantes, con el propósito de ofrecerles un acceso eficaz y sencillo a información crítica relacionada con su desarrollo académico y su experiencia en la institución educativa. Para abordar esta necesidad, se ha implementado un enfoque de desarrollo conocido como "metodología incremental".

Esta metodología ha aportado considerables ventajas al desarrollo del proyecto, ya que este, debido a su naturaleza compleja y su potencial expansión a lo largo del tiempo, requiere modificaciones y ampliaciones significativas. Estas modificaciones siempre se han abordado desde una perspectiva funcional, es decir, cada módulo adicional debe funcionar de manera óptima y ser plenamente compatible con los existentes.

La metodología incremental ha permitido que el proyecto evolucione de manera efectiva y eficiente. En lugar de intentar crear una aplicación completa de una sola vez, se ha dividido en módulos o incrementos, cada uno enfocado en una funcionalidad específica. Esto ha posibilitado que los usuarios puedan acceder a funciones valiosas desde el principio y que se beneficien de ellas, mientras que el equipo de desarrollo sigue trabajando en incrementos adicionales.

El desarrollo se dividió en múltiples fases, donde cada fase corresponde a un incremento del sistema, en el cual se inició con una identificación de requisitos; de cada requisito, de ser necesario se descomponían en subtarefas y se realizaba su correspondiente desarrollo como nueva funcionalidad creando una rama Feature siguiendo la metodología Gitflow. Una vez finalizado el desarrollo se solicitaba como Pull Request por la plataforma GitHub, en pos de que se haga sus respectivas pruebas en función a sus criterios de aceptación y se integrara finalmente al proyecto una vez las haya pasado. Así sucesivamente para cada incremento. Una vez finalizado el incremento, se realizaba una validación con el analista en pos de ver si efectivamente se arribó a lo deseado acorde a lo esperado de ese incremento. Una vez aprobado, se integraba con el sistema y se repetía la validación en su totalidad ahora con miembros del equipo de desarrollo la UCSM para obtener un feedback.

De cada incremento se tiene como artefactos tanto la documentación de los requisitos como una actualización en la documentación provista por la API para su consumo apropiado de la aplicación móvil. Los stakeholders incluyen a los desarrolladores, al grupo de muestra para la evaluación de usabilidad llevada a cabo posterior al último incremento y los usuarios finales siendo estos todos los usuarios de la Universidad Católica de Santa María.

En síntesis, para cada incremento las actividades principales incluyen la recopilación de requisitos, el diseño del sistema, la codificación, las pruebas y la implementación.

Los incrementos del presente proyecto son:

Incremento 1:

Este primer incremento del proyecto se enfoca en brindar a los estudiantes la capacidad de acceder y gestionar su información personal a través de la aplicación móvil. La información personal abarcará una serie de datos cruciales que son relevantes tanto para el estudiante como para la institución educativa. Estos datos, entre otros, incluirán:

1. **Nombre Completo:** El nombre completo del estudiante, lo que permitirá una identificación precisa en los registros y comunicaciones con la universidad.
2. **Dirección:** La dirección o residencia actual del estudiante, lo que permitirá a la institución tener una referencia actualizada de la ubicación del estudiante.

3. **Número de Teléfono:** El número de teléfono de contacto del estudiante, un elemento esencial para la comunicación efectiva entre el estudiante y la universidad, así como para cualquier comunicación urgente.
4. **Número de Identificación:** Este puede ser el número de identificación o de identidad del estudiante o cualquier otro número único que la universidad utilice para identificar a sus estudiantes en sus sistemas internos (ej. pasaporte).

Incremento 2:

El segundo incremento del proyecto se enfoca en brindar a los estudiantes una funcionalidad de búsqueda y selección de cursos de manera eficiente y organizada. Este módulo de la aplicación permitirá a los estudiantes acceder a una amplia gama de cursos y programas académicos ofrecidos por la Universidad Católica de Santa María (UCSM). A continuación, se detallan las principales características y ventajas de este incremento:

1. **Búsqueda de Cursos:** Los estudiantes tendrán la capacidad de buscar cursos en función del nombre del curso o el código del estudiante.
2. **Selección de Cursos:** Una vez que los estudiantes encuentren un curso de su interés, podrán seleccionarlo y acceder a las demás funcionalidades en relación a dicho curso. La selección de cursos se realiza de manera sencilla a través de la interfaz de la aplicación.
3. **Acceso a Información Detallada:** Al seleccionar un curso, los estudiantes podrán acceder a información detallada, como el plan de estudios, los horarios de clases, los recursos del curso y la descripción del profesor. Esto garantiza que los estudiantes tengan acceso a todos los detalles necesarios para tomar decisiones informadas sobre sus estudios.

Este incremento se desarrollará y se integrará de manera coherente con los demás módulos de la aplicación. La búsqueda y selección de cursos serán una parte integral de la experiencia del estudiante, mejorando la eficiencia y la capacidad de organización de su trayectoria académica en la UCSM.

Incremento 3:

En el tercer incremento del proyecto, se busca proporcionar a los estudiantes una función esencial: la capacidad de acceder y visualizar sus horarios de clases de manera conveniente a través de la aplicación móvil como también el acceso a las deudas de sus respectivas matrículas. Este módulo ofrece una serie de beneficios clave para mejorar la experiencia académica de los estudiantes:

1. **Acceso Integral:** Los estudiantes podrán acceder a sus horarios académicos de forma integral. Los horarios mostrarán información detallada, como el nombre del profesor, la ubicación del aula y el identificador de la sección teórica y el grupo práctico correspondientes a cada clase. Esto proporcionará a los estudiantes una visión completa de sus compromisos académicos.
2. **Facilidad de Uso:** La interfaz de usuario ha sido diseñada pensando en la facilidad de uso. Los horarios se presentarán de manera clara y organizada, lo que facilitará la navegación y la identificación de la información necesaria.
3. **Recuperación del Saldo de Deuda:** Este módulo permitirá a los estudiantes consultar el saldo de deuda de sus matrículas. Los estudiantes podrán acceder a un desglose detallado de sus deudas pendientes, incluyendo el monto total adeudado y las boletas que ya han sido pagadas. Esto facilitará la gestión financiera y la planificación de pagos futuros.

Al brindar a los estudiantes un acceso rápido y sencillo a sus horarios de clases, este incremento contribuirá significativamente a su organización y gestión del tiempo, lo que a su vez puede mejorar su rendimiento académico y su satisfacción en la UCSM.

Incremento 4:

En el cuarto incremento del proyecto, se implementará una función fundamental que permitirá a los estudiantes llevar un seguimiento detallado de su rendimiento académico. Esto incluye la consulta de calificaciones, registros académicos y el progreso en su programa de estudios. Este módulo tiene como objetivo proporcionar a los estudiantes una visión completa de su desempeño académico y empoderarlos para tomar decisiones informadas. Aquí se detallan las características clave:

1. **Consulta de Calificaciones:** Los estudiantes podrán consultar sus calificaciones de forma rápida y sencilla a través de la aplicación. Esto incluye calificaciones de exámenes, trabajos y otros elementos de evaluación. La información se presentará de manera clara y organizada.
2. **Registros Académicos:** Los estudiantes podrán acceder a sus registros académicos, lo que les permitirá revisar su historial académico. Podrán ver los cursos que han completado con éxito, los que están cursando actualmente y los que están pendientes.
3. **Progreso del Programa de Estudios:** La aplicación mostrará el progreso del estudiante en su programa de estudios. Esto incluirá un seguimiento de los cursos obligatorios y optativos que deben completar para obtener su título.
4. **Información Actualizada:** Toda la información proporcionada será actualizada en tiempo real, lo que garantiza que los estudiantes tengan acceso a datos precisos y actualizados.
5. **Apoyo a la Toma de Decisiones:** Al tener acceso a su rendimiento académico (avance de carrera y materias aprobadas) de manera constante, los estudiantes podrán tomar decisiones informadas sobre su educación y buscar áreas de mejora si es necesario.

Incremento 5:

En este quinto incremento, nuestra aplicación se centrará en mejorar la experiencia de los estudiantes al mantenerlos informados sobre los eventos que tendrán lugar en la Universidad Católica de Santa María (UCSM). Estos eventos pueden ser de diversa índole, desde conferencias académicas hasta competencias deportivas y actividades culturales. Aquí se describen en detalle las características clave de este incremento:

1. **Amplia Variedad de Eventos:** Los estudiantes tendrán acceso a una amplia variedad de eventos universitarios, lo que incluye actividades académicas, culturales, deportivas y sociales. Esto les permitirá estar al tanto de todas las oportunidades que ofrece la universidad.
2. **Notificaciones Personalizadas:** Los estudiantes podrán personalizar sus preferencias de notificación, lo que les permitirá recibir alertas solo para los tipos de eventos que les interesen. Esto garantiza que las notificaciones sean relevantes y no abrumadoras.
3. **Información Detallada:** Cada evento contendrá información detallada, como la fecha, la hora, el lugar y una descripción. Los estudiantes podrán acceder a todos los detalles que necesiten para decidir su participación.
4. **Recordatorios de Eventos:** La aplicación enviará recordatorios oportunos para asegurarse de que los estudiantes no se pierdan eventos importantes.
5. **Conexión Social:** Esta función fomenta la interacción social al permitir que los estudiantes participen en actividades extracurriculares y conozcan a otros miembros de la comunidad universitaria.

Incremento 6:

En el sexto incremento, nos enfocamos en brindar a los estudiantes un acceso integral a una amplia gama de recursos educativos y herramientas de aprendizaje. La premisa principal de esta funcionalidad es empoderar a los estudiantes al proporcionarles los medios para mejorar sus experiencias de aprendizaje. Aquí se describen las características clave de este incremento:

1. Bibliotecas Virtuales: Los estudiantes podrán acceder a bibliotecas virtuales que contienen una extensa colección de recursos, que incluyen libros electrónicos, revistas, tesis y otros documentos académicos. Esta característica les permitirá profundizar en sus estudios y llevar a cabo investigaciones de manera efectiva.
2. Materiales de Estudio: La aplicación ofrecerá acceso a materiales de estudio actualizados relacionados con los cursos en los que están inscritos los estudiantes. Esto puede incluir presentaciones, documentos de lectura y recursos multimedia.
3. Aula Virtual: Los estudiantes podrán acceder a aulas virtuales donde se llevan a cabo actividades de aprendizaje en línea. Esto incluye la posibilidad de participar en discusiones, entregar tareas y acceder a contenido adicional relacionado con los cursos.
5. Interfaz Amigable: La aplicación proporcionará una interfaz fácil de usar que permitirá a los estudiantes navegar y buscar recursos de manera eficiente.

Estructura del proyecto:

En el presente apartado, se detallará como se ha estructurado el proyecto de desarrollo, el cual fue particionado en una arquitectura de “front-end” y “back-end”. La primera parte está orientada explícitamente en todo lo relacionado a interfaz de usuario como experiencia de usuario (UX/UI) abarcando tanto la configuración de las tecnologías empleadas como son Ionic, Capacitor, Angular y las dependencias empleadas para su desarrollo, como así también la evaluación en base a los riesgos marcados por OWASP TOP 10 Mobile.

Por otra parte, del lado del back-end, se detalla todo lo relacionado a la lógica del servidor y la gestión de los datos, utilizando el framework Django Rest Framework (DRF) y la base de datos PostgreSQL. Esto también incluye tanto la configuración de estos, la estructuración de la base de datos partiendo de un diagrama entidad-relación (DER), la estructuración y orquestación de ambos contenedores en Docker usando Docker compose a modo de una mejor portabilidad y gestión; como, por último, al igual que el front-end, evaluar la respuesta a los riesgos marcados en el OWASP Api Security TOP 10.

Los códigos fuentes tanto del front-end como del back-end se han gestionado a través de la plataforma GitHub, utilizando Git como herramienta de control de versiones y siguiendo el flujo de trabajo Gitflow. Cada subsección posterior abordará en detalle cómo se llevó a cabo el desarrollo de cada componente, incluyendo las consideraciones tomadas para garantizar la calidad, seguridad y mantenibilidad del software desarrollado.

Front-end:

El conjunto de herramientas de software empleadas para el desarrollo de la interfaz de la aplicación para la autogestión estudiantil en la Universidad Católica de Santa María fueron el framework de Typescript Angular en la versión 17 en conjunto con Ionic 7, como también el framework CSS PrimeFlex de Primefaces para el diseño de una interfaz fluida, minimalista y, sobre todo, sencilla para el usuario. En el presente subapartado se irán describiendo y detallando con profundidad la elección de cada una de las tecnologías como su configuración. Se puede observar en la siguiente ilustración la imagen del repositorio publicado en GitHub:

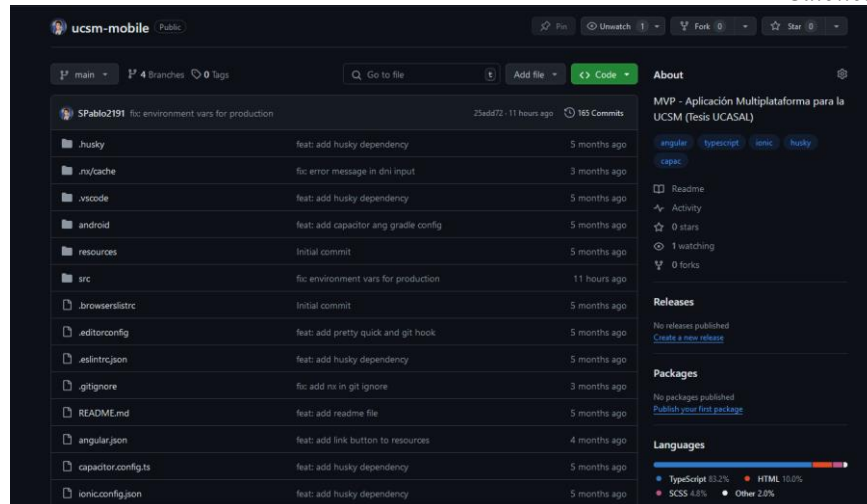


Ilustración 2 "Repositorio de GitHub de la aplicación Móvil"

Uso de Ionic, Capacitor y Angular:

El desarrollo de aplicaciones móviles en la actualidad enfrenta desafíos significativos, especialmente en lo que respecta a la creación de experiencias de usuario cohesivas, rendimiento óptimo y mantenibilidad del código. En este contexto, la elección de las tecnologías adecuadas para el desarrollo de la aplicación móvil para la autogestión estudiantil en la Universidad Católica de Santa María (UCSM) de Arequipa fue una consideración estratégica crucial. Si bien existen diversas opciones liderando en el mercado de framework de desarrollo multiplataforma documentado en la plataforma *statista* (*Cross-Platform Mobile Frameworks Used by Global Developers 2022* | Statista, n.d.) como se puede llegar visualizar en la ilustración 2:

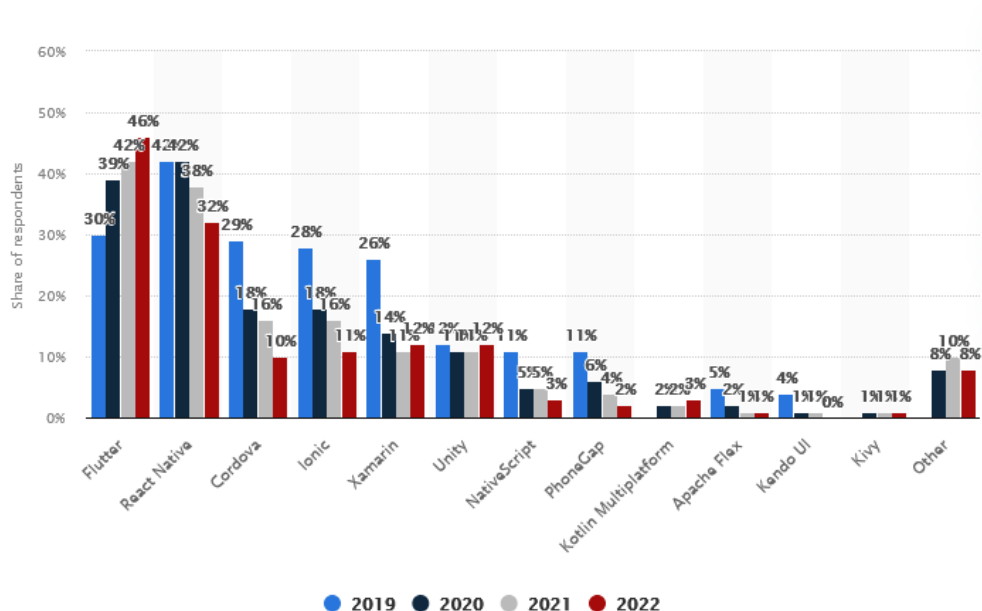


Ilustración 3 "Estadística de los mobile frameworks Cross-Platform usados por desarrolladores de software en el periodo 2019 a 2022"

Se optó por uno de los 5 frameworks más utilizados el cual fue Ionic (siendo la versión 7 la actual y utilizada en el presente proyecto). Dicha elección fue llevada a cabo en base a la versatilidad que ofrece dicha herramienta por su capacidad de creación de aplicaciones multiplataforma utilizando tecnologías web estándar como HTML, CSS y JavaScript, lo cual, a diferencia de Flutter (otro framework creado por Google y considerado por su corta curva de aprendizaje) el cual exige conocimiento previo del lenguaje

de programación Dart para su correcto uso. Si bien React Native no tiene este problema, demanda también el aprendizaje con herramientas particulares de su entorno teniendo su propia estrategia de ruteo de componentes y manejo de los mismos, mientras que otra ventaja que tiene Ionic es la integración que tiene con los frameworks de desarrollo web más utilizados del mercado los cuales son Vue, React y Angular, lo cual una vez se comprende el manejo con este entorno se puede desarrollar la respectiva aplicación sin mayores inconvenientes. A su vez, este último framework es mucho más estrechamente integrado con Ionic, lo que permite una sinergia natural entre ambos frameworks. Ionic se basa en Angular para la gestión de la lógica de la aplicación y la manipulación del DOM (Document Object Model), lo que garantiza un rendimiento óptimo y una experiencia de usuario fluida. Esta integración también simplifica el desarrollo, ya que los desarrolladores pueden aprovechar las características y funcionalidades de Angular dentro del entorno de Ionic, como el enrutamiento, la inyección de dependencias, y la modularidad del código.

Partiendo de las visibles ventajas que ofrece integrar Ionic y Angular, la elección de este último fue clara incluso percibiendo de que conlleva una reconocible curva de aprendizaje, ya que el mismo se basa en el Paradigma Orientado a objetos (POO) definiendo sus componentes como un conjunto conformado por la plantilla HTML, una hoja de estilo y la clase de Typescript que contiene la lógica del mismo; lo cual entonces permite aplicar patrones de diseño (como Modelo-Vista-Controlador Y Modelo-Vista-VistaModelo) frente a claros problemas que puede llegar a enfrentar el proyecto como son la escalabilidad, el mantenimiento y la reusabilidad de código y la robustez. En síntesis, Angular es muy recomendable para proyecto de mediana a gran escala lo cual lo convierte en una elección ideal para este desarrollo. Además, Angular proporciona un conjunto completo de herramientas y funcionalidades para el desarrollo de aplicaciones móviles complejas, incluyendo enrutamiento, gestión de formularios, comunicación con servidores, entre otros.

Ya, por último, se optó a Capacitor para la construcción de la aplicación nativa en conjunto de los frameworks previamente mencionados, por su capacidad para generar aplicaciones móviles para múltiples plataformas, incluyendo iOS, Android y la web. Esta capacidad multiplataforma ofrece una gran flexibilidad y alcance, permitiendo que la aplicación de autogestión estudiantil sea accesible desde una amplia gama de dispositivos y sistemas operativos. También deja a disposición del desarrollador el acceso a distintas funcionalidades nativas del dispositivo, como la cámara, el GPS, los sensores de movimiento y mucho más, a través de una API unificada. Por lo que ofrece varias oportunidades de mejora continua para futuras adiciones al proyecto.

Uso de npm vs pnpm:

Otro aspecto a tener en cuenta en el desarrollo de la aplicación fue la gestión de las dependencias y paquetes que tiene una influencia directa en la productividad, la estabilidad y la escalabilidad del proyecto. Ante esto, se optó por pnpm el cual es un gestor de paquetes de código abierto para NodeJS, diseñado para optimizar la instalación y gestión de dependencias de proyectos. Gracias algoritmo de almacenamiento que usa, lleva a cabo proceso de instalación de dependencias más optimizado que el gestor npm clásico, empleando los mismos comandos; por ende, no requiere de una curva de aprendizaje para migrar de un gestor de paquetes a otros. También utiliza un algoritmo de resolución de dependencias basado en el principio de "instalación hoja", lo que significa que solo se instalan las versiones necesarias de las dependencias en cada proyecto. Esto ayuda a prevenir los conflictos de versiones y garantiza la consistencia y la estabilidad del proyecto a lo largo del tiempo.

Compatibilidad Multiplataforma:

En el desarrollo de este proyecto, se eligieron Capacitor y Angular Ionic para construir una aplicación que se puede ejecutar en múltiples plataformas, incluyendo navegador web, Android y iOS. No obstante, la orientación principal se centra en dispositivos móviles.

Debido a las restricciones que impiden trabajar con dispositivos iOS, detalladas en el apartado correspondiente, se utilizó Capacitor CLI en conjunto con Android Studio para compilar el proyecto y

generar el archivo APK necesario para su instalación en dispositivos Android. Un APK (Android Package) es un formato de archivo utilizado para distribuir e instalar aplicaciones en dispositivos Android, asegurando que el software se ejecute de manera adecuada en el sistema operativo.

Esta elección asegura que la aplicación pueda ejecutarse eficazmente en la mayoría de los dispositivos Android disponibles en el mercado. Según la documentación de Capacitor, se admite Android Support API 22+ (Android 5.1 o versiones posteriores), lo que representa más del 99% del mercado de Android. Capacitor requiere un WebView de Android con la versión de Chrome 60 o superior. En dispositivos con Android 5-6 y 10+, Capacitor utiliza el Android System WebView, mientras que en versiones 7-9, Google Chrome proporciona el WebView. Esto garantiza una experiencia de usuario consistente y eficiente en una amplia gama de dispositivos Android, asegurando compatibilidad y optimización del rendimiento en diferentes entornos operativos.

Gestión de dependencias:

En este proyecto, se utiliza pnpm para gestionar las dependencias, lo cual ofrece varias ventajas en el manejo de paquetes. Una de las principales razones para elegir pnpm es su eficiente control de versiones, que ayuda a evitar conflictos entre las distintas versiones de las bibliotecas utilizadas. Esto asegura que el proyecto se mantenga estable y funcional a lo largo del tiempo.

Además, pnpm permite una clara diferenciación entre las dependencias de desarrollo y las de producción. Las dependencias de desarrollo son aquellas necesarias únicamente durante el proceso de desarrollo, mientras que las de producción son críticas para el funcionamiento final de la aplicación. Esta distinción facilita un entorno de trabajo más ordenado y optimizado.

En cuanto a la seguridad, se emplea el comando *pnpm audit* para auditar nuestras dependencias. Esta herramienta escanea las bibliotecas instaladas en busca de vulnerabilidades conocidas, ayudando a identificar y solucionar posibles riesgos de seguridad. Aunque en el presente proyecto se emplean paquetes ampliamente reconocidos de la fuente npmjs, es fundamental reconocer que no se garantiza automáticamente que todos los proyectos públicos estén libres de vulnerabilidades. Por ello, la auditoría de las dependencias es una práctica esencial.

El uso de pnpm no solo mejora la eficiencia en la gestión de paquetes, sino que también refuerza la seguridad del proyecto, al proporcionar mecanismos que permiten mantener las dependencias actualizadas y seguras. Es responsabilidad de los desarrolladores revisar regularmente los informes de seguridad y mitigar los riesgos, asegurando así un entorno de desarrollo robusto y confiable.

Finalmente se utilizó Dependabot de GitHub para gestionar y monitorear las dependencias de manera proactiva. Dependabot es una herramienta que automáticamente verifica las dependencias de un proyecto y alerta sobre posibles problemas, como vulnerabilidades o actualizaciones necesarias. Al integrarse con el repositorio de GitHub, Dependabot genera automáticamente pull requests para actualizar las dependencias a versiones más recientes y seguras, asegurando que el proyecto se mantenga actualizado y protegido contra vulnerabilidades conocidas. Esta funcionalidad no solo mejora la seguridad del proyecto, sino que también simplifica la tarea de mantenimiento, permitiendo a los desarrolladores centrarse en otras áreas críticas del desarrollo.

Arquitectura del proyecto:

Orientado a la arquitectura propia del proyecto, se utilizaron diversos patrones y prácticas con el fin de poder preservar una estructura escalable y clara, tanto definiendo los componentes y los respectivos módulos que conforman al proyecto. Se adoptó el estándar kebab-case para el nombramiento de los módulos, lo que garantizó coherencia y legibilidad en la estructura del proyecto. Esta elección se basó en las convenciones recomendadas por la comunidad de Angular y en su capacidad para facilitar la identificación de módulos y componentes.

En cuanto a los componentes, se optó por utilizar camelCase en su nombramiento para mantener consistencia con las convenciones de Angular y mejorar la claridad del código. Los componentes se estructuraron de manera modular y reutilizable, siguiendo las mejores prácticas de desarrollo de Angular.

Para los componentes genéricos y los pipes, se implementó una estrategia de standalone, lo que implicó la definición de archivos pequeños y autónomos que encapsulaban la lógica de manera sencilla y versátil. Esta aproximación permitió una mayor flexibilidad y reutilización de los componentes en diferentes partes de la aplicación.

Al margen de esto, para que el desarrollo del proyecto se realice de manera conjunta, es necesario trabajar con una herramienta de control de versión denominada Git, que permite gestionar distintas versiones del proyecto tanto de manera local como un repositorio gratuito en la nube denominado GitHub en el cual también yacerá la documentación. Para este trabajo en conjunto se optará por modelo de creación de ramas el modelo “Gitflow” basado en el uso de 2 ramas principales denominadas rama “main” (que contiene el historial de publicación oficial) y la rama “development” que sirve como rama de integración para las funcionalidades a crear e incorporar. Esto se puede visualizar en la siguiente ilustración:

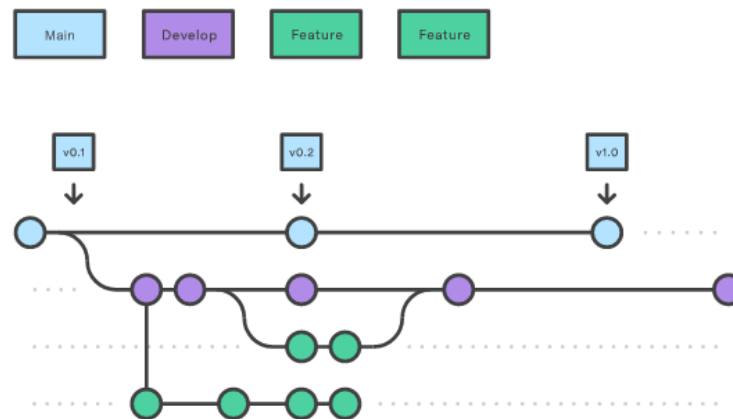


Ilustración 4: “Representación gráfica del modelo Gitflow”

Adicionalmente, se implementaron herramientas en favor a la integración en el flujo de trabajo con Git acorde al modelo de creación de ramas elegido en este proyecto las cuales fueron Husky que permite automatizar acciones antes de que se completen ciertos comandos de Git y “Pretty-Quick”, definida como una dependencia que formatea selectivamente los archivos que han sido modificados en un commit, utilizando Prettier como motor de formateo. Integrando estas herramientas, permite automatizar los cambios en el formato de los archivos en función de un script detallado, evitando debates sobre la sintaxis y el estilo de codificación, ya que Prettier impone un estilo de codificación consistente en todo el código. Todo esto contribuye a una mejor calidad en el repositorio y facilita la revisión en el código puesto que todos respetan un mismo formato.

Diseño del proyecto:

El proceso de diseño y desarrollo de los diferentes módulos de la aplicación móvil se llevó a cabo mediante una metodología cuidadosamente planificada que se centró en la usabilidad, la estética y la funcionalidad. Para esto, se utilizó la herramienta de diseño Figma, que permitió crear prototipos interactivos y colaborativos que facilitaron la visualización y la iteración de los diseños. Algunos ejemplares pueden ser apreciados en la ilustración 3. Cabe destacar que esos son solo las pantallas más “representativas” de la aplicación (que presentan información de gran relevancia para el usuario final como son las notas, pago de boletas, eventos, entre otros).

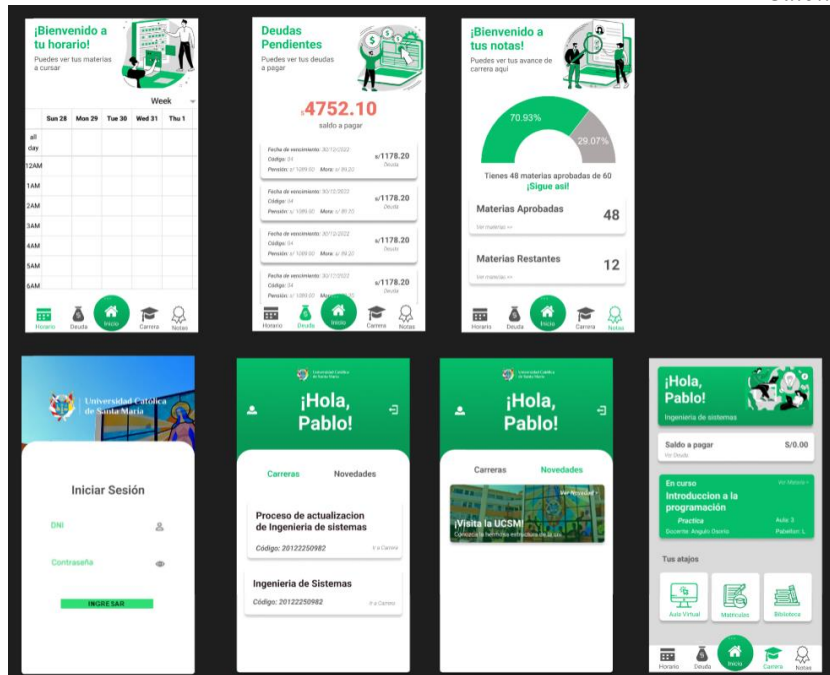


Ilustración 5: “Diseños preliminares de la aplicación realizados en Figma”

Los diseños de los módulos se basaron en las necesidades y requisitos planteados en la oficina de desarrollo de la universidad, contemplando también así la paleta de colores y las imágenes como iconos compartidos por el área de marketing de la misma institución en el manual para medios y/o plataformas digitales. En estos, a su vez, se priorizó la simplicidad, la coherencia y la accesibilidad en la disposición de los elementos visuales y en la navegación dentro de la aplicación.

Una vez definido el diseño, se procedió a hacer el diseño del prototipo acorde a las pautas marcadas y para poder garantizar una apariencia y una estructura consistentes en toda la aplicación, se empleó el framework PrimeFlex, que proporcionó una serie de clases CSS predefinidas que permitió un desarrollo más eficiente y una mayor cohesión en el diseño de la interfaz de usuario.

Uno de los grandes desafíos en cuanto al diseño que se presentaron en el proyecto fue la integración de un calendario, el cual permitiese mostrar las materias en su respectivo rango horario. Como solución se propuso la librería Ionic2-Calendar. Esta herramienta ofreció una solución robusta y altamente personalizable para la visualización y gestión de eventos en la aplicación. Se realizaron adaptaciones específicas para asegurar la integración fluida del calendario con el resto de los módulos y la coherencia con el diseño general de la aplicación como se puede apreciar en la siguiente ilustración (Ilustración 4):



Ilustración 6: “Resultado final del horario del alumno usando Ionic2-Calendar”

Además, se seleccionaron cuidadosamente los componentes de Ionic que se alineaban con las buenas prácticas en UX/UI y proporcionaban una experiencia de usuario intuitiva y atractiva. Estos componentes fueron utilizados en diversas partes de la aplicación, desde formularios y listas hasta elementos de navegación y botones de acción.

Por último, para la visualización de datos y análisis para presentar el avance de carrera, se implementó la biblioteca Charts.js, que permitió la creación de gráficos interactivos y dinámicos. Estos gráficos proporcionaron a los usuarios una representación visual clara y comprensible de los datos relevantes, contribuyendo así a una toma de decisiones informada y eficaz como se puede visualizar en la ilustración 5:



Ilustración 7: “Nuevo estudiante de prueba sin avance de carrera presente.”

Estas medidas implementadas a modo de poder arribar a un diseño efectivo fueron evaluadas por medio de una técnica de relevamiento de datos denominada “System Usability Scale” para tener una visión global de las evaluaciones subjetivas de usabilidad y poder en última instancia evaluar la eficacia (si los usuarios pueden alcanzar con éxito sus objetivos), eficiencia (cuánto esfuerzo y recursos se gasta en el logro de esos objetivos) y satisfacción (fue satisfactoria la experiencia). Esto se abordará más en detalle en el capítulo de “Resultados”.

Desarrollo de la aplicación:

En este proyecto de Angular (Sandoval, n.d.), se optó por una combinación de dos arquitecturas. La primera es el patrón tradicional de módulos "descompuestos", que implica la separación de lógica, plantilla, hoja de estilo y el archivo de módulo en sí, donde se gestionan las importaciones y exportaciones. La segunda arquitectura es la basada en componentes standalone, introducida a partir de Angular 14, que propone unificar todos estos elementos en un solo archivo.

Esta arquitectura híbrida resultó útil al permitir el uso de módulos descompuestos en situaciones donde la lógica es extensa o la pantalla requiere una estructura compleja, como en cada una de las vistas principales de la aplicación. Por otro lado, se optó por la estructura standalone para componentes individuales, tanto de la interfaz de usuario como personalizados, así como para los pipes de Angular. Esto se realizó con el objetivo de mantener el proyecto más ordenado y modular.

La aplicación sigue la jerarquía clásica de Angular, comenzando con una carpeta app como módulo raíz. A partir de esta base, se organizó el proyecto de la siguiente manera:

- **Directorio core:** Aquí se definen elementos reutilizables a lo largo del proyecto. Este directorio incluye servicios básicos que pueden ser utilizados posteriormente para los servicios específicos requeridos en el desarrollo actual. Esta estructura facilita la reutilización de código y la gestión eficiente de los recursos dentro del proyecto.
- **Directorio modules:** En este directorio se definen todos los módulos descompuestos, que se cargarán de manera diferida (lazy loading) al navegar a ellos. Están organizados en módulos públicos (pantallas que no requieren un token de sesión, como la de inicio de sesión) y privados (aquellos que sí lo requieren, como las pantallas de inicio, horarios, eventos, entre otros).
- **Directorio project:** A diferencia del directorio core, aquí se definen herramientas personalizadas para el proyecto, incluyendo servicios de Angular para realizar peticiones y recuperar datos, así como interfaces que ayudan a definir el tipo de información que se manejará, mejorando así la gestión de los datos.
- **Directorio shared:** Este directorio contiene componentes standalone, tanto personalizados (aquellos con estilos y funciones específicas para instancias determinadas de la aplicación) como de interfaz de usuario (componentes genéricos que pueden ser importados y personalizados según se requiera). Esto facilita la reutilización y adaptación de los componentes en diferentes partes del proyecto.

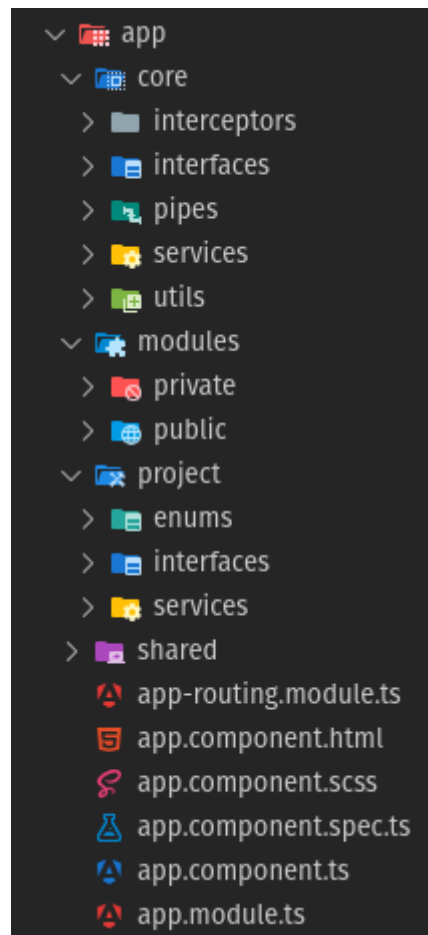


Ilustración 8: “Jerarquía de directorios del proyecto de aplicación móvil”

En cuanto a la gestión del token de sesión, así como los datos sensibles como el DNI y el enrollment ID, estos se manejan de manera segura dentro de la aplicación Angular. El token de sesión, una vez generado durante el inicio de sesión mediante el endpoint `api/login`, se almacena de forma segura en el almacenamiento local del navegador o dispositivo móvil. Junto con el token, se guarda también el DNI y el enrollment ID necesarios para las peticiones a la API que requieren autenticación.

Es importante destacar que estos datos sensibles se eliminan automáticamente del almacenamiento local cuando el usuario cierra la aplicación o realiza un cierre de sesión explícito utilizando el endpoint `api/logout`. Este enfoque asegura que la información personal y los tokens de sesión sean gestionados de manera eficiente y segura, siguiendo las mejores prácticas de seguridad y privacidad de datos en el desarrollo de aplicaciones web y móviles.

Además, se crearon accesos fáciles y directos que redireccionan a las páginas web de otros desarrollos de la universidad, como la biblioteca virtual, la plataforma de aulas virtuales y la plataforma de matrículas. Estos accesos se implementaron utilizando componentes de Angular que permiten la redirección mediante enlaces intuitivos y visibles dentro de la aplicación. Esto no solo mejora la accesibilidad y la experiencia del usuario, sino que también facilita la integración de los diversos servicios proporcionados por la universidad en una sola aplicación cohesiva, permitiendo a los estudiantes y al personal académico acceder rápidamente a los recursos necesarios para sus actividades académicas y administrativas.

Seguridad:

Para evaluar la seguridad en la aplicación móvil se fue evaluando en base a los potenciales riesgos que se presenta en este tipo de sistemas. En el OWASP Mobile Top 10 se destacan diez riesgos críticos que deben ser considerados por desarrolladores y profesionales de la seguridad. Cada uno de estos riesgos representa una preocupación significativa en la era actual de la movilidad y el acceso a la información a través de dispositivos móviles. A partir de esto, se detallará cada riesgo identificado y su correspondiente solución:

M1: Uso Incorrecto de Credenciales

Este riesgo se refiere a las prácticas inapropiadas relacionadas con el manejo de credenciales de autenticación en aplicaciones móviles. El manejo incorrecto de estas presenta un riesgo significativo de seguridad que puede ser explotado por agentes malintencionados para obtener acceso no autorizado a datos sensibles y funcionalidades críticas. Este riesgo puede manifestarse a través de varios escenarios de ataque, cada uno con sus propias implicaciones y posibles consecuencias:

1. Credenciales Duras (Hardcoded):

Un atacante examina el código fuente de la aplicación móvil y descubre la presencia de credenciales codificadas (hardcoded) dentro del mismo. Esta tiene un impacto potencial en el cual el atacante puede utilizar estas credenciales para acceder sin autorización a funcionalidades sensibles o al mismo servicio back-end asociado, comprometiendo así la seguridad de la aplicación y la privacidad de los datos.

Solución Propuesta: Se propuso evitar el uso de credenciales codificadas en el código fuente. En su lugar, se optó por utilizar variables de entorno seguras o servicios protegidos para gestionar y acceder a las credenciales las cuales son muy buenas prácticas incorporadas en el ámbito de desarrollo web con Angular y el cual de igual forma se puede proyectar en el desarrollo móvil nativo con Ionic y Capacitor. Además, se implementó la autenticación segura mediante el uso de tokens de acceso (JWT), en el back-end con Django Rest Framework.

2. Transmisión Insegura de Credenciales:

Durante la transmisión de datos entre la aplicación móvil y los sistemas back-end, las credenciales son enviadas de manera no segura, por ejemplo, sin cifrado o a través de canales inseguros. En este caso, un atacante puede interceptar las credenciales transmitidas de manera insegura y utilizarlas para hacerse pasar por un usuario legítimo y acceder de manera no autorizada a la aplicación.

Solución Propuesta: Se propuso utilizar protocolos de seguridad como HTTPS para cifrar la comunicación entre la aplicación móvil y los servidores back-end, garantizando una transmisión segura de las credenciales. Esto es posible empleando el módulo `HttpClient` de Angular, que

proporciona soporte nativo para HTTPS, lo que significa que puede cifrar la comunicación entre tu aplicación Angular Ionic y el servidor back-end de Django. Esto ayuda a garantizar que las credenciales y otros datos sensibles transmitidos entre la aplicación y el servidor estén protegidos de la interceptación y la manipulación por parte de terceros malintencionados. Esto solo limitado en el caso actual del prototipo, ahora por otro lado, en la UCSM se dispone actualmente en los servidores donde se alojará el servicio back-end desarrollado firewalls de próxima generación de Palo Alto que ofrecen una amplia gama de características de seguridad avanzadas, incluida la inspección de tráfico SSL/TLS. Esto significa que el firewall puede inspeccionar el tráfico HTTPS entrante y saliente para detectar y prevenir amenazas cibernéticas, incluidos los intentos de explotar vulnerabilidades de transmisión de datos.

3. Almacenamiento Inseguro de Credenciales:

Las credenciales de los usuarios son almacenadas de manera insegura en el dispositivo móvil, por ejemplo, en texto plano o sin cifrado adecuado. En este caso, un atacante con acceso físico al dispositivo puede extraer las credenciales almacenadas y utilizarlas para acceder sin autorización a la cuenta del usuario.

Solución Propuesta:

Se optó por evitar el almacenaje de los datos en el dispositivo del usuario. En su lugar, almacenar tokens de acceso en el almacenamiento seguro del dispositivo. Esto se puede lograr en el contexto de Angular, usando el almacenamiento local (Local Storage), el cual proporciona un mecanismo conveniente y seguro para almacenar información en el dispositivo del usuario de forma persistente, lo que permite que la aplicación acceda a estos datos incluso después de cerrar y volver a abrir la aplicación.

A su vez, se planteó tanto expiración de token de acceso provisto por servicio back-end como también se implementó prácticas de almacenamiento seguro de datos en el back-end, como son el cifrado de contraseñas usando la técnica de “hashing” definida en el contexto Django con el módulo “django.contrib.auth.hashers” el cual proporciona una serie de algoritmos de hash seguros que pueden ser utilizados para almacenar contraseñas de manera segura en la base de datos. Estos algoritmos incluyen PBKDF2, bcrypt, SHA-256, entre otros. Al utilizar uno de estos algoritmos, Django automáticamente aplica técnicas de salting y hashing para proteger las contraseñas almacenadas.

M2: Seguridad Inadecuada en la Cadena de Suministro

Este riesgo resalta la importancia de asegurar que los componentes y bibliotecas de terceros utilizados en aplicaciones móviles no introduzcan vulnerabilidades. La seguridad de la cadena de suministro es esencial para prevenir ataques a través de componentes comprometidos. Para mitigar este riesgo, se debe realizar un escaneo de seguridad de bibliotecas y componentes de terceros antes de su implementación de manera regular.

Este riesgo puede manifestarse a través de varios escenarios de ataque, cada uno con sus propias implicaciones, posibles consecuencias y sus soluciones:

1. Inyección de Malware:

En este escenario, un atacante inserta código malicioso en una aplicación móvil popular durante la fase de desarrollo. Posteriormente, el atacante firma la aplicación con un certificado válido y la distribuye a través de la tienda de aplicaciones, eludiendo los controles de seguridad. Los usuarios descargan e instalan la aplicación infectada, lo que resulta en el robo de credenciales de inicio de sesión y otros datos sensibles.

Solución Propuesta: Puesto que el desarrollo del mismo sistema estuvo coordinado por el equipo de desarrollo de la UCSM en repositorios privados con un control supervisado y periódico, y con dependencias verificadas; se pudo garantizar un desarrollo seguro. En cuanto a la distribución de la aplicación, se hará a través de sitios seguros de compleja admisión como Play Store y App Store para los entornos de Android y IOS, con información extensa para

distinguir las y con información del equipo de desarrollo. Sin embargo, se debería contemplar el definir un proceso de monitoreo continuo de las tiendas de aplicaciones para detectar y responder rápidamente a cualquier actividad sospechosa o aplicación maliciosa que pueda surgir.

También proporcionar educación y orientación a los estudiantes de la Universidad Católica de Santa María sobre la importancia de descargar aplicaciones solo de fuentes confiables y de verificar la autenticidad de la aplicación antes de instalarla.

2. *Explotación de Vulnerabilidades en Bibliotecas de Terceros:*

En este escenario, un atacante aprovecha vulnerabilidades en bibliotecas de terceros utilizadas en la aplicación móvil. Esto puede permitir al atacante insertar código malicioso o realizar actividades no autorizadas en el dispositivo del usuario.

Solución Propuesta: Se emplearon dependencias confiables como Ionic y Capacitor, las cuales son respaldadas por organizaciones reconocidas, lo que garantizan tanto un constante soporte como una garantía en cuanto algún tipo de riesgo de vulnerabilidad. Sin embargo, se debería realizar una evaluación exhaustiva de las dependencias y mantenerlas actualizadas regularmente.

M3: Autenticación/Autorización Insegura

La autenticación y autorización incorrectamente implementadas pueden exponer aplicaciones móviles a riesgos graves. Garantizar una autenticación sólida y autorización adecuada es crucial para proteger los datos y la funcionalidad de la aplicación.

1. Solicitudes de Servicio Ocultas:

En este escenario, los desarrolladores asumen que solo los usuarios autenticados podrán generar solicitudes de servicio que la aplicación móvil envía a su back-end para su procesamiento. Sin embargo, el código del servidor no verifica que la solicitud entrante esté asociada a un usuario conocido, lo que permite a los atacantes enviar solicitudes de servicio al back-end y ejecutar funcionalidades de manera anónima que afectan a usuarios legítimos de la solución.

Solución propuesta:

Se implementó una verificación exhaustiva de la identidad del usuario en el back-end para garantizar que todas las solicitudes de servicio estén asociadas a un usuario autenticado y autorizado. Esto se puede lograr gracias a la estructuración del mismo y como maneja su respectivo controlador a la hora de hacer la autenticación. También se utilizaron tokens de autenticación robustos, como JWT, para validar la identidad de los usuarios y prevenir la ejecución de funcionalidades de manera anónima.

2. Dependencia de la Interfaz:

En este escenario, los desarrolladores suponen que solo los usuarios autorizados podrán ver la existencia de una función específica en la aplicación móvil. Sin embargo, el código del back-end no verifica la identidad asociada a la solicitud, lo que permite a los atacantes realizar funcionalidades administrativas remotas utilizando cuentas de usuario con privilegios bajos.

Solución propuesta: El servicio web desarrollado con Django, cuenta con el módulo “auth”, el cual provee un modelo para la creación de usuarios, este mismo se hereda para la creación del modelo “Student” empleado para cada una de las instancias del usuario final, el cual tiene únicamente permisos de lectura para recuperar información del servicio actual.

3. Requisitos de Usabilidad

En este escenario, la aplicación permite contraseñas de solo 4 dígitos debido a requisitos de usabilidad. Sin embargo, esta longitud de contraseña facilita que los atacantes deduzcan rápidamente las contraseñas originales utilizando técnicas como las tablas hash arcoíris.

Solución Propuesta:

Se incrementó la complejidad de las contraseñas estableciendo requisitos mínimos de longitud y complejidad para mejorar la seguridad.

4. Referencia de Objeto Directo Insegura

En este escenario, un usuario puede modificar el ID del actor en una solicitud a la API y acceder a la información de otras cuentas de usuario debido a una validación insuficiente en el back-end.

Solución Propuesta:

Se implementó controles de autorización a nivel de objeto para garantizar que los usuarios solo puedan acceder a los recursos que les corresponden. Esto se logró en el proceso de inicio de sesión, puesto que únicamente si se obtiene el documento de identidad y la contraseña adecuada se puede generar el token de acceso para iniciar la sesión correspondiente y habilitar el acceso a su respectiva información.

5. Transmisión de Roles de LDAP

En este escenario, el back-end no valida de forma independiente los grupos de LDAP reportados por el usuario, lo que permite a los atacantes reportar ser miembros de grupos LDAP arbitrarios y realizar funcionalidades administrativas.

Solución Propuesta:

Gracias al módulo “Auth” ya mencionado, el cual es proporcionado por Django, permite estructurar lo que se define como grupos de usuarios, de los cuales se pueden generar usuarios con ciertos permisos de administración, regularlos acorde a los requerimientos. A disposición del servicio móvil, únicamente se habilito el grupo con permisos para visualizar y recuperar información, mas no modificarla, lo cual garantiza la imposibilidad de acceso a funcionalidades administrativas.

M4: Validación Insuficiente de Entrada/Salida

La validación insuficiente y la falta de sanitización de los datos provenientes de fuentes externas pueden conducir a vulnerabilidades críticas en la seguridad de una aplicación móvil. Este tipo de fallas puede exponer a la aplicación a ataques específicos para entornos móviles, como inyecciones SQL, inyecciones de comandos y ataques de scripting entre sitios (XSS). Dichas vulnerabilidades pueden resultar en acceso no autorizado a datos sensibles, manipulación de la funcionalidad de la aplicación e incluso compromiso del sistema móvil en su totalidad.

1. Ejecución de Código Remoto mediante Entrada Maliciosa:

En este escenario, un atacante identifica una aplicación móvil que carece de una adecuada validación y sanitización de la entrada de datos. Aprovechando esta vulnerabilidad, el atacante crea una entrada maliciosa que contiene caracteres inesperados o manipulados. Al no existir una validación adecuada, la aplicación no logra manejar correctamente esta entrada, lo que lleva a vulnerabilidades de seguridad. Como resultado, el atacante logra ejecutar código arbitrario, obteniendo acceso no autorizado a los recursos del dispositivo y a datos sensibles del usuario.

Solución Propuesta:

Para abordar esta vulnerabilidad, se ha fortalecido la validación en la entrada de datos en la aplicación. Se han implementado filtros y validaciones estrictas para detectar y bloquear cualquier carácter inesperado o malicioso presente en la entrada de datos. Además, se han aplicado técnicas de limpieza de datos para asegurar que solo la información válida y segura sea procesada por la aplicación.

2. Ataques de Inyección mediante insuficiente validación de Salida:

En este escenario, un atacante identifica una aplicación móvil que carece de una adecuada validación y sanitización de la salida de datos. Aprovechando esta vulnerabilidad, el atacante explota un punto de entrada donde se procesa contenido generado por el usuario o datos no confiables. Mediante la creación de una entrada maliciosa que contiene código o scripts (por ejemplo, HTML, JavaScript, SQL), el atacante se aprovecha de la falta de validación de salida. Al someter la entrada manipulada a través de la interacción del usuario, la aplicación falla en validar, permitiendo la ejecución de código inyectado u operaciones no deseadas. El atacante

logra ejecutar ataques basados en inyección como Cross-site scripting (XSS) o inyección SQL, comprometiendo la integridad de la aplicación y obteniendo acceso a información sensible.

Solución Propuesta:

Para mitigar esta vulnerabilidad, se han implementado medidas para mejorar la validación y sanitización de la salida de datos en la aplicación. Se han introducido filtros y técnicas de codificación adecuadas para prevenir la ejecución de código malicioso o la manipulación de datos sensibles. También el empleo de Django Cors Headers, para la configuración CORS (Cross-Origin Resource Sharing) para proteger la aplicación contra ataques de inyección y otros tipos de vulnerabilidades.

3. Ejecución de código remoto mediante Salida Malformada:

En este escenario, un atacante identifica una aplicación móvil que procesa datos proporcionados por el usuario y genera salida dinámica. El atacante crea datos especialmente formateados que aprovechan la insuficiente validación de salida de la aplicación. El atacante envía los datos malformados a la aplicación, ya sea a través de interacción directa o mediante la explotación de una API expuesta. La aplicación no logra validar correctamente la salida generada, lo que permite que los datos manipulados por el atacante ejecuten código o desencadenen acciones no deseadas. Al explotar esta vulnerabilidad, el atacante logra la ejecución remota de código, obteniendo control sobre el dispositivo móvil, sus recursos o datos sensibles.

Solución Propuesta:

Para abordar esta vulnerabilidad, se han realizado mejoras significativas en la validación de la salida de datos en la aplicación como la definición de tipado fuerte se demanda su respectiva estructura tanto para el consumo de los datos recibidos como en la transmisión. Se han implementado filtros y técnicas de codificación avanzadas para detectar y bloquear cualquier salida malformada o manipulada por el usuario.

M5: Comunicación Insegura

La comunicación no segura puede exponer datos confidenciales a interceptación y manipulación por parte de atacantes impulsados por diferentes motivos, como robo de información sensible, espionaje, robo de identidad y más. Para abordar este riesgo, se utilizó conexiones seguras (HTTPS) para todas las comunicaciones entre la aplicación móvil y el servidor.

Si bien en el mismo se plantean diversos escenarios, están relacionados con la comunicación entablada entre la aplicación el servicio, y como se ha planteado previamente, esta intervenida por un cortafuegos de Palo Alto, el cual garantiza el bloqueo de peticiones de carácter inválido o no confiable. Ahora por cuestiones de correlación con otros servicios empleados en la universidad y bien no perjudicar la experiencia de usuario no se hace uso de una autenticación de 2 factores.

M6: Controles de Privacidad Inadecuados

La falta de controles de privacidad puede poner en riesgo la información personal de los usuarios. Es crucial que las aplicaciones móviles respeten la privacidad y cumplan con las regulaciones de protección de datos.

1. Insuficiente saneamiento de registros y mensajes de error:

La generación de registros y excepciones es esencial para garantizar la calidad de una aplicación productiva. Sin embargo, los registros y mensajes de error pueden contener información personal identificable (PII) si los desarrolladores eligen incluir estos datos en ellos. Esto puede ser un problema si se exponen a terceros o si los mensajes de error se muestran al usuario, lo que podría comprometer la privacidad.

Solución propuesta:

De los “logs” o registros generados en el proyecto, hacer una adecuada gestión para evitar la exposición del dato inconsistente; únicamente exponer la sección o línea de código que reporto la falla. Para ese adecuado control de errores, no solo basta con operaciones de manejo de errores, sino dar una correspondiente retroalimentación a la aplicación que lo haya solicitado.

2. Uso de PII en parámetros de consulta de URL:

Los parámetros de consulta de URL suelen utilizarse para transmitir argumentos de solicitud a un servidor. Sin embargo, estos parámetros son visibles al menos en los registros del servidor y, a menudo, también en las analíticas del sitio web y posiblemente en el historial del navegador local.

Solución propuesta:

Como se debe evitar información sensible por medio de los parámetros; a la misma se la incorporara a través del cuerpo de la petición.

3. Exclusión de datos personales en copias de seguridad/fallos en establecer hasFragileUserData:

La mayoría de la información personal identificable (PII) procesada por una aplicación se almacena en su espacio de trabajo (sandbox). La aplicación debe configurar explícitamente qué datos incluir en las copias de seguridad del dispositivo. Un atacante podría obtener un dispositivo y crear una copia de seguridad u obtener una copia de seguridad de otra fuente, desde la cual se podría extraer el contenido del espacio de trabajo.

Solución propuesta:

Esta aplicación desarrollada, no preserva ningún dato sensible que genere algún aspecto de vulnerabilidad durante la instalación de la misma y en su misma instancia.

M7: Protecciones Binarias Insuficientes

La falta de protecciones adecuadas en el nivel binario de una aplicación móvil puede exponerla a ingeniería inversa y ataques. Reforzar la seguridad en este nivel es esencial para proteger el código y los activos de la aplicación. Los escenarios contemplados en el M7 son:

1. Compromiso de Claves de API Sensibles:

Se plantea una situación en la cual una aplicación móvil emplea una interfaz de programación de aplicaciones (API) comercial para llevar a cabo funciones que requieren autenticación. Sin embargo, las claves de acceso a esta API se encuentran incrustadas directamente en el código binario de la aplicación, sin medidas de protección adicionales. En este escenario, un agente malintencionado podría aprovechar técnicas de ingeniería inversa para extraer dichas claves de API del código binario de la aplicación. Esta vulnerabilidad podría permitir al atacante acceder y potencialmente utilizar la API de manera no autorizada, lo que podría resultar en pérdidas financieras significativas para el proveedor de la aplicación o comprometer la integridad y seguridad de la misma.

Solución propuesta:

En el presente desarrollo no se incluye servicios de terceros para alguna funcionalidad en particular; únicamente consume del servicio desarrollado con DRF.

2. Manipulación de Verificaciones de Pago y Licencia:

Se presenta un escenario en el cual una aplicación móvil ofrece ciertas funcionalidades de forma gratuita, con la opción de acceso completo mediante un pago. Sin embargo, si las verificaciones de pago y licencia se encuentran débilmente protegidas dentro del código binario de la aplicación, existe la posibilidad de que un atacante manipule estas verificaciones para desactivar los controles de pago y licencia. Como resultado, los usuarios podrían acceder a todas las características de la aplicación de manera gratuita, lo que podría acarrear pérdidas financieras considerables para el desarrollador de la misma.

Solución propuesta:

En el presente desarrollo no se incluye servicios de terceros para alguna funcionalidad en particular; únicamente consume del servicio desarrollado con DRF.

3. Exposición de Modelos de Inteligencia Artificial Confidenciales:

Se contempla un escenario en el cual una aplicación móvil destinada al ámbito médico hace uso de un modelo de inteligencia artificial (IA) para proporcionar respuestas a las consultas de los usuarios. Si este modelo de IA se encuentra integrado directamente en el código binario de la aplicación sin las debidas medidas de protección, existe el riesgo de que un atacante pueda acceder y extraer dicho modelo de IA del código binario. Esta vulnerabilidad podría permitir al atacante vender el modelo de IA a competidores, comprometiendo así la propiedad intelectual y la ventaja competitiva de la aplicación médica.

Solución propuesta:

En el presente desarrollo no incluye ninguno modelo de inteligencia artificial ya desarrollado para alguna funcionalidad en particular; únicamente consume del servicio desarrollado con DRF.

M8: Configuración de Seguridad Incorrecta

La configuración incorrecta de la seguridad es un riesgo común en aplicaciones móviles. Garantizar una configuración adecuada es esencial para prevenir vulnerabilidades y exposición de datos. A continuación, se detalla los escenarios de ataque en caso de una configuración de seguridad Incorrecta:

1. Ajustes Predeterminados Inseguro:

Un escenario común involucra el lanzamiento de una aplicación móvil con ajustes predeterminados que tienen configuraciones de seguridad débiles. Esto incluye el uso de protocolos de comunicación inseguros, dejar nombres de usuario y contraseñas predeterminados sin cambios, y no deshabilitar las funciones de depuración en las compilaciones de producción. Los atacantes aprovechan estas configuraciones deficientes para obtener acceso no autorizado a datos sensibles o realizar acciones maliciosas.

Solución propuesta:

La configuración que se definirá para la API REST será contemplando varios aspectos detallados en el apartado de back-end; sin embargo, se puede describir a la misma como robusta y completa, siguiendo las consideradas “buenas prácticas” en el desarrollo de servicios con DRF.

2. Configuración de Ruta del Proveedor de Archivos Insegura:

Una aplicación móvil expone su ruta raíz en un proveedor de contenido de archivos exportado, lo que permite que otras aplicaciones accedan a sus recursos. Esta configuración inadecuada puede exponer datos sensibles o permitir el acceso no autorizado a recursos de la aplicación.

Solución propuesta:

Esta restringido el acceso a la ruta raíz, únicamente se puede acceder a los aspectos de documentación de cada endpoint y al admin, pero en el entorno de desarrollo, cuya instancia está limitada a los miembros del equipo de desarrollo

3. Permisos de Almacenamiento Excesivamente Permisivos:

Una aplicación móvil almacena las preferencias compartidas de la aplicación con permisos de lectura mundiales, lo que permite que otras aplicaciones las lean. Esto puede resultar en la exposición de datos sensibles a otras aplicaciones, comprometiendo la privacidad y la seguridad de los usuarios.

Solución Propuesta:

Los datos cargados en la aplicación solo perduran en el ciclo de vida de la misma aplicación, por ende, no hay una exposición real que ponga en peligro a la aplicación desarrollada.

4. Actividad Exportada:

Una aplicación móvil exporta alguna actividad que está destinada para uso interno, lo que proporciona a los atacantes una superficie de ataque adicional para la aplicación. Esto puede ser aprovechado por los atacantes para realizar acciones no autorizadas o manipular el comportamiento de la aplicación.

Solución propuesta:

Ya que alcance del proyecto solo está limitado a la lectura de información por parte del usuario, no hay riesgo real en casa de una manipulación sobre la aplicación desarrollada.

5. **Permisos Innecesarios:**

Una aplicación móvil solicita permisos excesivos que no son esenciales para su funcionalidad básica. Por ejemplo, una simple aplicación de linterna solicita acceso a los contactos, la ubicación y la cámara del usuario. Esto expone los datos del usuario a riesgos innecesarios, ya que la aplicación podría potencialmente abusar de los permisos otorgados o filtrar información sensible de manera no intencional.

Solución Propuesta:

La aplicación actual no requiere permisos para aspectos nativos del dispositivo.

M9: Almacenamiento Inseguro de Datos

El almacenamiento inseguro de datos puede dar lugar a fugas de información confidencial. Proteger adecuadamente los datos almacenados en dispositivos móviles es un componente crítico de la seguridad de la aplicación. Los escenarios de ataques planteados fueron:

1. **Almacenamiento de Contraseñas en Texto Plano:**

La aplicación móvil almacena las contraseñas de los usuarios en formato de texto plano dentro de una base de datos local o archivo, facilitando a un atacante recuperar y abusar de estas credenciales si obtienen acceso no autorizado al dispositivo.

Solución propuesta:

No se almacenará ningún dato que presente algún peligro en la privacidad de los usuarios de la aplicación.

2. **Almacenamiento Local No Seguro:**

La aplicación móvil guarda datos sensibles del usuario, como información de identificación personal (PII), localmente en el dispositivo sin utilizar controles de acceso o cifrado adecuados. Esto permite que cualquier persona con acceso físico al dispositivo extraiga y visualice los datos.

Solución propuesta:

Si bien se almacena en el “Local Storage” el token de acceso que le permite al usuario hacer las solicitudes; la misma es de carácter temporal y finita, es decir, se vence dicho token, obligando al usuario el volverse a registrar.

3. **Caché de Datos No Segura:**

La aplicación móvil almacena en caché datos sensibles, como tokens de autenticación de usuario o información de sesión, sin implementar medidas de seguridad apropiadas. Si un atacante obtiene acceso a la caché del dispositivo, puede obtener estas credenciales y personificar al usuario.

Solución propuesta:

Los tokens de acceso tienen un límite de duración, pasado ese tiempo es necesario volver a autenticar.

4. **Registro No Protegido:**

La aplicación móvil registra datos sensibles, incluidas las acciones del usuario, las respuestas de API o los mensajes de error, sin controles de seguridad adecuados. Esto puede provocar una

exposición no intencionada de información sensible si un atacante obtiene acceso al dispositivo o intercepta los archivos de registro.

Solución propuesta:

Los registros generados por la aplicación están limitados únicamente a fallos de aplicación, sin registro por inconsistencia de dato que de alguna manera perjudique al usuario.

5. **Configuración Insegura de Almacenamiento en la Nube:**

La aplicación móvil utiliza servicios de almacenamiento en la nube para almacenar datos de usuario, pero configura incorrectamente los permisos de almacenamiento, lo que permite el acceso no autorizado a la información almacenada. Esto puede provocar una fuga de datos o una exposición no autorizada de datos sensibles.

Solución Propuesta:

Los datos serán preservados en servidores protegidos de la Universidad Católica de Santa María.

6. **Manejo Incorrecto de Archivos Temporales:**

La aplicación móvil crea archivos temporales para procesar o almacenar datos sensibles, pero no maneja correctamente ni elimina estos archivos después. Esto deja la información sensible expuesta y vulnerable a accesos no autorizados.

Solución propuesta:

Ionic no proporciona una funcionalidad específica para manejar archivos temporales de forma nativa. Sin embargo, se siguieron prácticas de codificación segura a modo de evitar la exposición de información sensible en archivos temporales y asegurarse de eliminar los archivos temporales de manera segura después de su uso.

M10: Criptografía Insuficiente

La criptografía inadecuada puede resultar en la exposición de datos confidenciales. Utilizar cifrado sólido y técnicas criptográficas adecuadas es fundamental para proteger la información en aplicaciones móviles. Se definen los siguientes escenarios de ataque:

1. **Ataques de Intermediario (MitM)**

Un atacante intercepta la comunicación entre la aplicación móvil y el servidor. La utilización de cifrado débil permite al atacante descifrar los datos interceptados, modificarlos y volver a cifrarlos antes de enviarlos al destinatario previsto. Esto puede llevar a accesos no autorizados, manipulación de datos o la inserción de contenido malicioso.

Solución propuesta:

El cifrado del canal de comunicación está planteado en HTTPS en Django, en el cual se establece una capa de cifrado sobre la comunicación HTTP, lo que hace que sea mucho más difícil para un atacante interceptar y manipular los datos transmitidos entre la aplicación móvil y el servidor.

2. **Ataques de Fuerza Bruta**

Los atacantes prueban sistemáticamente varias combinaciones de claves hasta encontrar la correcta para descifrar los datos. La utilización de un cifrado débil puede reducir el tiempo necesario para tales ataques, exponiendo potencialmente información sensible.

Solución propuesta:

Se definió claves con un cierto grado de complejidad, y se definió limitadores de intentos a la hora de autenticar al usuario a modo que este limitado, las pruebas a fuerza bruta.

3. **Ataques de Degradación Criptográfica**

Las aplicaciones móviles pueden admitir múltiples protocolos o algoritmos de cifrado para establecer conexiones seguras. Si se permite la utilización de cifrado débil como opción secundaria, los atacantes pueden aprovechar esta debilidad y forzar a la aplicación a utilizar un cifrado débil. Como resultado, pueden descifrar los datos interceptados con mayor facilidad y lanzar ataques posteriores.

Solución propuesta:

Se adhiere a un algoritmo de cifrado hash a modo de garantizar la seguridad en las conexiones e intercambio de información.

4. **Vulnerabilidades en la Gestión de Claves**

Las prácticas débiles de gestión de claves pueden socavar la seguridad de los sistemas criptográficos utilizados en las aplicaciones móviles. Por ejemplo, si las claves de cifrado se almacenan de manera insegura o son fáciles de adivinar, los atacantes pueden obtener acceso no autorizado a las claves y descifrar los datos cifrados. Esto puede provocar violaciones de datos y vulneraciones de la privacidad.

Solución propuesta:

Definición de claves “fuertes” para cada usuario a modo de garantizar un grado alto de protección, y mitigar así la vulnerabilidad.

5. **Defectos en la Implementación Criptográfica:**

La criptografía débil también puede originarse a partir de defectos en la implementación en la propia aplicación móvil. Estos defectos pueden incluir el uso incorrecto de bibliotecas criptográficas, la generación insegura de claves, la generación inapropiada de números aleatorios o el manejo inseguro de funciones relacionadas con el cifrado. Los atacantes pueden aprovechar estos defectos para eludir o debilitar la protección de cifrado.

Solución propuesta:

Se emplean dependencias proporcionadas por el mismo framework de Django, el cual tiene un robusto nivel de seguridad, y herramientas como técnicas fáciles de implementar, pero complejas de eludir.

Back-end:

El conjunto de herramientas de software empleadas para el desarrollo del servicio web a consumir como back-end para el sistema de autogestión estudiantil en la Universidad Católica de Santa María fueron tanto el framework de Python llamado Django en la versión 4.2 en conjunto con el framework orientado al desarrollo de API REST, Django Rest Framework versión 3.15, el cual permite orientar el desarrollo del servicio web con la arquitectura REST, a modo de garantizar el uso de las mejores prácticas para el mismo. A su vez, se lo definió en un contenedor de Docker en conjunto de otro contenedor que contiene la base de datos de PostgreSQL; ambos coordinados y comunicados con Docker Compose. En el presente subapartado se irán describiendo y detallando con profundidad la elección de cada una de las tecnologías como su configuración. En adición, se puede visualizar en la próxima ilustración el repositorio disponible en GitHub del proyecto back-end:

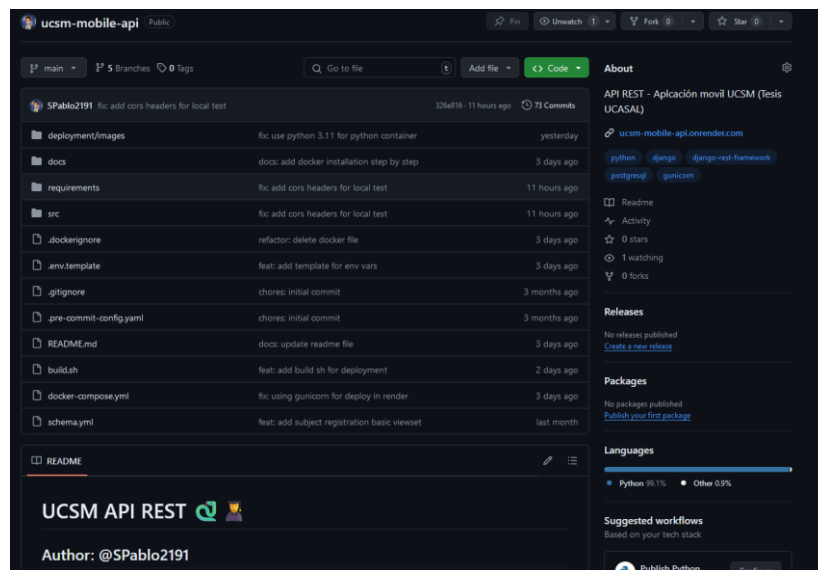


Ilustración 9 “Repositorio de GitHub de la API REST”

Selección del framework:

La selección del framework para el desarrollo del back-end de la aplicación móvil de la UCSM fue un proceso meticuloso que involucró la consideración de varios factores clave. En primera instancia, se optó para trabajar con el lenguaje de programación de Python debido a su facilidad de aprendizaje y uso, así como a su amplia comunidad y soporte, lo que facilita el desarrollo y la resolución de problemas. Python también se destaca por su versatilidad y escalabilidad, lo que lo hace adecuado para proyectos en crecimiento como la aplicación móvil de la UCSM. Por último, el equipo de desarrollo del ERP de la universidad, previamente a trabajado con varias tecnologías derivadas de este lenguaje, por ende, se puede omitir un proceso de capacitación para familiarizarse con dicho lenguaje, y directamente orientar los esfuerzos a la propia tecnología y proyecto.

En cuanto al framework, se decidió utilizar Django sobre alternativas como Flask o FastAPI debido a su conjunto completo de características listas para usar y su enfoque en la escalabilidad y robustez. Django ofrece funcionalidades integradas como un sistema de administración de bases de datos, autenticación de usuarios y seguridad integrada, lo que facilita un desarrollo más rápido y eficiente. Además, su documentación completa y comunidad activa de desarrolladores brindan soporte adicional durante el proceso de desarrollo.

Para la creación de la API REST que servirá como interfaz entre la aplicación móvil y el back-end, se optó por Django REST Framework en lugar de Django Ninja debido a su madurez, estabilidad y amplia gama de funcionalidades avanzadas. Django REST Framework ofrece características como serialización de datos, autenticación flexible y permisos basados en roles, lo que facilita el desarrollo de una API

REST completa y segura. Además, su flexibilidad y opciones de personalización permiten adaptar la API a los requisitos específicos de la aplicación móvil de la UCSM y agregar funcionalidades adicionales según sea necesario. Por último, la capacidad que tiene para responder ante proyectos de mediana o gran envergadura le permite resaltar por encima de Django Ninja, siendo este último más orientado a proyectos más simples y puntuales. (Django Rest Framework vs. Django Ninja: Choosing the Right Framework for Your API | Medium, StartxLabs Technologies)

Evaluación de Alternativas de Servicios en la Nube:

En el desarrollo de un Producto Mínimo Viable (MVP), se consideró fundamental optar por alternativas de servicios en la nube que ofrecieran planes gratuitos. Entre las opciones más reconocidas se evaluaron Vercel, Render y Fly.io, cada una con características específicas que se alinean con las necesidades del proyecto.

Las tres opciones en sus planes gratuitos son de tipo serverless, lo que significa que permiten ejecutar aplicaciones y servicios sin necesidad de gestionar servidores físicamente. En un entorno serverless, los recursos se escalan automáticamente según la demanda, y el proveedor de servicios en la nube se encarga de la infraestructura subyacente. Esto simplifica el desarrollo y despliegue de aplicaciones, permitiendo a los desarrolladores centrarse en el código sin preocuparse por la administración de servidores.

A continuación, se detalla cada una:

- **Vercel:** es una plataforma unificada en la nube que permite a los desarrolladores desplegar, gestionar y escalar aplicaciones y sitios web. Ofrece funcionalidades como despliegues automatizados, dominios personalizados y una potente interfaz de línea de comandos (CLI). Además, cuenta con integraciones con servicios populares como GitHub, Slack y Zapier, lo que facilita la colaboración y el flujo de trabajo (*Vercel: Build and Deploy the Best Web Experiences with the Frontend Cloud*, n.d.).
- **Render:** se destaca por su capacidad para ayudar a equipos de software a desplegar productos de manera rápida y a cualquier escala. Es conocida por alojar desde prototipos sencillos hasta aplicaciones complejas con múltiples servicios, garantizando fiabilidad y tiempo de actividad constante. Render también proporciona la opción de crear una base de datos PostgreSQL con un plazo de uso gratuito de hasta 30 días, lo que resulta ideal para pruebas y desarrollo inicial. Aunque se desactiva por inactividad, una vez activo, presenta tiempos de respuesta eficientes, evaluados mediante pruebas en Postman visible en la ilustración presentada a continuación:

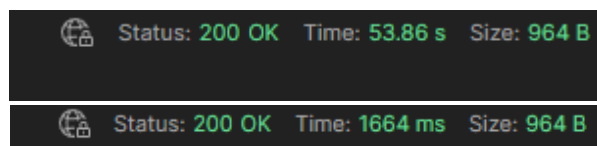


Ilustración 2: “Tiempos de respuesta de primera petición (en espera de reactivación de servicio) y una segunda petición (servicio activo)”

Además, ofrece documentación detallada para un despliegue adecuado y un tablero que permite el seguimiento del consumo del servicio basado en las peticiones realizadas.

- **Fly.io:** se enfoca en ofrecer una infraestructura de nube pública moderna y flexible. Sus máquinas virtualizadas permiten un lanzamiento instantáneo y la ejecución de contenedores hardware-virtualizados, con opciones que van desde una sola solicitud HTTP hasta semanas de tiempo de actividad. Sin embargo, presenta un plazo de uso gratuito muy breve para mantener instancias operativas, lo que limita su utilidad para este tipo de proyectos (*Deploy App Servers Close to Your Users · Fly*, n.d.).

Tras evaluar estas alternativas, se optó por Render para el despliegue de la API REST como servicio web para el consumo de la aplicación. Su versatilidad, combinada con la disponibilidad de una base de datos PostgreSQL y un tiempo de respuesta adecuado, hizo que se destacara frente a Fly.io, que tiene limitaciones en el plazo de uso, y Vercel, que está más orientado a tecnologías de JavaScript y carece de documentación extensa para otras tecnologías. A continuación, se puede observar el servicio web desplegado de forma exitosa en Render:

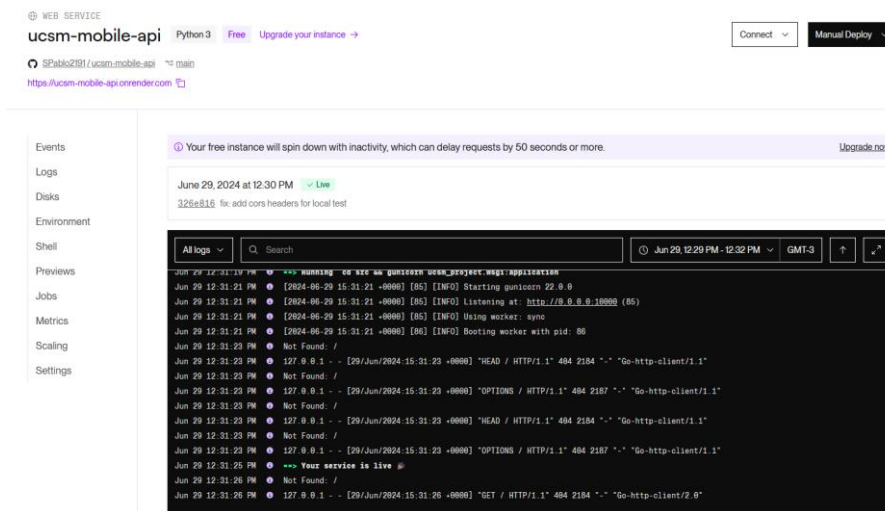


Ilustración 10 “Servicio Web desplegado en Render”

Evaluación de Portabilidad:

Para desarrollo del back-end del proyecto, a partir de la tecnología seleccionada, se reconoció una de las practicas más recomendadas para el manejo de dependencias y “encapsulamiento” del proyecto propias de las herramientas desarrolladas en Python la cual es la creación de un entorno virtual con venv. Venv permite gestionar las dependencias de manera aislada, asegurando que cada proyecto mantenga sus versiones específicas de paquetes sin interferencias. Esto es particularmente útil en equipos con recursos limitados, ya que proporciona un entorno ligero y eficiente, permitiendo ejecutar proyectos sin afectar al sistema operativo principal. De esta forma solo se optaría por manejarlas credenciales de alguna base datos propia del dispositivo o en la nube.

Otra alternativa, la cual es la recomendada para futuras propuestas de despliegue en entornos reales de producción, es el uso de Docker Compose. La mismo permitió la creación de tres contenedores: uno para pgAdmin4, facilitando operaciones de lectura y gestión de datos; otro para la base de datos PostgreSQL; y un tercero para la API REST, utilizando Gunicorn, un servidor de aplicaciones Python que maneja múltiples solicitudes simultáneamente, mejorando el rendimiento y la capacidad de respuesta de la API.

Docker Compose ofrece importantes ventajas, como la portabilidad y la consistencia del entorno de desarrollo y producción, garantizando que todos los componentes funcionen de manera uniforme en cualquier sistema compatible. Encapsula todas las configuraciones y dependencias necesarias, lo que simplifica considerablemente el proceso de despliegue en servidores, permitiendo que se mantenga una configuración estable y reproducible.

Se ha documentado en el proyecto la instalación tanto con venv como con Docker, ya que ambas opciones son viables(Sandoval, 2024): venv es recomendable en situaciones donde los recursos son limitados, proporcionando un entorno limpio y controlado, mientras que Docker proporciona una solución robusta y encapsulada, ideal para implementaciones en servidores. Esta dualidad en las opciones de instalación permite una mayor flexibilidad y adaptabilidad según las necesidades específicas del entorno de trabajo.

Arquitectura y configuración del proyecto:

La elección de la arquitectura adecuada es fundamental para el éxito de cualquier proyecto de desarrollo de software. En el contexto de la creación de una API REST para la aplicación móvil de la UCSM, fue crucial seleccionar las mejores prácticas y herramientas que faciliten el desarrollo, la implementación y el mantenimiento del sistema.

La elección de la arquitectura REST para el diseño del back-end de la aplicación móvil de autogestión estudiantil para la UCSM se basó en varios factores clave. En primer lugar, la arquitectura REST (Representational State Transfer) es altamente compatible con el modelo cliente-servidor utilizado comúnmente en aplicaciones web y móviles, lo que la hace ideal para la comunicación entre la aplicación móvil y el servidor. Además, REST se basa en estándares web abiertos y ampliamente adoptados, como HTTP, lo que facilita la interoperabilidad y la integración con otras tecnologías y servicios web.

Otro factor determinante fue la naturaleza escalable y flexible de la arquitectura REST, que permite agregar nuevas funcionalidades y recursos a la API de manera modular y sin afectar las partes existentes del sistema. Esto es especialmente importante en un entorno académico en constante evolución, donde las necesidades y requisitos del usuario pueden cambiar con el tiempo.

Además, la arquitectura REST fomenta la separación clara entre el cliente y el servidor, lo que facilita la gestión y la escalabilidad del sistema a medida que crece en tamaño y complejidad. Al utilizar los principios RESTful, como la identificación de recursos a través de URIs (Uniform Resource Identifiers) y el uso de métodos HTTP estándar para realizar operaciones CRUD (Create, Read, Update, Delete) en esos recursos, se establece una interfaz clara y coherente para interactuar con la API.

Para garantizar un desarrollo ordenado y eficiente del back-end de la API REST en Django REST Framework (DRF), fue esencial seguir las mejores prácticas de estructuración del proyecto. Estas prácticas se centran en la modularidad, la claridad y la coherencia en la organización del código. Dividir el proyecto en aplicaciones más pequeñas y especializadas, cada una con su propio propósito claro y funcionalidad específica, facilita la gestión y la escalabilidad del sistema. Además, el uso de serializadores de DRF para convertir los objetos de Python en representaciones JSON y viceversa ayuda a mantener la consistencia en la estructura de los datos y facilita la manipulación de la información en la API. Las vistas basadas en clases (CBV) proporcionan un enfoque más estructurado y reutilizable para definir la lógica de la API, mientras que el enrutador de DRF permite establecer un enrutamiento claro y coherente. La implementación adecuada de la autenticación y la autorización utilizando las herramientas proporcionadas por DRF garantiza la seguridad de la API y protege los datos del usuario.

Pero para manejar de manera apropiada el desarrollo del proyecto, y que el mismo pueda ser trabajado sin importar el dispositivo en el que este sea ejecutado, se hará uso de una herramienta de empaquetamiento y distribución de aplicaciones denominada Docker. Para ello, se define un archivo de Docker (Dockerfile) el cual contiene los pasos necesarios para construir la imagen del contenedor, incluyendo la instalación de las dependencias del proyecto y la configuración del entorno de ejecución. La comunicación e integración con la base de datos será abordada con mayor profundidad en próximos apartados.

Como se mencionó, para la creación de archivo de Docker, era necesario el poder instalar las dependencias utilizadas, y una práctica común para gestionarla en proyectos de Python es utilizar el archivo de texto “requirements”. En el mismo se detallan las dependencias del proyecto y sus versiones correspondientes. Lo útil del mismo, es la posibilidad también de utilizarlo en entornos virtuales de Python como venv si opta por emplear una alternativa de Docker.

Ya para la gestión de variables de entornos, las cuales son esenciales para la configuración y la seguridad de una aplicación Django, se empleó la herramienta “python-dotenv” la cual facilita esta gestión al cargar las variables desde un archivo “.env” en el entorno de desarrollo situado en el directorio raíz del

proyecto como claves secretas, credenciales de base de datos y configuraciones específicas del entorno. Dentro del archivo de configuración de Django (settings.py), se utiliza python-dotenv para cargar las variables de entorno definidas en el archivo .env, asegurando así que la configuración del proyecto sea coherente en diferentes entornos.

Para mejorar la seguridad y permitir el acceso controlado a la API desde diferentes orígenes, se implementó el paquete django-cors-headers. Esta herramienta facilita la configuración de CORS (Cross-Origin Resource Sharing), que es esencial para aplicaciones web que interactúan con servicios alojados en distintos dominios. Configurando django-cors-headers, se permitió que la aplicación móvil pudiera comunicarse de manera segura y eficiente con el servidor, especificando los orígenes permitidos y los métodos HTTP aceptados.

Adicionalmente, se implementó la expiración de tokens de autenticación para mejorar la seguridad. Esta práctica garantiza que las sesiones de usuario no autorizadas se limiten temporalmente, reduciendo la ventana de oportunidad para accesos no deseados. La configuración de expiración de tokens ayuda a proteger la API y los datos del usuario, al requerir renovaciones periódicas de autenticación.

Por último, se añadió la funcionalidad de paginación de la información. La paginación es crucial en la gestión de grandes conjuntos de datos, ya que mejora el rendimiento y la usabilidad del sistema. Al limitar la cantidad de datos devueltos en una sola respuesta, se reducen los tiempos de carga y se mejora la experiencia del usuario. Además, facilita la navegación y la visualización ordenada de la información, permitiendo a los usuarios acceder de manera eficiente a los datos relevantes sin sobrecargar el sistema.

En la configuración del proyecto Django, se definieron los ALLOWED_HOSTS para restringir las solicitudes HTTP a orígenes específicos. Solo se incluyeron los dominios correspondientes al entorno de producción en Render y el entorno local de desarrollo. Esta restricción es fundamental para proteger el servidor de posibles ataques y accesos no autorizados, asegurando que solo las solicitudes provenientes de las fuentes legítimas sean procesadas.

Además, se incorporó drf-spectacular para la generación automática de documentación de la API utilizando Swagger. Este enfoque facilita la visualización y comprensión de los endpoints disponibles, proporcionando una interfaz interactiva para los desarrolladores y usuarios interesados en explorar las funcionalidades de la API. La implementación de Swagger se detalla en el apartado de "Definición de endpoints", donde se explican las características y beneficios de esta herramienta en el desarrollo y mantenimiento de la API.

Definición de Endpoints:

En la API REST desarrollada con Django Rest Framework (DRF), se han definido varios endpoints para manejar diversas funcionalidades de la aplicación móvil de autogestión estudiantil. A continuación, se detallan los principales endpoints y sus funciones:

Inicio de Sesión:

Este endpoint permite a los usuarios iniciar sesión en el sistema. Se accede mediante una petición POST a api/login y requiere un cuerpo de solicitud que incluya el documento de identificación y la contraseña del usuario.

Debt:

El endpoint GET en api/debts/get-debt/?enrollment_id=<uuid:id> permite recuperar la deuda de un curso específico y sus cuotas asociadas. Esto facilita a los estudiantes consultar su situación financiera relacionada con un curso particular. Para ello es necesario tanto disponer de un token de sesión compartido por el header de petición como el id del enrollment o curso solicitado.

Enrollment:

Se han habilitado dos endpoints para manejar la información de inscripción de los estudiantes. El primer endpoint GET en `api/enrollments/?identification_document=<string:dni>` devuelve un listado de todos los cursos en los que un estudiante está inscrito. El segundo GET en `api/enrollments/<string:id>` permite recuperar información detallada de un curso específico.

Event:

Para la gestión de eventos, se dispone de dos endpoints. El GET en `api/events` proporciona un listado completo de eventos, mientras que el GET en `api/events/<string:id>` permite acceder a la información de un evento en particular.

Semester:

En la categoría de semestres, el GET en `api/semesters` devuelve una lista de todos los semestres disponibles. Además, el GET en `api/semesters/<string:id>` permite recuperar la información de un semestre específico.

Student:

El endpoint GET en `api/students/<string:id>` permite obtener el perfil completo de un estudiante, facilitando el acceso a sus datos personales y académicos.

Subject Registration:

Este módulo cuenta con varios endpoints GET para manejar la información de las materias en las que los estudiantes están registrados. Se puede acceder a las materias inscritas a través de `api/subject-registrations/?enrollment_id=<uuid:id>`, a una materia en particular con `api/subject-registrations/<string:id>`, a las materias restantes por cursar mediante `api/subject-registrations/get-remaining-subjects/?enrollment_id=<uuid:id>`, y a estadísticas de materias aprobadas y restantes con `api/subject-registrations/get-statistics/?enrollment_id=<uuid:id>`.

La mayoría de estos endpoints están diseñados para operaciones de lectura, utilizando peticiones GET que recuperan datos específicos del sistema. Las operaciones de modificación de datos más sensibles se gestionarán a través del administrador de Django, lo cual se explica con más detalle en el apartado de desarrollo de la aplicación. Esta organización permite mantener la integridad y seguridad de la información manejada por el sistema. A su vez, como se mencionaba en secciones anteriores, se añadió la configuración de `'drf-spectacular'` para el uso de Swagger. Esto permite disponer de un endpoint en `'api/docs'`, donde se pueden visualizar los esquemas que devuelve la API, los tipos de errores posibles y realizar pruebas directamente con la aplicación. Esta funcionalidad facilita la comprensión y la interacción con la API por parte de los desarrolladores. Se puede visualizar dicha interfaz en la siguiente ilustración:

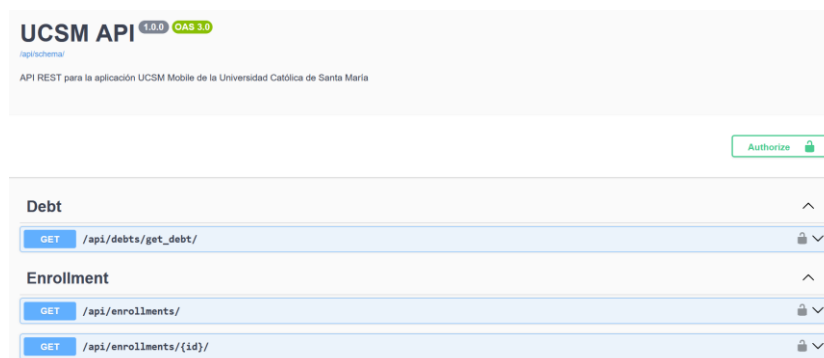


Ilustración 11 “Interfaz de Swagger de API REST”

Base de datos:

En el presente apartado se busca detallar la elección del sistema de gestión de bases de datos (SGBD) como el desarrollo del mismo incluyendo toda consideración necesaria para la elaboración del mismo. La elección de PostgreSQL como SGBD fue a partir de contemplar una serie de ventajas y consideraciones importantes; en una primera instancia, PostgreSQL es conocido por su robustez, confiabilidad y capacidad para manejar grandes volúmenes de datos. Esto lo convierte en una opción ideal para un sistema que requiere un almacenamiento seguro y eficiente de datos, como el sistema de autogestión de alumnos de la UCSM. Además, el equipo de trabajo del ERP de la misma institución ya tiene experiencia previa con PostgreSQL, ya que varios proyectos del ERP fueron desarrollados con este SGBD. Esta familiaridad con la tecnología agiliza el proceso de desarrollo y reduce el tiempo de aprendizaje, lo que resulta en una mayor eficiencia y calidad del trabajo realizado.

A partir de esto, se optó por utilizar la plantilla oficial de Docker para PostgreSQL para la creación de la base de datos. Esta decisión se realizó porque la misma plantilla proporciona una base estable y confiable para el entorno de la base de datos. Estas plantillas son mantenidas por expertos en Docker y PostgreSQL, lo que garantiza su calidad y seguridad. Al aprovechar una plantilla oficial, se reduce el riesgo de errores de configuración y se garantiza una configuración optimizada y bien probada para el despliegue de PostgreSQL en un contenedor Docker. Además, la plantilla incluye una configuración predeterminada que abarca aspectos importantes como la autenticación, la configuración de la red y la gestión de recursos, lo que permite a los desarrolladores desplegar fácilmente una instancia de PostgreSQL lista para usar en un entorno de contenedor Docker.

Otra ventaja para contemplar fue la portabilidad y la consistencia del entorno de base de datos. Al utilizar una imagen de contenedor Docker, se encapsula la configuración y los datos de PostgreSQL en un entorno aislado y autocontenido. Esto facilita la replicación del entorno de desarrollo en múltiples sistemas y asegura que todos los miembros del equipo trabajen en el mismo entorno consistente, lo que reduce los errores y aumenta la eficiencia del desarrollo.

Ya para el diseño de la misma base de datos que serviría como recurso para aplicación y que el mismo garantice una estructura clara y coherente, fue necesario realizar un Diagrama Entidad-Relación (DER). Este DER sirve como una representación visual de la estructura de la base de datos, mostrando las tablas, los campos y las relaciones entre ellas. Al desarrollar el DER, se buscó simplificar y organizar la estructura de la base de datos de manera que fuera fácil de entender y seguir al crear los modelos en Django. Este mismo está disponible en el Anexo 1 como una ilustración completa, distinguiendo o clasificando las diversas tablas con una serie de colores distintivos: Azules, como tablas primarias (representan entidades principales en el sistema), Amarillas las tablas secundarias simples con una relación de clave foránea (relacionadas de forma directa con las tablas primarias y contienen información adicional o complementaria sobre las entidades principales) y Rojas a las tablas secundarias con más de una clave foránea en relación (tablas que son más complejas y pueden representar relaciones muchos a muchos o relaciones donde una entidad está vinculada a múltiples entidades principales).

En cuanto a la creación de las tablas y relaciones en la base de datos, se optó por utilizar el ORM (Object-Relational Mapping) de Django junto con la dependencia `psycopg2`. La inclusión de la dependencia `psycopg2` en el proyecto tiene una relación directa con el uso de PostgreSQL como sistema de gestión de bases de datos (SGBD; este es un adaptador de base de datos PostgreSQL para Python que permite la comunicación entre la aplicación Django y la base de datos PostgreSQL. Al utilizarlo, Django puede interactuar de manera eficiente con la base de datos PostgreSQL, ejecutando consultas, recuperando y almacenando datos de manera segura y confiable.

Para optimizar el desarrollo y asegurar la coherencia entre el modelo de datos y la estructura de la base de datos, se empleó el ORM (Object-Relational Mapping) de Django en conjunto con `psycopg2`. Esta abstracción facilitó la interacción con la base de datos al permitir definir los modelos de datos en Python y luego generar automáticamente las tablas y relaciones correspondientes en PostgreSQL.

En relación a los modelos que se pueden crear con las herramientas mencionadas, cabe destacar que en el contexto de Django se pudo aprovechar las migraciones automáticas que ofrece este framework para gestionar la evolución de la estructura de la base de datos a lo largo del tiempo. Las migraciones de Django permiten realizar cambios en el modelo de datos de manera incremental y coherente, sin necesidad de escribir consultas SQL manualmente ni preocuparse por la integridad de la base de datos e incluso permitiendo hacer cancelar los cambios en caso de tener que volver a una migración anterior.

Además, se decidió utilizar UUID (Universally Unique Identifier) en lugar de Big Int para los IDs de los modelos. Los UUID son identificadores únicos generados de manera aleatoria que tienen una probabilidad extremadamente baja de colisión.

En cuanto al almacenamiento seguro de contraseñas, se utilizará el sistema de hash de contraseñas proporcionado por Django. Este sistema utiliza algoritmos de hash seguros y técnicas de salting para proteger las contraseñas de los usuarios contra ataques de fuerza bruta y otras vulnerabilidades de seguridad.

Finalmente, el uso del panel de administración de Django proporciona una ventaja significativa al facilitar las pruebas e integración con la aplicación Angular Ionic. El panel de administración de Django ofrece una interfaz intuitiva y fácil de usar para administrar y visualizar los datos en la base de datos, lo que facilita la verificación y validación de los datos durante el desarrollo y las pruebas de integración con la aplicación móvil. Esto garantiza que la integración entre el back-end y el front-end se realice de manera fluida y eficiente, facilitando el proceso de desarrollo y asegurando la calidad del producto final.

En el proceso de desarrollo y gestión de la base de datos, se utilizó pgAdmin, una herramienta de administración para PostgreSQL. Esta aplicación facilita la administración de la base de datos a través de una interfaz gráfica, permitiendo a los desarrolladores y administradores realizar operaciones como la creación de tablas, la ejecución de consultas SQL y la gestión de usuarios y permisos de manera eficiente y organizada. pgAdmin también ofrece la posibilidad de visualizar y modificar datos, lo que resulta esencial para mantener la integridad y la coherencia de la base de datos durante todo el ciclo de desarrollo. La interfaz con el tablero se puede visualizar en la siguiente ilustración:

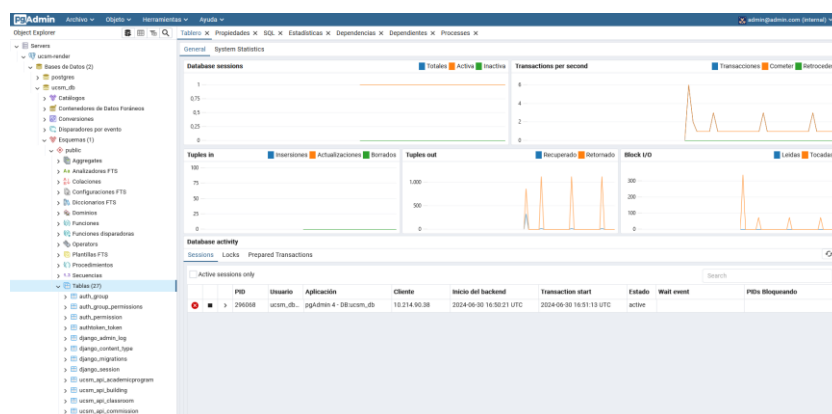


Ilustración 12 "Interfaz de pgAdmin 4"

Además, se creó una instancia de la base de datos en Render Cloud. Esta implementación permite que la base de datos sea consumida como un servicio web, emulando el funcionamiento actual con los servidores de la universidad. Utilizar Render Cloud garantiza una alta disponibilidad y escalabilidad, facilitando el acceso y la gestión remota de la base de datos, lo que resulta crucial para simular entornos de producción y realizar pruebas de integración y despliegue en tiempo real. Esta configuración también proporciona un entorno de prueba robusto para evaluar el rendimiento y la seguridad de la base de datos en condiciones similares a las de un entorno de producción.

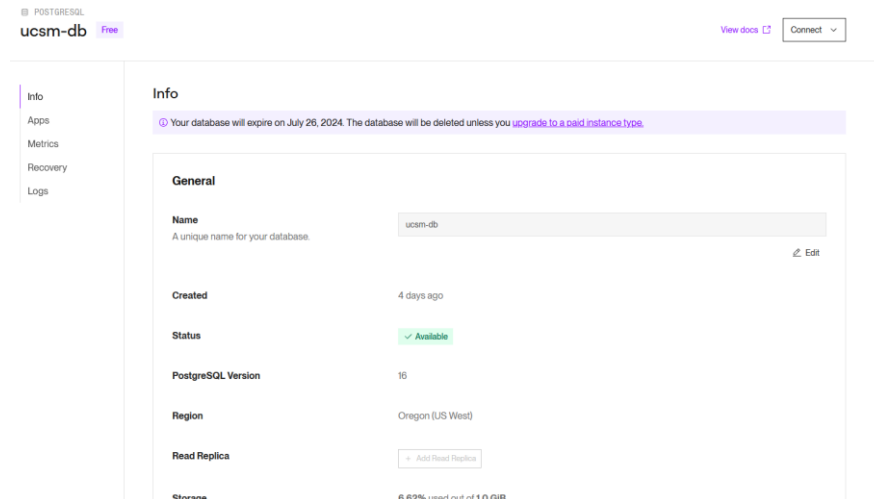


Ilustración 13 "Instancia de base de datos PostgreSQL en Render"

Configuración de Docker compose:

La configuración de Docker Compose para el back-end del proyecto de la aplicación móvil de la UCSM ofrece una solución eficiente y escalable para gestionar los servicios necesarios en el entorno de desarrollo. Docker Compose es una herramienta que permite definir y gestionar aplicaciones multi-contenedor de forma sencilla, facilitando la configuración y el despliegue de los servicios requeridos para el proyecto.

La utilización de Docker Compose ofrece diversas ventajas para el desarrollo del back-end de la aplicación móvil de la UCSM. En primer lugar, proporciona un entorno de desarrollo consistente y reproducible, lo que garantiza que todos los miembros del equipo trabajen en la misma configuración de servicios, independientemente de las diferencias en sus entornos de desarrollo locales. Además, Docker Compose simplifica la gestión de dependencias y la configuración del entorno, lo que reduce el tiempo dedicado a la configuración y soluciona posibles conflictos entre los servicios.

En cuanto a la lógica de definición de los servicios en el archivo `docker-compose.yml`, se dividen en tres servicios principales: web, postgres y pgadmin:

- El servicio web se refiere a la API desarrollada con Django, que actúa como el back-end de la aplicación móvil. En el archivo docker-compose.yml, se define este servicio mediante la especificación de la imagen Docker a utilizar, las variables de entorno necesarias y los puertos de red que se deben exponer. Además, se puede incluir la configuración de volúmenes para montar directorios locales como volúmenes en el contenedor, lo que facilita el desarrollo y la depuración de la aplicación.
- El servicio postgres se refiere a la base de datos PostgreSQL utilizada por la aplicación. Al igual que con el servicio web, en el archivo docker-compose.yml se define este servicio especificando la imagen Docker de PostgreSQL a utilizar, así como las variables de entorno necesarias para configurar la base de datos, como el nombre de usuario, la contraseña y el nombre de la base de datos. También se pueden definir volúmenes para persistir los datos de la base de datos entre ejecuciones del contenedor.
- El servicio pgadmin es una imagen de pgAdmin que se conecta a la base de datos de PostgreSQL. Este servicio permite a los desarrolladores visualizar y gestionar las tablas, facilitando la administración de la base de datos. Se configura en el archivo `docker-compose.yml` especificando la imagen de pgAdmin, las variables de entorno para la autenticación y los puertos necesarios para acceder a la interfaz web de pgAdmin.

En cuanto a los Dockerfiles asociados a cada servicio, estos contienen las instrucciones necesarias para construir las imágenes Docker correspondientes. En el Dockerfile del servicio web, se incluyen las instrucciones para instalar las dependencias del proyecto Django, copiar los archivos de código fuente al contenedor y configurar el entorno de ejecución de la aplicación. Por otro lado, en el Dockerfile del servicio postgres, se incluyen las instrucciones para configurar e inicializar la base de datos PostgreSQL, así como para exponer el puerto de red utilizado por el servicio.

Cabe destacar que, por defecto, Docker Compose crea una red de puente (bridge) interna que permite la comunicación entre los contenedores. Esta red se crea automáticamente y conecta todos los contenedores definidos en el archivo Docker Compose. En este caso, los contenedores "web", "postgres" y "pgadmin" estarán conectados a esta red interna. Todo lo mencionado se puede observar más claramente en el gráfico presente en la ilustración:

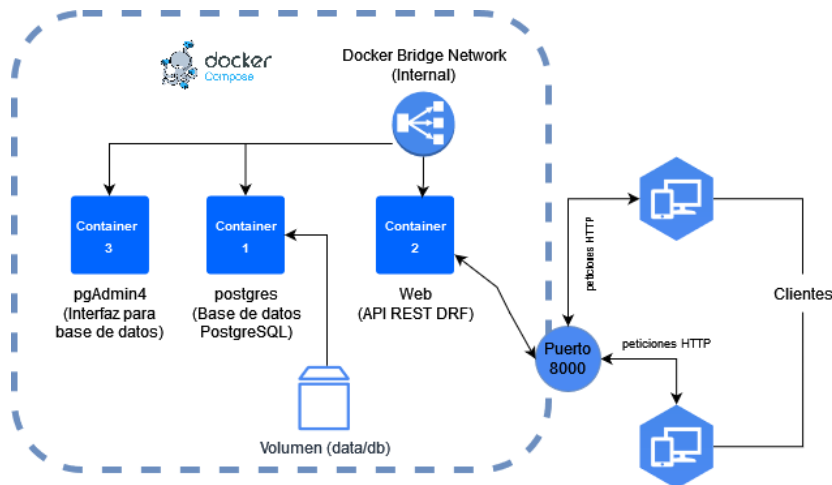


Ilustración 14: "Diagrama de la estructura de los contenedores gestionados"

Gestión de dependencias:

En este proyecto, se utilizó Python 3.11 (python:3.11.9-alpine en Docker y en la imagen de Render) como base. Para la gestión de dependencias, se optó por uno de los métodos más comunes en Python: la creación de archivos de texto plano que enumeran todos los paquetes necesarios para el proyecto, conocidos como archivos de requerimientos. Estos archivos permiten especificar los paquetes de terceros que se deben instalar desde PyPI (Python Package Index), un repositorio oficial de software para Python donde se alojan las bibliotecas y herramientas disponibles para la comunidad.

El gestor de paquetes utilizado es pip, una herramienta que facilita la instalación, actualización y gestión de paquetes desde PyPI. Pip permite automatizar el proceso de instalación de dependencias, asegurando que todas las bibliotecas necesarias estén disponibles en el entorno de desarrollo o producción.

Para una organización más eficiente, se planificó el uso de diferentes archivos de requerimientos para producción y desarrollo. Esta separación permite probar nuevas funcionalidades en un entorno de desarrollo, utilizando dependencias específicas, sin afectar el entorno de producción. Sin embargo, en la fase actual del prototipo, se utiliza un único archivo de requerimientos.

Adicionalmente, se empleó Dependabot de GitHub para el monitoreo de dependencias. Dependabot escanea regularmente los archivos de requerimientos y alerta sobre cualquier problema, como vulnerabilidades de seguridad o actualizaciones necesarias. Esto ayuda a mantener las dependencias del proyecto seguras y actualizadas, asegurando un entorno de desarrollo robusto y confiable.

Desarrollo de la aplicación:

En el desarrollo de la aplicación, se ha optado por utilizar Django Rest Framework (DRF) para construir una API REST robusta y eficiente. El proyecto se inició mediante el comando `django-admin` para crear

un proyecto y se configuró la aplicación correspondiente. Django, conocido por su modularidad al permitir la creación de múltiples aplicaciones dentro de un mismo proyecto, ofrece una estructura organizada que facilita la extensibilidad futura hacia otras áreas de la universidad.

Un componente clave del desarrollo fue la implementación del administrador de Django, gestionado a través de manage.py, donde se generaron los archivos de migración para definir y actualizar los modelos de datos en la base de datos. Este proceso incluyó la creación de un superusuario inicial para el prototipo, permitiendo un acceso completo al sistema. Para futuras trabajos en relación a este proyecto, se puede implementar un sistema de roles y permisos más específico, aprovechando las capacidades de gestión de usuarios del administrador de Django. Para el presente proyecto, se planteó el rol de los estudiantes, que tendrán permisos limitados para operaciones específicas, como la solicitud de datos y la gestión de tokens de sesión, aspectos cruciales para la seguridad y la operatividad del sistema.

En el contexto del desarrollo de la API REST, se han utilizado conceptos clave como serializers y viewsets proporcionados por DRF. Los serializers son componentes esenciales que convierten modelos complejos de Django en representaciones JSON para su intercambio entre la API y el cliente. Por otro lado, los viewsets son clases que agrupan la lógica de las vistas y manejan las solicitudes HTTP para diferentes endpoints de la API. Utilizando ReadOnlyModelViewSet, se optimiza el código al automatizar operaciones comunes como la recuperación de datos, siguiendo el principio de DRY (Don't Repeat Yourself).

Aunque DRF incluye una interfaz de prueba para endpoints, se integró DRF Spectacular para mejorar la documentación y facilitar la visualización de la API. Esta herramienta no solo ofrece una representación clara de los endpoints disponibles y sus parámetros, sino que también mejora la gestión de tokens de sesión, proporcionando una capa adicional de seguridad y facilitando el proceso de desarrollo y depuración. Integrar estas tecnologías no solo mejora la eficiencia del desarrollo, sino que también asegura una API bien documentada y fácil de mantener.

En cuanto a la gestión de tokens de sesión, se emplearon las funcionalidades integradas de Django Rest Framework (DRF) para manejar la autenticación. Al iniciar sesión a través del endpoint api/login, se genera un token de sesión que se asigna al usuario y que tiene una duración predeterminada de 24 horas. Este período de tiempo está diseñado para equilibrar la seguridad y la conveniencia del usuario final. Una vez que el token expira, se requiere que el usuario inicie sesión nuevamente para obtener un nuevo token válido, lo cual refuerza la seguridad al garantizar que los accesos sean temporales y controlados. Esta estrategia no solo asegura la protección de los datos y recursos del sistema, sino que también cumple con los estándares de seguridad recomendados para aplicaciones web modernas. Esto se puede observar en la siguiente ilustración:

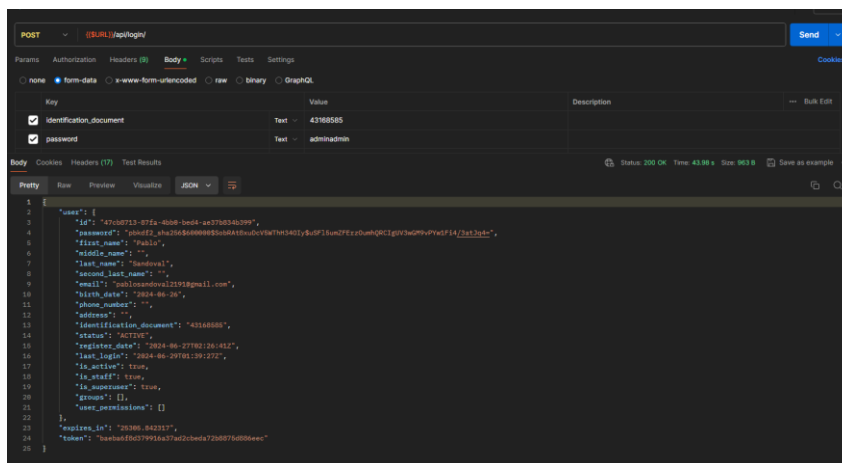


Ilustración 15 "Prueba de endpoint login con Postman"

Seguridad:

“Por naturaleza, las API exponen la lógica de la aplicación y los datos confidenciales, como la información de identificación personal, y debido a esto se han convertido cada vez más en un objetivo para los atacantes. Sin APIs seguras, la innovación rápida sería imposible” esto denota OWASP previo a mencionar las distintas estrategias y soluciones propuestas para comprender y mitigar algunas de las vulnerabilidades y riesgos que pueden tener las APIs; a partir de las mismas se ira proponiendo que enfoque se aplicó en la presente API REST en Django desarrollada. Cabe destacar que cada uno de los ítems se identifica con el título API-identificador, en lo que se reconoce cada riesgo posible en cada uno de los puestos del Top 10 de OWASP API REST.

API1: Autorización de nivel de objeto roto

Este riesgo se abordó en Django mediante la implementación de decoradores personalizados que verificaban la autorización a nivel de objeto. En cada función que accedía a datos, se realizaron comprobaciones para asegurarse de que el usuario tuviera los permisos adecuados. Esto se logró definiendo permisos personalizados y verificando la propiedad del objeto antes de permitir el acceso.

API2: Autenticación rota

Para solucionar este problema, se optó por utilizar bibliotecas de autenticación segura, como Django REST Framework en Django. Además, se siguieron las mejores prácticas para garantizar el almacenamiento seguro de contraseñas y la transmisión segura de datos de autenticación. Se evitó implementar una solución de autenticación personalizada y se confió en soluciones ampliamente aceptadas.

API3: Autorización de nivel de propiedad de objetos roto

En Django, se abordó este riesgo implementando verificaciones de autorización en vistas y funciones que accedían a objetos. Se utilizaron decoradores y permisos personalizados para asegurarse de que los usuarios solo tuvieran acceso a los objetos que les pertenecían.

API4: Consumo de recursos sin restricciones

Para evitar un consumo excesivo de recursos, se aplicaron restricciones en las solicitudes utilizando Django REST Framework. Esto incluyó la configuración de límites de velocidad y cuotas de solicitudes por usuario o dirección IP para garantizar un uso controlado de los recursos y prevenir la posibilidad de una denegación de servicio.

API5: Autorización de nivel de función rota

Se resolvió este riesgo en Django definiendo políticas de control de acceso de manera clara y precisa. Se implementaron permisos personalizados y decoradores que reflejaban las jerarquías, grupos y roles de manera adecuada para garantizar que los atacantes no pudieran acceder a funciones administrativas de otros usuarios.

API6: Acceso sin restricciones a flujos comerciales confidenciales

Se realizó una evaluación exhaustiva de los flujos comerciales en la aplicación y se aseguró de que las API no expusieran funcionalidades críticas sin restricciones. Se definieron y aplicaron políticas de autorización estrictas para proteger las operaciones comerciales críticas.

API7: Falsificación de solicitud del lado del servidor (SSRF)

Para mitigar el SSRF (fallas de falsificación de solicitud del lado del servidor) en Django, se llevaron a cabo estrictas validaciones y restricciones en los URI proporcionados por los usuarios en las solicitudes a recursos remotos. Se mantuvo una lista blanca de destinos permitidos empleando django-cors-headers como se mencionó en apartados anteriores.

API8: Configuración incorrecta de seguridad

Se siguieron las mejores prácticas de seguridad al configurar la aplicación en Django. Esto implicó la revisión y validación continua de la configuración para evitar fallos de seguridad como guardar las variables de entorno por separado, técnicas de hashing para las contraseñas y permisos limitados. Se aplicaron configuraciones seguras en el servidor web, las bases de datos y otros componentes.

API9: Gestión de inventario inadecuada

Para abordar este riesgo, se mantuvo una documentación detallada de la API y un inventario actualizado de las versiones implementadas en Django. Esto contribuyó a mitigar problemas como versiones obsoletas de la API y la exposición de puntos finales de depuración. Se estableció un proceso de gestión de versiones sólido y se mantuvo la documentación al día.

API10: Consumo no seguro de API

Se implementó una validación exhaustiva de los datos recibidos de las API de terceros en Django. Se realizaron verificaciones de datos y se aplicaron filtros para identificar y descartar cualquier entrada sospechosa o no confiable por medio de limitar que dispositivos pueden consumir el servicio con la lista `CORS_ALLOWED_ORIGINS` (la cual podría cambiar cuando la aplicación este disponibles en tiendas de aplicaciones de Android o IOS). Además, se establecieron políticas de seguridad rigurosas al integrar servicios de terceros para garantizar que no comprometieran la seguridad de la API.

Configuración de entorno de prueba:

La configuración del entorno para el proyecto conlleva la consideración de diversas restricciones que influyen en la forma en que se lleva a cabo la prueba del proyecto en su totalidad. Debido a la falta de acceso a un servidor institucional para el desarrollo y prueba del proyecto, se debieron buscar alternativas que permitieran realizar pruebas de manera efectiva.

Ante esta situación, se optó por trabajar con opciones serverless para llevar a cabo las pruebas del proyecto. Los servicios serverless son una forma de implementar aplicaciones sin la necesidad de gestionar la infraestructura subyacente. En lugar de preocuparse por la administración de servidores y recursos, los desarrolladores pueden centrarse en escribir código y crear aplicaciones escalables y de alto rendimiento. Una de las principales ventajas de los servicios serverless es su capacidad para escalar automáticamente según la demanda, lo que permite manejar cargas de trabajo variables de manera eficiente.

Dada esta elección, se decidió desplegar el servicio en partes. Para la instancia de la base de datos PostgreSQL, se optó por utilizar la versión gratuita de Supabase. Supabase es una plataforma de base de datos que ofrece una base de datos PostgreSQL completamente administrada, junto con una variedad de funciones adicionales, como autenticación de usuarios, almacenamiento de archivos y API REST. La versión gratuita de Supabase proporciona acceso limitado pero suficiente para propósitos de prueba y desarrollo.

Por otro lado, para el servicio de Django, se eligió utilizar Render, un servicio de hosting gratuito que ofrece una plataforma para implementar aplicaciones web y APIs de manera sencilla. Utilizando Render, fue posible alojar el servicio de Django y exponerlo al público de manera segura y confiable. Para vincular el servicio de Django con la base de datos PostgreSQL de Supabase, se definieron las variables de entorno con las claves de acceso necesarias, lo que permitió establecer la conexión entre ambos servicios de manera transparente. La comunicación resultante entre los servicios la aplicación compilada como .apk para celulares Android utilizando Capacitor se puede observar en la siguiente ilustración:



Ilustración 16: “Comunicación de los servicios para las pruebas de Integración”

Además de las pruebas de funcionamiento del proyecto, también se llevó a cabo un proceso de pruebas de usabilidad para evaluar la experiencia del usuario. Para este propósito, se utilizó el servicio actual desarrollado en PHP, integrándolo con la nueva interfaz desarrollada para la aplicación móvil de la UCSM. Esta integración permitió a los usuarios probar la nueva interfaz con datos reales y compararla con la versión anterior. Al proporcionar a los usuarios la oportunidad de interactuar con la aplicación utilizando datos reales y escenarios de uso cotidianos, fue posible obtener una retroalimentación más precisa y detallada sobre la experiencia de usuario. Esta retroalimentación fue invaluable para identificar áreas de mejora y realizar ajustes en la interfaz para garantizar una experiencia óptima para los usuarios finales.

Evaluación de Aplicaciones:

A partir del prototipo desarrollado y considerando las restricciones mencionadas en este proyecto, en este apartado se realizará una comparación con el producto existente hasta el momento.

En cuanto a la aplicación móvil, se destacan varios puntos positivos en comparación con la actual PWA. El código fue desarrollado siguiendo prácticas de programación que facilitan la incorporación de cambios o nuevas funcionalidades, implementando herramientas de corrección de código para garantizar su legibilidad y mantenimiento. Además, la aplicación obtuvo una alta tasa de aceptación en la evaluación de usabilidad, lo que sugiere un buen diseño de UX/UI. Aunque aún no se ha implementado, la capacidad de integrar funcionalidades nativas de los dispositivos móviles en los que se trabaja añade un valor significativo, permitiendo una experiencia más rica y personalizada para los usuarios.

En relación con el back-end, el uso del estilo de arquitectura REST, en conjunto con el framework Django REST Framework (DRF), ha facilitado la creación de un servicio con una documentación clara, utilizando herramientas como Drf-Spectacular. Esto permite identificar qué tipos de endpoints son los más apropiados para su uso, a diferencia del servicio desarrollado en PHP, que solo maneja operaciones POST sin diferenciar el tipo de operación. Aunque se creó una base de datos propia sin seguir la arquitectura actual de la universidad, se logró una estructura ordenada que facilita las consultas y la recuperación de información adecuada. Además, al implementar estrategias de hosts permitidos y CORS, se limita el acceso a la API solo a ciertos dispositivos, lo cual es crucial para evitar posibles ataques, como se plantea en la sección de seguridad.

Finalmente, el uso de Docker Compose en el back-end asegura que el servicio pueda ser transferido y desplegado en cualquier dispositivo, como se abordó en el apartado de portabilidad. Esto garantiza una mayor flexibilidad y consistencia en el desarrollo y despliegue del proyecto, lo que es esencial en entornos de desarrollo colaborativos y en constante evolución.

Evaluación de la Calidad del Software:

A partir del desarrollo llevado a cabo, y siguiendo lo que estipula la normativa ISO de Calidad de Software ISO 25010, en el presente apartado se irá evaluando si el mismo contempla las 8 características principales de calidad:

- **Idoneidad funcional:** Esta característica contempla el grado o nivel en que el software resultante proporciona funciones que satisfacen las necesidades establecidas para determinadas condiciones. A partir de lo que se plantea, se puede concluir que se arribó a los resultados, gracias a la validación por parte del equipo oficial de desarrollo de la Universidad Católica de Santa María que llevo a cabo el desarrollo del ERP de la misma institución. Para el desarrollo de este software, los requerimientos y las tecnologías han sido planteados por el mismo equipo y el producto resultante fue validado por los mismos marcando la idoneidad del mismo.
- **Fiabilidad:** Esta característica representa el grado en que un sistema realiza sus funciones bajo determinadas condiciones y período de tiempo. En el presente proyecto, es difícil medir los resultados, ya que, en última instancia, la infraestructura marca la diferencia. En el prototipo actual, el servicio está alojado en una plataforma serverless llamada Render, en un plan "individual" que es gratuito. Este plan se desactiva por inactividad después de un cierto tiempo sin recibir peticiones HTTP, lo que puede causar demoras en la respuesta. Sin embargo, al trasladarse a los servidores de la propia universidad, estos aspectos podrían mejorar. A su vez, se procurará, realizando casos de prueba, garantizar que los tiempos de respuesta sean óptimos sin caer por algún error no contemplado.
- **Eficiencia en el desempeño:** El rendimiento en relación con la cantidad de recursos utilizados en las condiciones establecidas. Una forma de contemplar esto fue aprovechando el almacenamiento local que ofrece el framework Angular, lo cual sirve para mejorar el rendimiento y la experiencia del usuario, especialmente en condiciones de red limitada. En el caso de la API REST, al estar en gran parte limitada a operaciones de lectura, no se llevan a cabo consultas complejas que ponga en peligro el desempeño del mismo servicio.
- **Usabilidad:** El grado o nivel en que el software permite ser entendido, aprendido, usado y fácil para el usuario. Esto se puede medir desde la aplicación móvil y fue llevado a cabo una evaluación de usabilidad con un pequeño grupo de muestra conformado por alumnos de la Universidad Católica de Santa María, quienes serían los usuarios finales de aplicación. Los resultados obtenidos se pueden observar en detalle en el capítulo de “Resultados” del presente proyecto.
- **Seguridad:** El grado de protección de la información y los datos para que las personas o los sistemas no autorizados no puedan leerlos o modificarlos y que permita el acceso y uso a las personas o sistemas autorizados. Se planteó para conocer y mitigar los casos de riesgos más comunes, seguir lo estipulado por la OWASP TOP 10 tanto para API REST como para Mobile.
- **Compatibilidad:** El grado en que dos o más sistemas o componentes pueden intercambiar información y/o realizar sus funciones. En el caso del presente proyecto que se desarrolló 2 aplicaciones (una aplicación móvil y un servicio API REST) se contempla la compatibilidad en cuanto a que se reciba los datos correctamente desde la API en la aplicación móvil. Esto implica que ambos usan los mismos formatos de datos (esto es posible gracias a los serializers de DRF que permite pasar convertir los modelos a formato JSON que es el formato convención en las API REST y el más manejable para la app la cual fue desarrollado en un framework de Typescript, lenguaje que maneja ese tipo de objetos) y protocolos (empleando HTTPS para

peticiones seguras entre el servicio y la app), y que la misma puede interpretar y utilizar las respuestas de la API sin problemas.

- **Mantenibilidad:** El grado de eficacia y eficiencia con que el software puede ser modificado y/o actualizado. Es decir, qué tan fácil y rápido se pueden hacer cambios en el software sin causar problemas. Gracias a que tanto la aplicación móvil y la servicio web fueron desarrollados siguiendo la estrategia de git flow con la herramienta de control de versión Git, se pudo ir añadiendo los cambios sin causar conflictos con otros componentes del mismo proyecto. A su vez se siguió las buenas prácticas que se marcan tanto en el framework Django como en el de Angular, siguiendo una arquitectura de componentes modularizados, lo cual garantiza que se pueden ir integrando los componentes con la certeza de que, si uno fallase, solo causaría problemas en dicho modulo, sin afectar al resto. Haciendo fácil de detectar la falla.
- **Transferibilidad:** El grado o nivel en que un sistema o componente transferido de un hardware, software u otro entorno operativo a otro. En este proyecto, este aspecto fue considerado al utilizar Docker Compose tanto para el back-end como para la base de datos. Docker Compose asegura que estas tecnologías puedan funcionar de manera consistente e independiente del contexto en el que se ejecuten, facilitando la portabilidad del sistema entre diferentes ambientes. Para la aplicación móvil, se empleó PNPM, una herramienta más potente que npm, especialmente útil para gestionar problemas de versiones de los paquetes utilizados. Esto garantiza que la aplicación pueda ser desarrollada y desplegada de manera efectiva en distintas plataformas y configuraciones, manteniendo la coherencia y funcionalidad esperadas.

Planificación del proyecto:

La organización del proyecto se visualizará mediante un Diagrama de Gantt, el cual exhibirá la secuencia de las actividades requeridas para ejecutar el proyecto y el tiempo estimado para cada una de ellas. A continuación, se presentará el diagrama de Gantt general del proyecto, seguido de una descripción detallada de cada actividad.

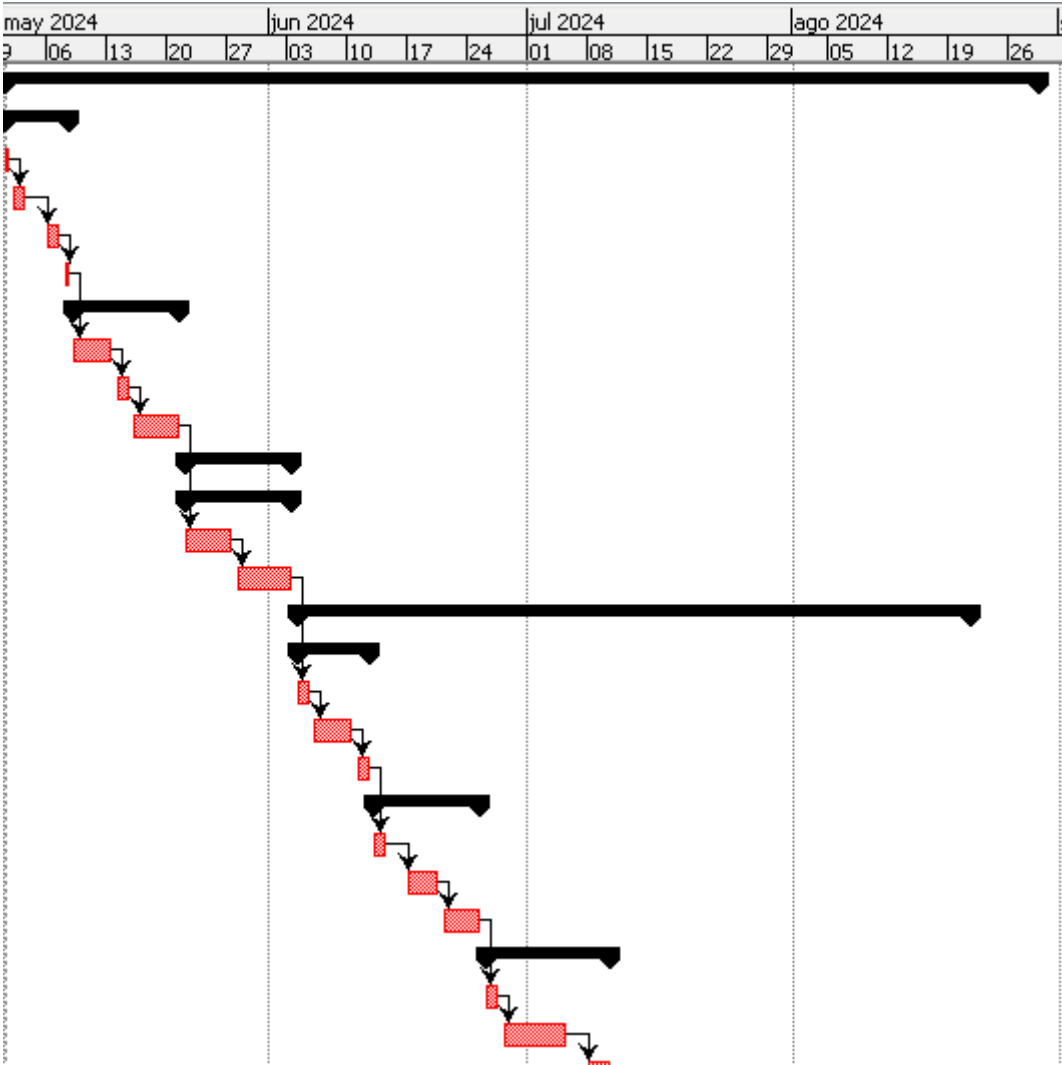


Ilustración 17: “diagrama de Gantt del proyecto”

☐ Aplicación móvil y servicio web para la autogestión	87 days	01/05/24 08:00	29/08/24 17:00	
☐ Analisis e Inicio del Proyecto	6 days	01/05/24 08:00	08/05/24 17:00	
Identificación del Problema	1 day	01/05/24 08:00	01/05/24 17:00	
Definición del Alcance	2 days	02/05/24 08:00	03/05/24 17:00	3
Definición del Objetivo General y Especifico	2 days	06/05/24 08:00	07/05/24 17:00	4
Inicio del Proyecto	1 day	08/05/24 08:00	08/05/24 17:00	5
☐ Definición y Planificación del Proyecto	9 days	09/05/24 08:00	21/05/24 17:00	
Analisis de Requerimientos	3 days	09/05/24 08:00	13/05/24 17:00	6
Analisis de Factibilidad	2 days	14/05/24 08:00	15/05/24 17:00	8
Analisis de Costos - Beneficios	4 days	16/05/24 08:00	21/05/24 17:00	9
☐ Investigación	9 days	22/05/24 08:00	03/06/24 17:00	
☐ Estudio de Seguridad	9 days	22/05/24 08:00	03/06/24 17:00	
Analisis de riesgos Owasp top 10 Mobile	4 days	22/05/24 08:00	27/05/24 17:00	10
Analisis de riesgos Owasp top 10 APIREST	5 days	28/05/24 08:00	03/06/24 17:00	13

Ilustración 18: “diagrama de Gantt del proyecto – análisis, planificación e investigación”

Desarrollo del Software	57 days	04/06/24 08:00	21/08/24 17:00	
Incremento 1	7 days	04/06/24 08:00	12/06/24 17:00	
Analisis y Diseño	2 days	04/06/24 08:00	05/06/24 17:00	14
Desarrollo	3 days	06/06/24 08:00	10/06/24 17:00	17
Testing	2 days	11/06/24 08:00	12/06/24 17:00	18
Incremento 2	9 days	13/06/24 08:00	25/06/24 17:00	
Analisis y Diseño	2 days	13/06/24 08:00	14/06/24 17:00	19
Desarrollo	4 days	17/06/24 08:00	20/06/24 17:00	21
Testing	3 days	21/06/24 08:00	25/06/24 17:00	22
Incremento 3	11 days	26/06/24 08:00	10/07/24 17:00	
Analisis y Diseño	2 days	26/06/24 08:00	27/06/24 17:00	23
Desarrollo	6 days	28/06/24 08:00	05/07/24 17:00	25
Testing	3 days	08/07/24 08:00	10/07/24 17:00	26
Incremento 4	10 days	11/07/24 08:00	24/07/24 17:00	
Analisis y Diseño	2 days	11/07/24 08:00	12/07/24 17:00	27
Desarrollo	5 days	15/07/24 08:00	19/07/24 17:00	29
Testing	3 days	22/07/24 08:00	24/07/24 17:00	30
Incremento 5	10 days	25/07/24 08:00	07/08/24 17:00	
Analisis y Diseño	2 days	25/07/24 08:00	26/07/24 17:00	31
Desarrollo	5 days	29/07/24 08:00	02/08/24 17:00	33
Testing	3 days	05/08/24 08:00	07/08/24 17:00	34
Incremento 6	10 days	08/08/24 08:00	21/08/24 17:00	
Analisis y Diseño	2 days	08/08/24 08:00	09/08/24 17:00	35
Desarrollo	5 days	12/08/24 08:00	16/08/24 17:00	37
Testing	3 days	19/08/24 08:00	21/08/24 17:00	38

Ilustración 19: “diagrama de Gantt del proyecto - desarrollo”

Evaluación	6 days	22/08/24 08:00	29/08/24 17:00	
Configuración del entorno de prueba	2 days	22/08/24 08:00	23/08/24 17:00	39
Pruebas Realizadas	1 day	26/08/24 08:00	26/08/24 17:00	41
Configuración de aplicación para prueba de usabilidad	2 days	22/08/24 08:00	23/08/24 17:00	39
Resultados Esperados	1 day	27/08/24 08:00	27/08/24 17:00	42
Resultados Obtenidos	1 day	28/08/24 08:00	28/08/24 17:00	44
Comparativa	1 day	29/08/24 08:00	29/08/24 17:00	45

Ilustración 20: “diagrama de Gantt del proyecto - Evaluación”

Análisis de Costos:

En la presente sección se abordarán en detalles los diversos costos a contemplar para el desarrollo de proyecto en su totalidad. Dichos costos están clasificados por recursos humanos (orientado a los profesionales que podrán construir la solución propuesta), el hardware y el software requeridos. A su vez es remarcable mencionar, que como este proyecto se llevó a cabo con fines académicos, todos los costos contemplados como las tareas de los respectivos roles fueron asumidas por el alumno.

Costos de Recursos Humanos:

Los Gastos de Personal comprenden los costos relacionados con el equipo humano requerido para la ejecución del proyecto. Para calcular estos gastos, se tomó como base la escala de honorarios del Colegio Profesional de Ingenieros, Arquitectos y Profesionales Afines (COPAIPA) como referencia para el sector informático (Consejo Profesional de Agrimensores, n.d.). Posteriormente, se contempló la conversión de los respectivos subtotales a sol peruano contemplando el contexto del desarrollo. Cabe destacar que esto solo se limita al estipulado en la planificación del presente proyecto. Aspectos del despliegue del mismo sobre la infraestructura de la universidad no son contemplados por lo que no se aborda para este un rol de Administrador de Bases de Datos o roles afines. Para trabajos futuros, se pueden plantear el trabajo en conjunto con el Centro de Datos propio de la institución y estimar nuevos valores acorde a dicha fecha.

Los recursos contemplados son:

Cargo	Cantidad Necesaria	Costo por Hora	Total de Horas en el Proyecto	Subtotal (ARS)	Subtotal (PE)
Analista Funcional	1	\$4080	1008	\$4.112.640	S/17540.96
Analista Programador	2	\$3400	1584	\$5.385.600	S/22970.30
TOTAL RR. HH				\$9.498.240	S/40511.26

Tabla 23: "Costos de Recursos Humanos"

Costos de Hardware:

Los Gastos de Hardware se refieren a los costos asociados con los equipos y dispositivos necesarios para la puesta en marcha del proyecto. Dado que ya se contaba con el equipo necesario para completar el proyecto en su totalidad, no fue necesario adquirir nuevos dispositivos o equipos de hardware. Se realizó una investigación sobre el costo actual del hardware utilizado para el proyecto, utilizando como referencia los precios publicados en Mercado Libre para productos similares en condiciones comparables.

Concepto	Cantidad Necesaria	Costo (ARS)	Costo (PEN)
Ultrabook Dell Inspiron 3505 gris 15.6", AMD Ryzen 5 3450U 16GB de RAM 256GB SSD, AMD Radeon RX Vega 8 (Ryzen 2000/3000) 60 Hz 1366x768px Windows 10 Home¹	1	\$949.999	S/ 4051.87
Notebook Lenovo Ideapad 310S-14AST Procesador: AMD A9 RAM: 8GB Almacenamiento: 256GB SSD Pantalla: 14" Gráfica: AMD Radeon R5²	2	\$599.998	S/ 2559.07
TOTAL		\$1.549.997	S/ 6610.94

Tabla 24: "Costos de Hardware"

Costos de Software:

La gestión de los Costos de Software en el proyecto se llevó a cabo aprovechando una serie de herramientas y plataformas de software disponibles de forma gratuita. En primer lugar, se utilizó la versión gratuita de Supabase para alojar la base de datos PostgreSQL requerida. Supabase ofrece una opción gratuita que proporciona una cantidad suficiente de almacenamiento y recursos para proyectos de tamaño moderado. Para el alojamiento del servicio de Django, se optó por el servicio gratuito de Render, el cual ofrece opciones de alojamiento sin costo para aplicaciones web y APIs. Además, para el

¹ Valor Tomado de Mercado Libre en fecha 14/04/2024: https://www.mercadolibre.com.ar/ultrabook-dell-inspiron-3505-gris-156-amd-ryzen-5-3450u-16gb-de-ram-256gb-ssd-amd-radeon-rx-vega-8-ryzen-20003000-60-hz-1366x768px-windows-10-home/p/MLA16999018#backend=item_decorator&backend_type=function&client=bookmarks-polycard

² Valor Tomado de Mercado Libre en fecha 14/04/2024: https://articulo.mercadolibre.com.ar/MLA-1739225040-notebook-lenovo-t440p-core-i5-sin-bateris-8gb-ssd-240-win10-JM?searchVariation=182456348479#searchVariation=182456348479&position=1&search_layout=stack&type=item&tracking_id=722afa37-2a62-4b79-bf5d-021fab9021c

desarrollo de la aplicación móvil, se emplearon frameworks gratuitos como Ionic y Angular. Ionic es un framework de desarrollo de aplicaciones móviles multiplataforma que ofrece una variedad de características útiles sin ningún costo asociado, mientras que Angular, utilizado en conjunto con Ionic, es un framework de desarrollo web de código abierto mantenido por Google. Por último, para el desarrollo del back-end de la aplicación, se utilizaron Django y Django REST Framework (DRF), ambos de forma gratuita. Django es un framework de desarrollo web de alto nivel en Python, mientras que DRF es una extensión de Django que facilita la creación de APIs RESTful. En conjunto, el uso de estas herramientas y plataformas de software gratuitas permitió reducir significativamente los costos asociados con el software utilizado en el proyecto, manteniendo al mismo tiempo un alto nivel de calidad y funcionalidad.

Costos Totales:

Los costos Totales para poder llevar a cabo el proyecto serán detallados a continuación:

Concepto	Costo (ARS)	Costo (PEN)
Costo de Recursos Humanos	\$9.498.240	S/40511.26
Costo de Hardware	\$1.549.997	S/ 6610.94
Costo de Software	\$0.00	S/ 0.00
Total	\$11.048.247	S/47.122.2

Tabla 25: "Costos Totales"

Análisis de Factibilidad:

En el presente apartado se llevará a cabo el análisis de factibilidad comprendiendo varios aspectos del mismo proyecto que determinan su viabilidad como son lo económico, operativo, técnico y legal.

Factibilidad Económica:

La sección de Factibilidad Económica del proyecto se enfoca en evaluar si el desarrollo de este es financieramente viable y si puede generar beneficios suficientes para cubrir los costos asociados. En este contexto, se concluye que el proyecto es factible económicamente. Esto se fundamenta en la disponibilidad de la infraestructura de hardware necesaria y las licencias de software requeridas para llevar a cabo el proyecto. Por lo tanto, los únicos gastos que se deben contemplar son los relacionados con los Recursos Humanos. Los beneficios que ofrece el mismo proyecto se conforman en una primera instancia en la reducción de costos administrativos, optimizando los tiempos de los empleados del centro de estudiantes a la hora de tener que buscar y proveer información académica de cada estudiante que se presenta en la misma, puesto que ahora con esta herramienta desarrollada, se puede emplear para consultar esa misma información. Esta herramienta también puede contribuir a la retención estudiantil al mejorar la experiencia académica y reducir la frustración asociada con la burocracia administrativa. Como último punto a destacar, ya que la aplicación cuenta con la funcionalidad de mostrar los diversos eventos que se desarrollaran en la institución; la misma herramienta entonces, puede servir como otro medio o canal de difusión para la universidad, reduciendo costos en proyectos de difusión y marketing.

Factibilidad Operativa:

En el contexto del proyecto de la aplicación de autogestión para la Universidad Católica de Santa María, la Factibilidad Operativa se refiere a la evaluación de si se cuenta con los recursos humanos necesarios para ejecutar el proyecto. En este sentido, se determina que el proyecto es factible operativamente, dado que se dispone de los recursos humanos requeridos y se cuenta con la capacitación necesaria para abordar el proyecto de manera efectiva.

Factibilidad Técnica:

En el contexto del proyecto de la aplicación de autogestión para la Universidad Católica de Santa María, la Factibilidad Técnica se refiere a la evaluación de la capacidad técnica y tecnológica para la implementación del proyecto, incluyendo la infraestructura necesaria para su desarrollo y funcionamiento adecuado. En este sentido, el proyecto cuenta con una factibilidad técnica adecuada puesto que se dispone de la infraestructura de hardware necesaria para el desarrollo y la ejecución de la aplicación, incluyendo servidores y dispositivos móviles para pruebas. Además, se ha seleccionado un conjunto de tecnologías y herramientas adecuadas para el desarrollo del proyecto, como Django, Django REST Framework, Ionic, Angular y PostgreSQL, todas ellas ampliamente utilizadas y con una sólida base técnica.

La infraestructura de software necesaria para las pruebas del proyecto está disponible de manera gratuita, lo que garantiza la viabilidad técnica del proyecto. Se ha optado por utilizar servicios como Supabase y Render, que ofrecen opciones gratuitas para alojamiento de bases de datos y hosting de aplicaciones web, respectivamente. Además, las tecnologías seleccionadas son conocidas por su estabilidad, seguridad y capacidad para satisfacer las necesidades del proyecto.

Factibilidad Legal:

Dentro del contexto del proyecto de la aplicación de autogestión para la Universidad Católica de Santa María, la Factibilidad Legal se refiere a la evaluación exhaustiva de si el proyecto cumple con todas las leyes y regulaciones vigentes, tanto a nivel local como nacional, relacionadas con la privacidad de datos, la protección de la información personal, la propiedad intelectual y cualquier otra normativa relevante.

Para garantizar la Factibilidad Legal del proyecto, es crucial realizar un análisis detallado de todas las leyes y regulaciones aplicables al tratamiento de datos personales de los estudiantes y cualquier otra información sensible que se maneje a través de la aplicación. En este sentido, se deben considerar normativas como la Ley de Protección de Datos Personales (El Peruano, n.d.), las regulaciones de privacidad en línea y las políticas de transformación digital del Perú (Presidencia del Consejo de Ministros, n.d.).

Ahora, a partir de las restricciones del proyecto como tal, no se usó ningún dato sensible de los estudiantes de la UCSM para las pruebas del MVP, y no se expuso en ningún momento las credenciales del servicio actual de la universidad; el proyecto está acorde con lo que estipula la Ley N°29733 de protección de datos personales. También el presente proyecto promueve lo estipulado en el Decreto Supremo N°118-2018-PCM de aprovechar las oportunidades de las nuevas tecnologías digitales, la innovación en los procesos productivos, las aplicaciones digitales y la economía digital para lograr la transformación digital del Perú.

Capítulo 6: “Resultados”

Introducción:

En esta sección, se abordarán las pruebas realizadas en el contexto del presente proyecto, las cuales se dividen en tres categorías principales: pruebas funcionales, pruebas de integración y pruebas de usabilidad. Estas pruebas desempeñan un papel crucial en la evaluación del rendimiento y la eficacia tanto de la aplicación móvil desarrollada como de la API REST implementada utilizando Django Rest Framework.

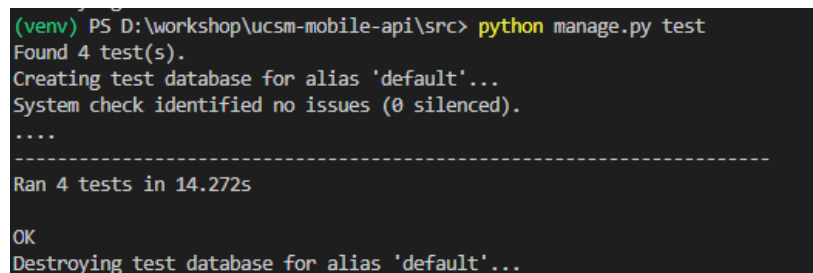
Pruebas Realizadas:

Para el presente proyecto se buscó realizar tres tipos de pruebas; por un lado, pruebas funcionales, probando cada endpoint de la aplicación cumpla con los criterios expresados en las historias de usuario. Por otro lado, en base a pruebas de integración entre el front-end (la aplicación móvil desarrollada) y el back-end (API REST de Django Rest Framework), se evaluará la unificación de ambas. Por último, pruebas de usabilidad las cuales se centran en evaluar la facilidad de uso, la accesibilidad y la satisfacción del usuario al interactuar con la aplicación.

Pruebas Funcionales:

Las pruebas funcionales se realizaron tanto para el back-end como para la aplicación móvil, con un enfoque principal en la API y los endpoints definidos en Django Rest Framework (DRF). Estas pruebas fueron esenciales para asegurar que cada funcionalidad de la aplicación cumpliera con los requisitos especificados y operara de manera correcta y eficiente.

En el back-end, se definieron y ejecutaron casos de prueba específicos para cada uno de los endpoints de la API. Utilizando las herramientas proporcionadas por DRF, se automatizaron algunos casos de prueba críticos. En primer lugar, se verificó que un usuario pudiera iniciar sesión correctamente proporcionando las credenciales válidas mediante un caso de prueba de Inicio de Sesión Exitoso. En segundo lugar, se evaluaron escenarios en los que las credenciales eran incorrectas o faltaban, asegurando que el sistema respondiera con los mensajes de error adecuados en el caso de Inicio de Sesión Fallido. Adicionalmente, se comprobó que un usuario pudiera cerrar sesión de manera segura y que su token de sesión se invalidara correctamente en el caso de Fin de Sesión. Finalmente, se probó la capacidad de recuperar el perfil completo de un estudiante utilizando un identificador válido en el caso de Recuperación del Perfil del Estudiante.



```
(venv) PS D:\workshop\ucsm-mobile-api\src> python manage.py test
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
-----
Ran 4 tests in 14.272s

OK
Destroying test database for alias 'default'...
```

Ilustración 21 "Pruebas funcionales con django test"

Para evaluar el resto de los casos de prueba, se empleó Postman, una herramienta popular para el desarrollo y la prueba de APIs. Postman ofrece varias ventajas que contribuyeron significativamente a la eficiencia y la efectividad de las pruebas. En primer lugar, su facilidad de uso proporciona una interfaz intuitiva que permite a los desarrolladores crear, organizar y ejecutar pruebas de API de manera rápida y sencilla. En segundo lugar, aunque se utilizó principalmente para pruebas manuales, Postman permite la automatización de pruebas a través de scripts, lo que facilita la ejecución repetitiva y consistente de casos de prueba. Además, Postman permite verificar fácilmente las respuestas de la API, asegurándose

de que los datos retornados sean correctos y cumplan con las expectativas. La gestión de variables y entornos en Postman también es una ventaja significativa, ya que permite simular diferentes escenarios y configuraciones sin necesidad de modificar el código de prueba. Finalmente, Postman ofrece capacidades de documentación integrada, permitiendo mantener un registro detallado de los casos de prueba y sus resultados.

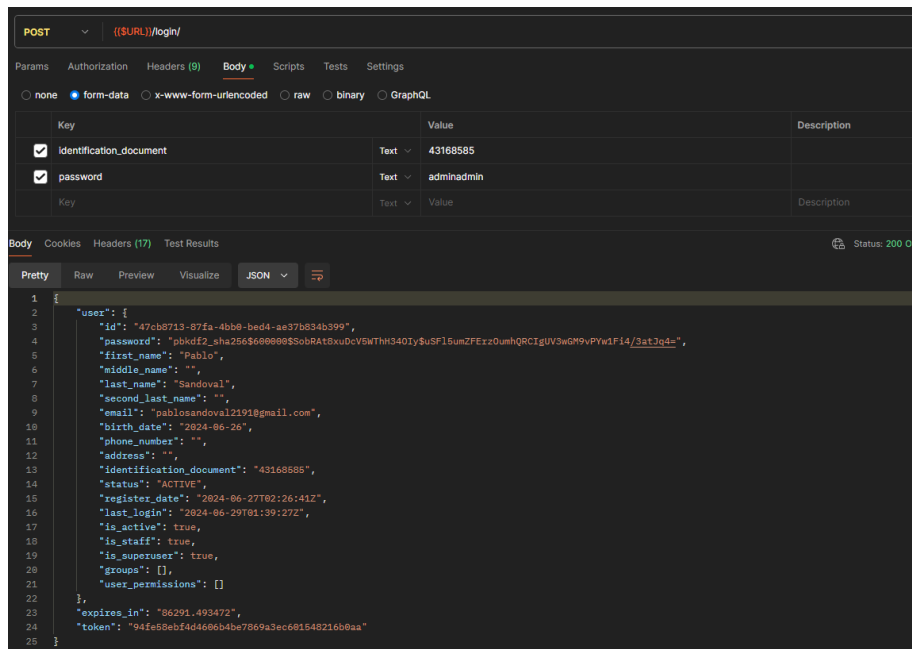


Ilustración 22 "Caso de Prueba de inicio de sesión exitoso"

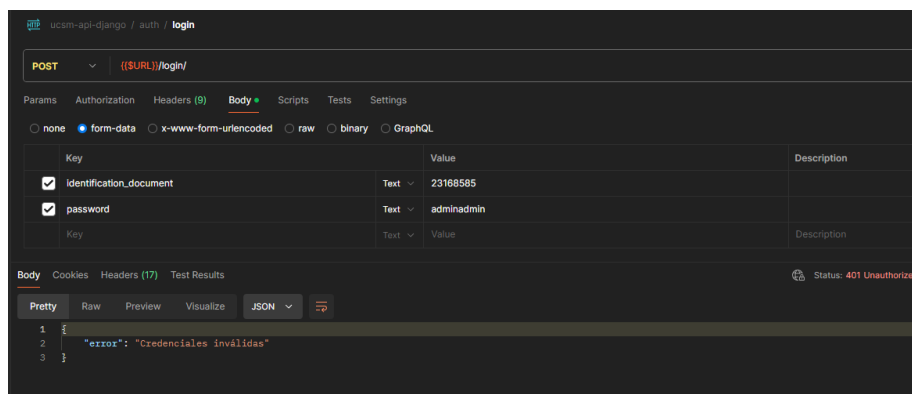
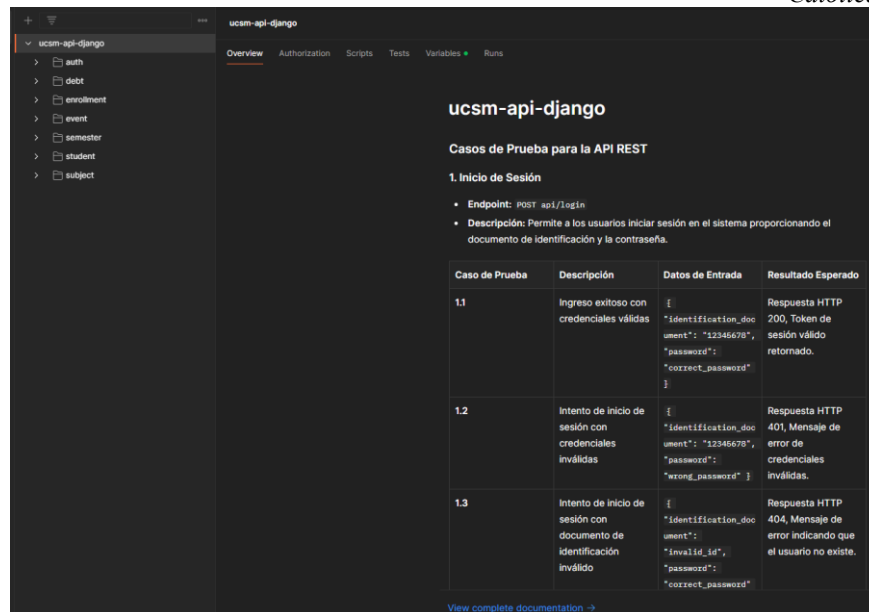


Ilustración 23 "Caso de Prueba de inicio de sesión fallido"

Los casos de prueba definidos y ejecutados abarcaron una amplia gama de funcionalidades de la API. Para la Consulta de Deuda, se verificó la capacidad de recuperar la deuda de un curso específico y sus cuotas asociadas. En cuanto a la Información de Inscripción, se evaluó la capacidad de obtener la lista de cursos en los que un estudiante está inscrito y los detalles de un curso específico. Para la Gestión de Eventos, se probaron las funcionalidades para recuperar la lista de eventos disponibles y los detalles de eventos individuales. En la categoría de Acceso a Semestres, se verificó la capacidad de recuperar la lista de semestres disponibles y la información detallada de un semestre específico. Para el Perfil del Estudiante, se realizaron pruebas para asegurar que se pudiera recuperar el perfil completo de un estudiante. Por último, en el módulo de Registro de Materias, se evaluó la capacidad de manejar la información de las materias inscritas, los detalles de materias individuales, las materias restantes por cursar y las estadísticas de materias. En el anexo 2 se detalla los casos de pruebas abordados para la API REST.



ucsm-api-django

Casos de Prueba para la API REST

1. Inicio de Sesión

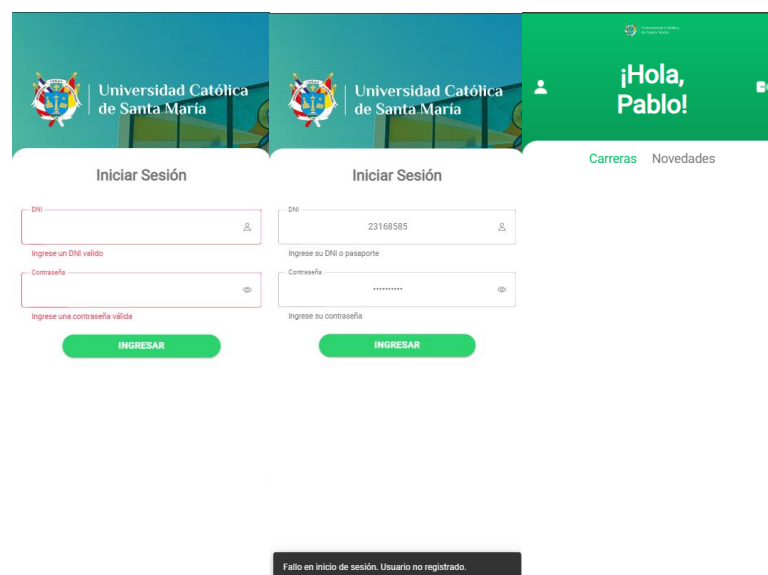
- Endpoint: POST api/login
- Descripción: Permite a los usuarios iniciar sesión en el sistema proporcionando el documento de identificación y la contraseña.

Caso de Prueba	Descripción	Datos de Entrada	Resultado Esperado
1.1	Ingreso exitoso con credenciales válidas	{ "identification_document": "12345678", "password": "correct_password" }	Respuesta HTTP 200, Token de sesión válido retornado.
1.2	Intento de inicio de sesión con credenciales inválidas	{ "identification_document": "12345678", "password": "wrong_password" }	Respuesta HTTP 401, Mensaje de error de credenciales inválidas.
1.3	Intento de inicio de sesión con documento de identificación inválido	{ "identification_document": "invalid_id", "password": "correct_password" }	Respuesta HTTP 404, Mensaje de error indicando que el usuario no existe.

[View complete documentation ->](#)

Ilustración 24 "Documentación de casos de pruebas en Postman"

En el marco del proyecto de la aplicación móvil, se verificó la funcionalidad de inicio de sesión, donde los usuarios pudieron acceder a la aplicación con credenciales válidas, siendo redirigidos a la pantalla principal y recibiendo un mensaje de bienvenida; así como se comprobó que, al ingresar credenciales inválidas, se desplegara un mensaje de error adecuado. Adicionalmente, se evaluó la visualización de horarios académicos, confirmando que los estudiantes pudieran acceder a información completa y organizada sobre sus clases, incluyendo detalles como el nombre del profesor, la ubicación del aula y los identificadores de sección y grupo práctico.



Universidad Católica de Santa María

¡Hola, Pablo!

Carreras Novedades

Iniciar Sesión

DNI:

Ingrese un DNI válido

Contraseña:

Ingrese una contraseña válida

INGRESAR

DNI:

Ingrese su DNI o pasaporte

Contraseña:

Ingrese su contraseña

INGRESAR

Fallo en inicio de sesión. Usuario no registrado.

Ilustración 25 "Casos de pruebas de inicio de sesión en aplicación móvil"

Asimismo, se realizaron pruebas para la visualización de deudas, permitiendo a los usuarios consultar sus deudas de matrícula y las boletas pagadas con todos sus detalles pertinentes. Se incluyeron también pruebas para el listado de inscripciones, donde se verificó que los estudiantes pudieran ver todos los cursos en los que están inscritos, y para la gestión de eventos, asegurando que los usuarios pudieran acceder a un listado completo de eventos universitarios con información relevante. Por otra parte, se

llevaron a cabo pruebas en la sección de perfil del usuario, confirmando que los datos personales y académicos se presentaran de manera completa y precisa.

Además, se evaluaron las funcionalidades relacionadas con las materias inscritas, permitiendo a los usuarios acceder a una lista detallada de sus materias actuales, así como las materias restantes por cursar, asegurando una visión clara de su progreso académico. Finalmente, se realizaron pruebas para las estadísticas de materias, donde los usuarios pudieron visualizar de manera efectiva el número de materias aprobadas y las restantes por cursar.

Pruebas de Integración:

Para la evaluación de integración en la aplicación móvil de autogestión estudiantil, se llevó a cabo un conjunto exhaustivo de pruebas que simulaban la interacción completa del usuario con el sistema. Este proceso abarcó desde la navegación por la interfaz de usuario hasta la validación de los datos almacenados en la base de datos. Se inició con la carga de datos de prueba utilizando la interfaz de administración de Django, asegurando que los datos estuvieran disponibles y correctamente estructurados para las pruebas subsiguientes.

Adicionalmente, se llevó a cabo la renderización de un archivo APK de la aplicación móvil para realizar pruebas reales en dispositivos físicos. Esto implicó consumir el servicio API desde un dispositivo móvil, garantizando así que la aplicación funcionara de manera óptima en un entorno real de uso. Los resultados detallados de estas pruebas de integración se documentaron exhaustivamente en el proceso de evaluación, proporcionando una visión clara del desempeño y la robustez del sistema en diferentes escenarios operativos.

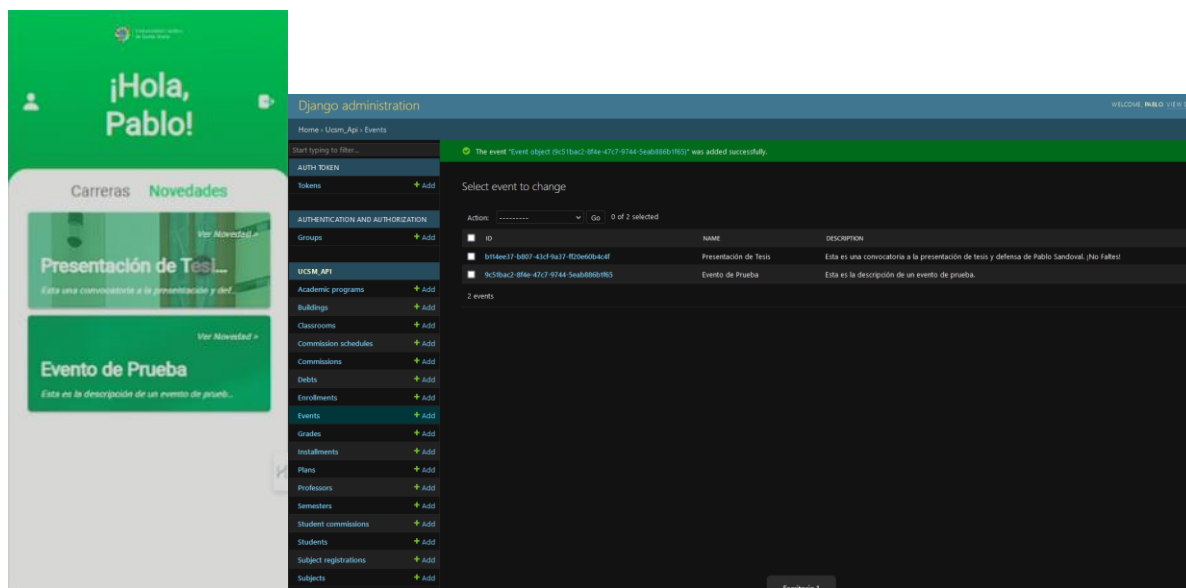


Ilustración 26 "Carga de datos de prueba para pruebas de integración con aplicación móvil"

Pruebas de Usabilidad:

Para la evaluación de usabilidad se aplicó un cuestionario en "Google Forms" basado en la herramienta de evaluación estándar utilizada para medir la usabilidad de un sistema, producto o servicio denominada "System Usability Scale" con el objetivo de conocer la opinión de los usuarios e identificar áreas de mejora. El grupo de encuestados está conformado por los estudiantes del noveno semestre de Ingeniería en Sistemas de la Universidad Católica de Santa María. Esta evaluación consta de una serie de preguntas diseñadas para evaluar la facilidad de uso percibida y la satisfacción del usuario. Estas preguntas están diseñadas para ser simples y directas, y cubren una variedad de aspectos relacionados con la usabilidad, como la facilidad de aprendizaje, la eficiencia del sistema, la facilidad de uso y la satisfacción general

del usuario. Cabe destacar que, para fines prácticos, solo se generó la versión para dispositivos Android generando un apk el cual permite instalar la aplicación en el dispositivo. A su vez, se puso disposición el respectivo instalador en una plataforma de descarga gratuita llamada “MediaFire” para que pueda descargarlo y usarlo. El mismo cuestionario se puede observar en la figura:

The screenshot shows a Google Form titled "UCSM Mobile 2024". The form content includes a welcome message, a description of the evaluation purpose (System Usability Scale), a link to the APK file on MediaFire, and a question about the ease of use of the mobile application. The question is: "¿Considera que la aplicación móvil es fácil de usar?" with a scale from 1 (Totalmente en desacuerdo) to 5 (Totalmente de acuerdo). The form also shows the user's email address (pablosandovali2191@gmail.com) and a "No compartido" status.

Ilustración 27: “Evaluación de Usabilidad por medio de Google Forms”

Resultados Esperados:

Se anticipa que los resultados de este proyecto reflejen una integración efectiva entre la aplicación móvil y la API desarrollada, gracias a la metodología incremental implementada. Este enfoque permitió la creación simultánea y progresiva de los módulos tanto de la API como de la aplicación móvil, lo que facilitó las pruebas en paralelo y la identificación temprana de posibles problemas de integración. Se espera que esta estrategia contribuya a la cohesión y compatibilidad entre los componentes del sistema, garantizando un funcionamiento armonioso y una experiencia de usuario fluida.

En términos de seguridad, se prevé que los resultados sean satisfactorios debido a la atención dedicada a contemplar los riesgos de seguridad según las pautas de OWASP (Open Web Application Security Project). Tanto en el desarrollo de la aplicación móvil como en la implementación de la API, se han tenido en cuenta las mejores prácticas de seguridad recomendadas por OWASP, lo que debería reducir significativamente la exposición a vulnerabilidades y ataques cibernéticos. Se espera que este enfoque proactivo en seguridad contribuya a la protección de los datos del usuario y a la confianza en la aplicación.

Además, se aguarda obtener buenos resultados en términos de usabilidad, gracias a la adopción de prácticas de diseño centradas en el usuario (UX/UI). El seguimiento de estas prácticas garantiza una interfaz intuitiva y amigable para el usuario, lo que se traduce en una experiencia de usuario positiva y satisfactoria. Se espera que la aplicación móvil diseñada con atención a los principios de UX/UI proporcione una navegación fluida y una interacción intuitiva, lo que contribuirá a la satisfacción general del usuario y a una mayor adopción de la aplicación.

Resultados Obtenidos:

En la evaluación de Usabilidad, se obtuvo 25 opiniones de los usuarios, las cuales se pueden observar en la tabla de resultados mostrada en Tabla 11. De la misma, se consideró como promedio final un total de 46.2 de 50, demostrando que el software pasó la prueba de usabilidad considerándose eficiente.

¿Considera que la aplicación móvil es fácil de usar? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿Se siente confiado al utilizar la aplicación móvil? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿Puede realizar las tareas en la aplicación móvil sin mucha dificultad? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿Encuentra que la aplicación móvil es intuitiva y fácil de entender? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿El diseño de la aplicación móvil le parece atractivo y agradable? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿Se siente cómodo navegando por la aplicación móvil? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿Encuentra que la aplicación móvil es consistente en su funcionamiento? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿La aplicación móvil le permite completar sus tareas de manera eficiente? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿Encuentra que la aplicación móvil tiene una organización lógica de la información? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	¿La aplicación móvil le proporciona retroalimentación clara sobre sus acciones? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). [.]	Totales
5	5	5	5	5	5	4	5	4	5	48
4	4	3	3	3	3	3	3	3	3	32
5	3	5	5	5	5	5	5	5	4	47
5	5	5	5	5	5	5	5	5	5	50
5	4	4	5	5	5	3	5	4	5	45
5	5	5	5	5	5	4	3	4	4	45
4	5	3	4	4	3	3	3	4	2	35
5	5	5	5	5	5	5	5	5	4	49
5	4	5	5	4	5	4	4	4	4	44
4	4	4	4	4	4	4	4	4	4	40
5	5	5	5	5	5	5	5	5	5	50
5	5	5	5	5	5	5	5	5	5	50
5	5	5	5	5	5	5	5	5	5	50
4	4	4	4	4	4	4	4	4	4	40
5	5	5	5	5	5	5	5	5	5	50
5	5	5	5	5	5	5	5	5	5	50
5	5	5	5	5	5	5	5	5	5	50
5	5	5	5	4	5	5	5	5	5	49
5	4	3	5	4	5	5	5	3	4	43
4	4	5	5	5	5	5	4	5	5	47
4	4	4	4	4	5	4	4	4	4	41
5	5	5	5	5	5	5	5	5	5	50
5	5	5	5	5	5	5	5	5	5	50
5	5	5	5	5	5	5	5	5	5	50
5	5	5	5	5	5	5	5	5	5	50
Promedio	4.76	4.6	4.6	4.76	4.64	4.76	4.52	4.52	4.48	46,2

Tabla 26: “Tabla de resultados de la Evaluación de Usabilidad”

La posibilidad que me ofrece “Google Forms” es el poder visualizar los resultados obtenidos en cada pregunta en un gráfico de barras, los cuales se puede observar en cada una de las siguientes figuras:

¿Considera que la aplicación móvil es fácil de usar? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

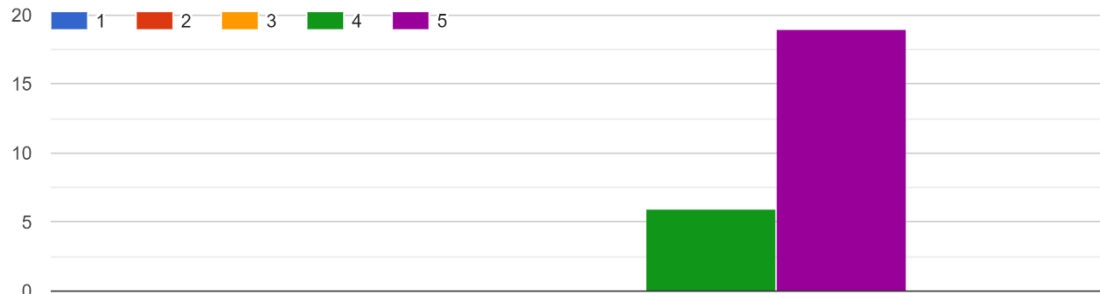


Ilustración 28: “Grafico de respuestas de la pregunta 1”

¿Se siente confiado al utilizar la aplicación móvil? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

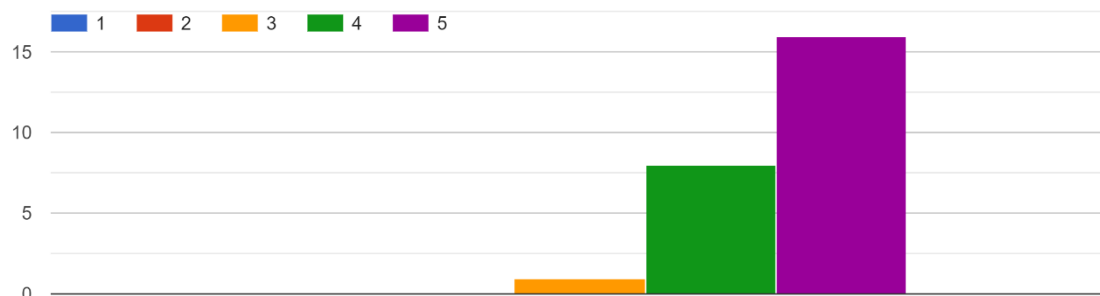


Ilustración 29: “Grafico de respuestas de la pregunta 2”

¿Puede realizar las tareas en la aplicación móvil sin mucha dificultad? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

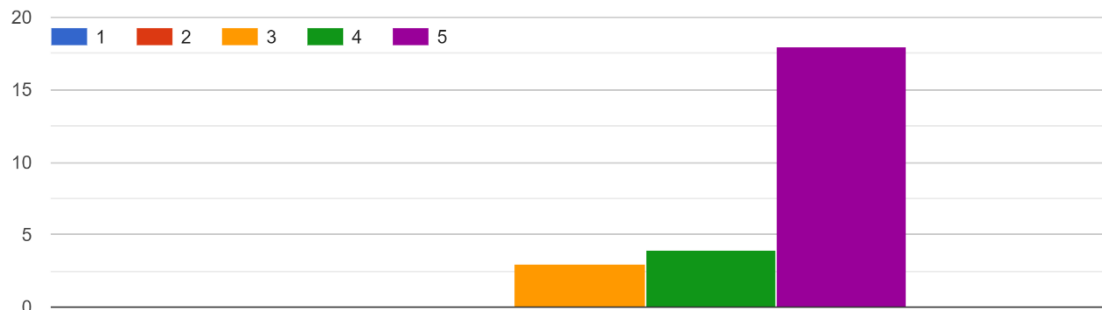


Ilustración 30: “Grafico de respuestas de la pregunta 3”

¿Encuentra que la aplicación móvil es intuitiva y fácil de entender? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

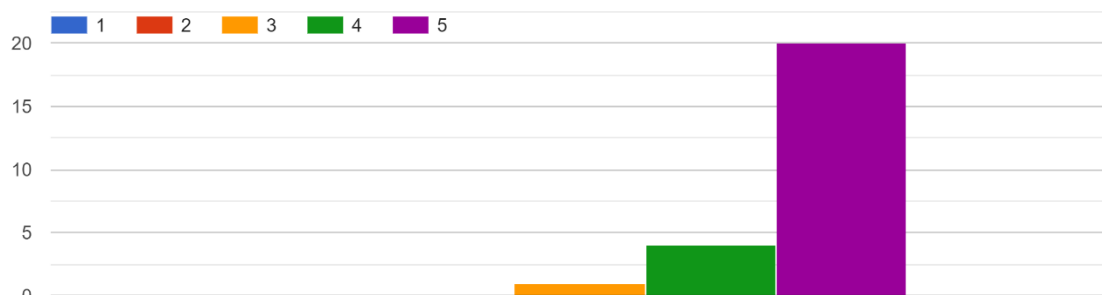


Ilustración 31: “Grafico de respuestas de la pregunta 4”

¿El diseño de la aplicación móvil le parece atractivo y agradable? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

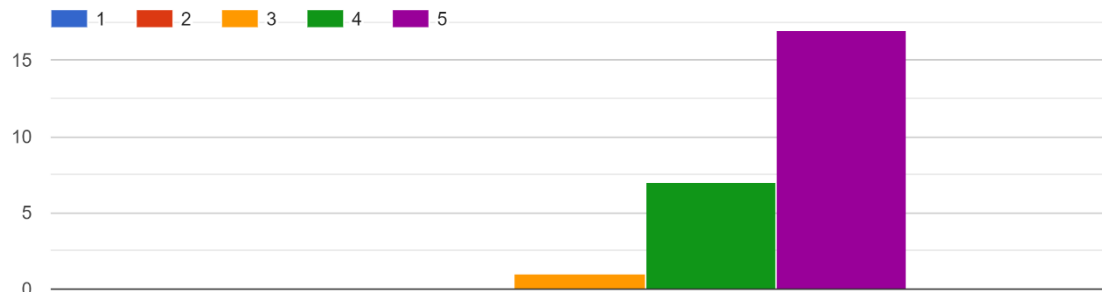


Ilustración 32: “Grafico de respuestas de la pregunta 5”

¿Se siente cómodo navegando por la aplicación móvil? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

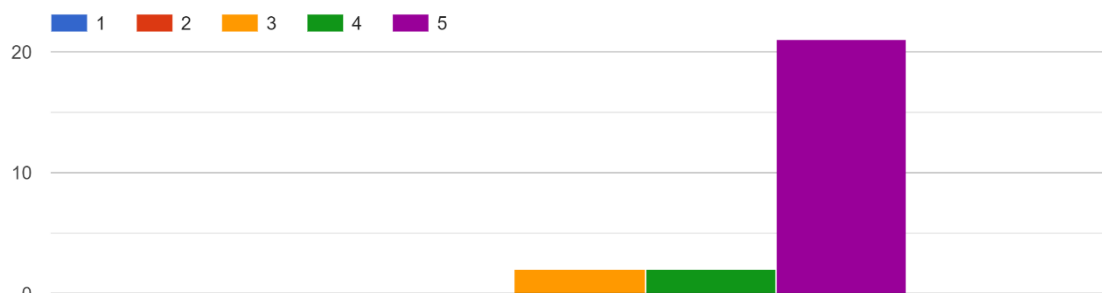


Ilustración 33: “Grafico de respuestas de la pregunta 6”

¿Encuentra que la aplicación móvil es consistente en su funcionamiento? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

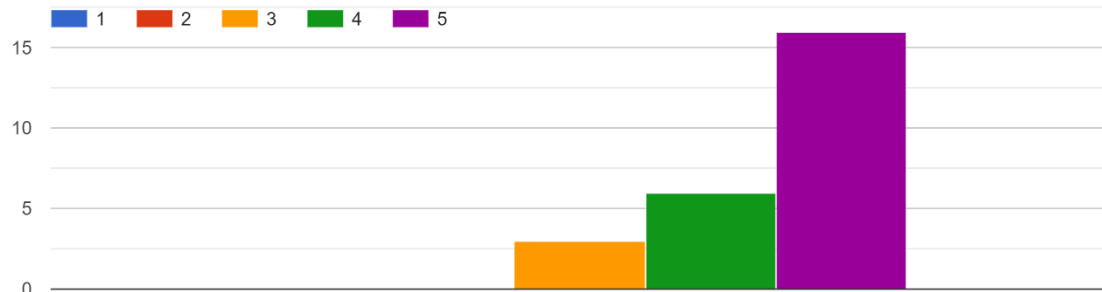


Ilustración 34: “Grafico de respuestas de la pregunta 7”

¿La aplicación móvil le permite completar sus tareas de manera eficiente? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

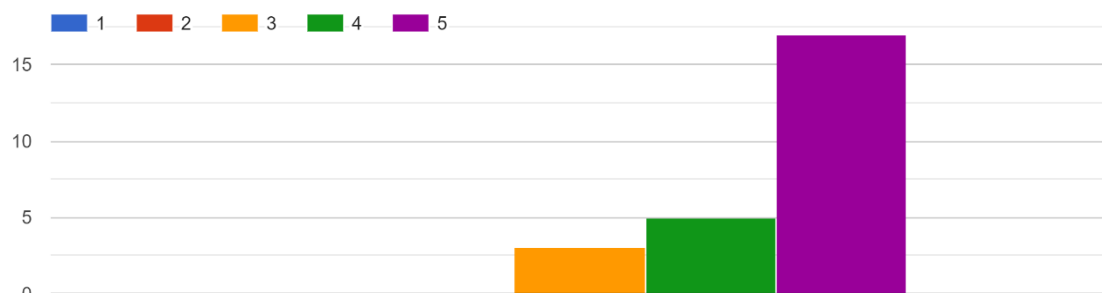


Ilustración 35: “Grafico de respuestas de la pregunta 8”

¿Encuentra que la aplicación móvil tiene una organización lógica de la información? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

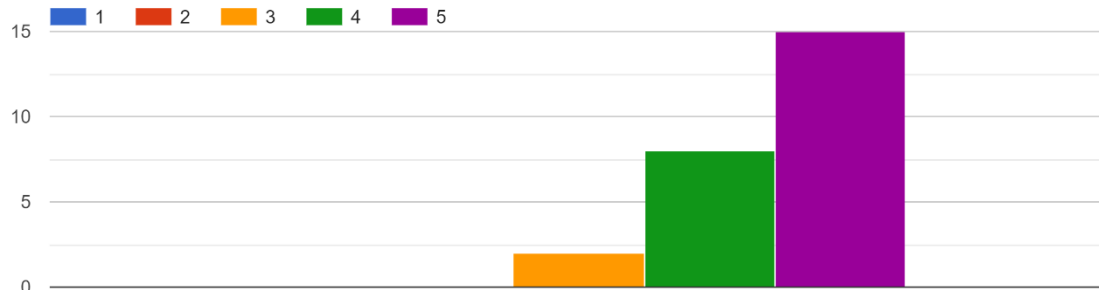


Ilustración 36: “Grafico de respuestas de la pregunta 9”

¿La aplicación móvil le proporciona retroalimentación clara sobre sus acciones? desde 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo).

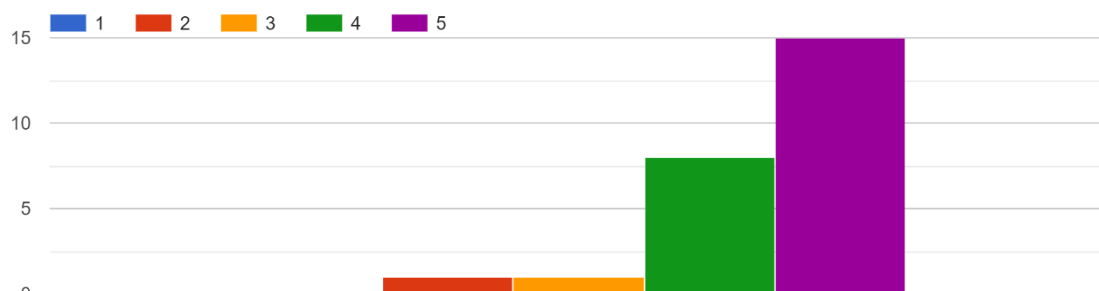


Ilustración 37: “Grafico de respuestas de la pregunta 10”

De los datos recabados, se destaca que la aplicación en su conjunto no resultó percibida como compleja por la mayoría de los evaluadores. Además, los participantes consideraron que sería fácil para otras personas aprender a utilizar este sistema, lo que indica una alta capacidad de aprendizaje y adaptación por parte de los usuarios potenciales.

Estos resultados demuestran un alto nivel de satisfacción por parte de los evaluadores con el diseño y la funcionalidad de la aplicación desarrollada. Se confirma así que se han seguido adecuadamente las prácticas de diseño centrado en el usuario y se han implementado eficazmente los principios de experiencia de usuario (UX) y diseño de interfaz de usuario (UI), lo que ha contribuido significativamente a la percepción positiva de la usabilidad por parte de los usuarios.

Las pruebas realizadas con los servicios serverless, utilizando Render para la API y Supabase para la base de datos, arrojaron resultados exitosos. Se logró verificar de manera satisfactoria el funcionamiento de la aplicación a través de un APK alternativo diseñado específicamente para probar la API. Este enfoque permitió evaluar de forma precisa la capacidad de la API para manejar las solicitudes y procesar

los datos de manera eficiente, asegurando así la robustez y la escalabilidad del sistema. Los resultados obtenidos respaldan la fiabilidad y el rendimiento de la arquitectura serverless implementada, lo que confirma su idoneidad para el proyecto.

Comparativa:

Los resultados esperados y obtenidos en el proyecto proporcionan una visión completa de su desempeño y efectividad.

En cuanto a la integración entre la aplicación móvil y la API desarrollada, se anticipaba una cohesión efectiva gracias a la metodología incremental implementada. Esta estrategia permitió la creación progresiva de los módulos, facilitando las pruebas en paralelo y la detección temprana de problemas de integración. Los resultados obtenidos refuerzan esta expectativa, demostrando una percepción positiva de la usabilidad y una experiencia de usuario satisfactoria.

En términos de seguridad, se esperaba que la aplicación estuviera protegida contra vulnerabilidades y ataques cibernéticos, gracias a la implementación de las mejores prácticas de seguridad de OWASP. Los resultados obtenidos respaldan esta expectativa, indicando que el software ha superado la prueba de usabilidad y se considera eficiente.

En lo que respecta a la usabilidad, se esperaba una interfaz intuitiva y amigable para el usuario, siguiendo las prácticas de diseño centradas en el usuario. Los resultados de la evaluación de usabilidad muestran una alta satisfacción de los evaluadores con el diseño y la funcionalidad de la aplicación, confirmando que se han implementado adecuadamente los principios de experiencia de usuario (UX) y diseño de interfaz de usuario (UI).

Capítulo 7: “Conclusión”

La culminación exitosa de este proyecto, que abarca desde el diseño y desarrollo de una aplicación móvil hasta la implementación de una API REST, marca un hito significativo para la Universidad Católica de Santa María (UCSM) en Arequipa, Perú. Este logro se fundamenta en un enfoque meticuloso y colaborativo que ha permitido mejorar tanto la experiencia de los estudiantes como la gestión académica dentro de la institución.

Al adoptar la metodología incremental, se pudo avanzar de manera eficiente y efectiva en la creación de la aplicación móvil y la API REST. Este enfoque nos permitió desarrollar y probar progresivamente cada módulo, facilitando la detección temprana de problemas y la integración sin problemas de ambos componentes. Los resultados de las pruebas de integración entre el front-end y el back-end fueron altamente satisfactorios, validando la efectividad de nuestra estrategia de desarrollo.

En términos de seguridad, nos comprometimos a seguir las mejores prácticas recomendadas por OWASP, lo que nos permitió identificar y mitigar posibles vulnerabilidades en la aplicación móvil y la API REST. Los costos asociados con el desarrollo del proyecto se mantuvieron dentro de los límites establecidos, gracias a la planificación cuidadosa y al uso eficiente de los recursos disponibles.

Las pruebas de usabilidad arrojaron resultados alentadores, con una puntuación promedio de 46.2 sobre 50 en la escala SUS (System Usability Scale). Esto indica una percepción positiva por parte de los usuarios sobre la facilidad de uso y la eficiencia del sistema, respaldando nuestra atención a las prácticas de diseño centradas en el usuario.

Trabajos Futuros:

El desarrollo de la aplicación móvil y la API REST para la Universidad Católica de Santa María (UCSM) sienta las bases para una gestión académica más eficiente y una experiencia estudiantil mejorada. Sin embargo, existen áreas de mejora y expansión que deben abordarse en futuros trabajos.

En primer lugar, se propone la implementación y despliegue a mayor escala de la aplicación en la UCSM. Esta fase implicaría un despliegue en producción que permita a todos los estudiantes acceder a la aplicación y experimentar sus beneficios en tiempo real. Para evaluar el impacto de esta implementación, se llevarán a cabo pruebas A/B, comparando el sistema actual con la nueva aplicación. Esto no solo medirá la efectividad de la solución, sino también permitirá determinar si la inversión realizada en el desarrollo valió la pena y si se traduce en mejoras significativas en la administración académica y la experiencia estudiantil.

Otro aspecto crucial de los trabajos futuros es la adición de módulos que habiliten transacciones financieras dentro de la aplicación. Esta expansión brindará a los estudiantes la comodidad de gestionar sus asuntos financieros directamente desde la aplicación, eliminando la necesidad de desplazarse físicamente a las oficinas de la universidad. Sin embargo, esta implementación debe realizarse con extrema precaución para garantizar la seguridad de las transacciones financieras y proteger los datos sensibles de los estudiantes.

Bibliografía:

- About npm* | *npm Docs*. (n.d.). Retrieved April 5, 2024, from <https://docs.npmjs.com/about-npm>
- About the OWASP Foundation* | *OWASP Foundation*. (n.d.). Retrieved October 7, 2023, from <https://owasp.org/about/>
- Angular - Angular coding style guide*. (n.d.). Retrieved October 8, 2023, from <https://angular.io/guide/styleguide>
- Angular - Getting started with standalone components*. (n.d.). Retrieved March 31, 2024, from <https://angular.io/guide/standalone-components>
- Angular - Lazy-loading feature modules*. (n.d.). Retrieved October 9, 2023, from <https://angular.io/guide/lazy-loading-ngmodules>
- Banca Móvil BCP - Apps en Google Play*. (n.d.). Retrieved September 23, 2023, from https://play.google.com/store/apps/details?id=com.bcp.bank.bcp&hl=es_419
- Capacitor - Cross-platform Native Runtime for Web Apps* | *Capacitor Documentation*. (n.d.). Retrieved September 22, 2023, from <https://capacitorjs.com/docs>
- Cloud Application Hosting for Developers* | *Render*. (n.d.). Retrieved June 25, 2024, from <https://render.com/>
- Consejo Profesional de Agrimensores, I. y P. A. (n.d.). *Consejo Profesional de Agrimensores, Ingenieros y Profesiones Afines*. Retrieved April 13, 2024, from <https://copaipa.org.ar/>
- Cross-platform mobile frameworks used by global developers 2022* | *Statista*. (n.d.). Retrieved April 2, 2024, from <https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/>
- Dependabot*. (n.d.). Retrieved June 29, 2024, from <https://github.com/dependabot>
- Deploy app servers close to your users · Fly*. (n.d.). Retrieved June 29, 2024, from <https://fly.io/>
- Django Rest Framework vs. Django Ninja: Choosing the Right Framework for Your API* | by *StartxLabs Technologies* | *Medium*. (n.d.). Retrieved April 10, 2024, from <https://medium.com/@StartXLabs/django-rest-framework-vs-django-ninja-choosing-the-right-framework-for-your-api-377bc925dcb6>
- Django Software Foundation. (2022). *Django Documentation Release*. <https://buildmedia.readthedocs.org/media/pdf/django/3.0.x/django.pdf>
- Docker: Accelerated Container Application Development*. (n.d.). Retrieved June 25, 2024, from <https://www.docker.com/>
- Docker Compose overview* | *Docker Docs*. (n.d.). Retrieved June 25, 2024, from <https://docs.docker.com/compose/>
- Dorfman, M. (Merlin), & Thayer, R. H. (1990). *Standards, guidelines, and examples on system and software requirements engineering*. 607.
- Download Android Studio & App Tools - Android Developers*. (n.d.). Retrieved June 28, 2024, from <https://developer.android.com/studio>
- dpage/pgadmin4 - Docker Image* | *Docker Hub*. (n.d.). Retrieved June 29, 2024, from <https://hub.docker.com/r/dpage/pgadmin4/>

El Peruano. (n.d.). *Ley de Datos Personales*.

Escuelas Profesionales - UCSM. (n.d.). Retrieved September 23, 2023, from <https://www2.ucsm.edu.pe/escuelas-profesionales/>

Fast, disk space efficient package manager | pnpm. (n.d.). Retrieved April 2, 2024, from <https://pnpm.io/es/>

Figma: la herramienta de diseño de interfaz colaborativo. (n.d.). Retrieved June 23, 2024, from <https://www.figma.com/es-la/>

Flujo de trabajo de Gitflow | Atlassian Git Tutorial. (n.d.). Retrieved April 5, 2024, from <https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow>

Git. (n.d.). Retrieved June 25, 2024, from <https://git-scm.com/>

GitHub. (n.d.). Retrieved June 25, 2024, from <https://github.com/>

Holovaty, A., & Kaplan-Moss, J. (2008). *El libro de Django*.

Ionic Docs - Ionic Documentation. (n.d.). Retrieved September 22, 2023, from <https://ionic.io/docs>

iso25000. (n.d.). *ISO 25010*. Retrieved June 27, 2024, from <https://iso25000.com/index.php/normas-iso-25000/iso-25010>

Leader in Cybersecurity Protection & Software for the Modern Enterprises - Palo Alto Networks. (n.d.). Retrieved September 23, 2023, from <https://www.paloaltonetworks.com/>

Murray, N., Coury, F., Lerner, A., & Taborda, C. (2020). *ng-book The Complete Guide to Angular*.

OWASP API Security Project | OWASP Foundation. (2023). <https://owasp.org/www-project-api-security/>

OWASP Mobile Top 10 | OWASP Foundation. (2023). <https://owasp.org/www-project-mobile-top-10/>

OWASP Top Ten | OWASP Foundation. (2023). <https://owasp.org/www-project-top-ten/>

Perez, S. (2022). *How the mobile app ecosystem adapted to the COVID-19 pandemic in 2021 | TechCrunch*. <https://techcrunch.com/2022/01/10/how-the-mobile-app-ecosystem-adapted-to-the-covid-19-pandemic-in-2021/?guccounter=1>

Pipes • Overview • Angular. (n.d.). Retrieved June 25, 2024, from <https://angular.dev/guide/pipes>

Postman documentation overview | Postman Learning Center. (n.d.). Retrieved June 30, 2024, from <https://learning.postman.com/docs/introduction/overview/>

Presidencia del Consejo de Ministros. (n.d.). *Política Nacional de Transformación Digital al 2030*.

PrimeFlex. (n.d.). Retrieved September 22, 2023, from <https://www.primefaces.org/primeflex/>

Q1 Data: The Impact of COVID-19 on Daily App Usage - CleverTap. (n.d.). Retrieved September 23, 2023, from <https://clevertap.com/blog/q1-data-impact-of-covid-19/>

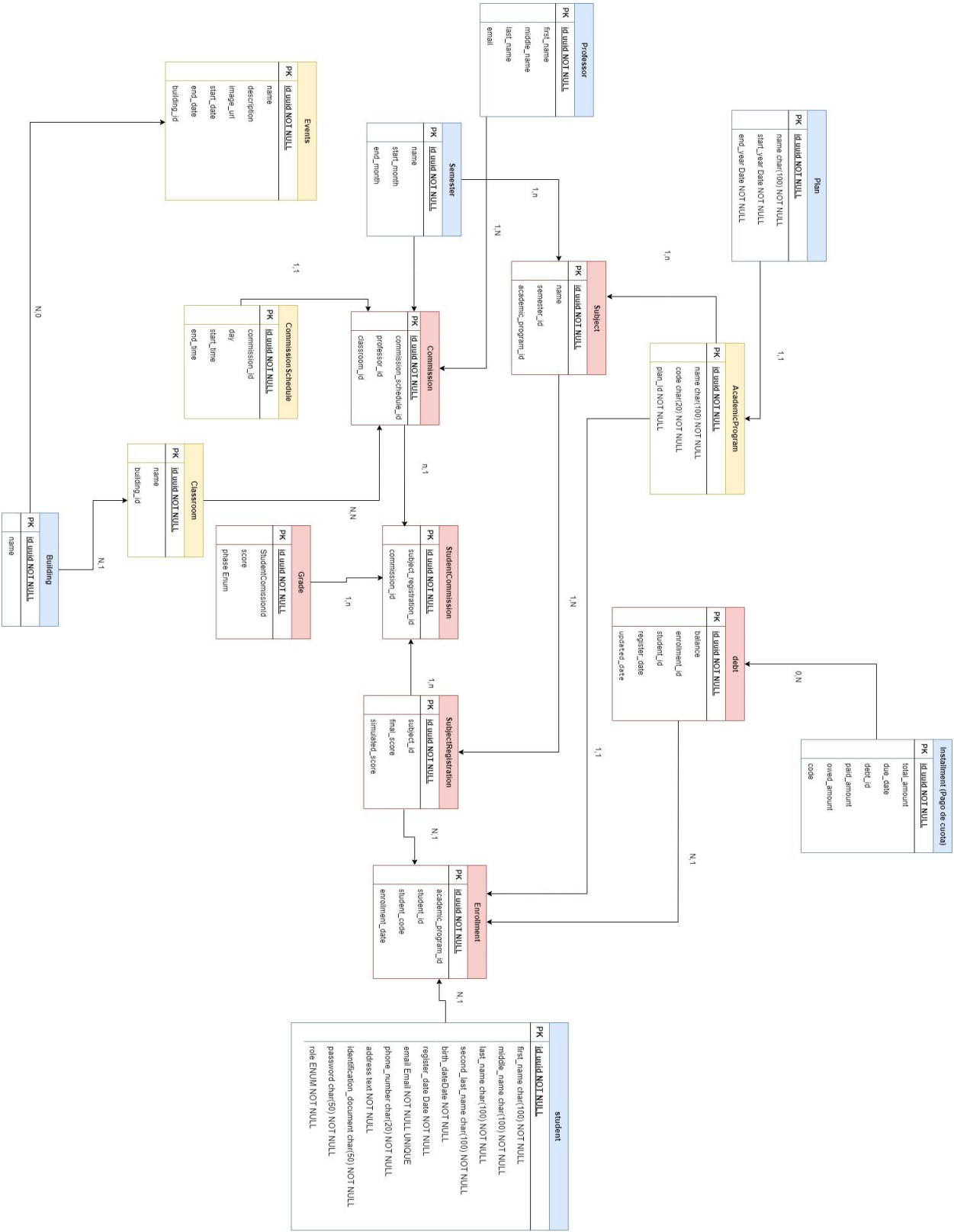
¿Qué es un patrón de diseño? (n.d.). Retrieved March 31, 2024, from <https://refactoring.guru/es/design-patterns/what-is-pattern>

Sandoval, P. (n.d.). *SPablo2191/ucsm-mobile*. Retrieved June 28, 2024, from <https://github.com/SPablo2191/ucsm-mobile>

- Sandoval, P. (2024). *ucsm-mobile-api/docs/setup.md at main · SPablo2191/ucsm-mobile-api*.
<https://github.com/SPablo2191/ucsm-mobile-api/blob/main/docs/setup.md>
- Slade, L. (2020). *The Agents of Transformation Report 2020: COVID-19 Special Edition - Application Performance Monitoring Blog | AppDynamics*. <https://www.appdynamics.com/blog/news/agents-of-transformation-report-2020/>
- Sommerville, I., Campos Olguín, V., & Fuenlabrada Velázquez, S. (2011). *Ingeniería de software*. Pearson Educación de México.
- Stack Overflow Contributors. (2019). *django-rest-framework*.
<https://github.com/YeasirArafatRatul/DjangoBooks/blob/master/django-rest-framework.pdf>
- The PostgreSQL Global Development Group. (n.d.). *PostgreSQL: The world's most advanced open source database*. Retrieved June 25, 2024, from <https://www.postgresql.org/>
- UCSM inicia el Año Académico 2020 utilizando aula virtual – Universidad Católica de Santa María. (n.d.). Retrieved September 23, 2023, from <https://www.ucsm.edu.pe/ucsm-inicia-el-ano-academico-2020-utilizando-aula-virtual/>
- Universidad Católica de Santa María – UCSM. (n.d.). Retrieved September 23, 2023, from <https://www.ucsm.edu.pe/>
- Vercel: *Build and deploy the best web experiences with the Frontend Cloud*. (n.d.). Retrieved June 29, 2024, from <https://vercel.com/>

Anexo 1:

Diagrama de Entidad-Relación (DER) de la base de datos para la aplicación móvil:



Anexo 2:

Tablas de casos de prueba de API REST:

Caso de Prueba	Descripción	Datos de Entrada	Resultado Esperado
1.1	Ingreso exitoso con credenciales válidas	{ "identification_document": "12345678", "password": "correct_password" }	Respuesta HTTP 200, Token de sesión válido retornado.
1.2	Intento de inicio de sesión con credenciales inválidas	{ "identification_document": "12345678", "password": "wrong_password" }	Respuesta HTTP 401, Mensaje de error de credenciales inválidas.
1.3	Intento de inicio de sesión con documento de identificación inválido	{ "identification_document": "invalid_id", "password": "correct_password" }	Respuesta HTTP 404, Mensaje de error indicando que el usuario no existe.
1.4	Intento de inicio de sesión sin proporcionar una contraseña	{ "identification_document": "12345678", "password": "" }	Respuesta HTTP 400, Mensaje de error indicando que la contraseña es requerida.

Caso de Prueba	Descripción	Datos de Entrada	Resultado Esperado
2.1	Recuperar deuda con un <code>enrollment_id</code> válido	?enrollment_id=valid_uuid	Respuesta HTTP 200, Detalles de la deuda y cuotas asociadas para el curso.
2.2	Recuperar deuda con un <code>enrollment_id</code> inválido	?enrollment_id=invalid_uuid	Respuesta HTTP 404, Mensaje de error indicando que el curso no existe.
2.3	Recuperar deuda sin un <code>enrollment_id</code> proporcionado	?enrollment_id=	Respuesta HTTP 400, Mensaje de error indicando que el <code>enrollment_id</code> es requerido.
2.4	Recuperar deuda sin token de sesión válido	?enrollment_id=valid_uuid (sin header de autorización)	Respuesta HTTP 401, Mensaje de error indicando que se requiere autenticación.

Caso de Prueba	Descripción	Datos de Entrada	Resultado Esperado
3.1	Obtener listado de cursos de un estudiante con <code>dni</code> válido	?identification_document=12345678	Respuesta HTTP 200, Lista de cursos en los que el estudiante está inscrito.
3.2	Obtener listado de cursos con <code>dni</code> inválido	?identification_document=invalid_dni	Respuesta HTTP 404, Mensaje de error indicando que el estudiante no existe.
3.3	Obtener detalles de un curso con un <code>id</code> válido	/api/enrollments/valid_uuid	Respuesta HTTP 200, Información detallada del curso.
3.4	Obtener detalles de un curso con un <code>id</code> inválido	/api/enrollments/invalid_uuid	Respuesta HTTP 404, Mensaje de error indicando que el curso no existe.
3.5	Obtener listado de cursos sin <code>dni</code> proporcionado	?identification_document=	Respuesta HTTP 400, Mensaje de error indicando que el <code>dni</code> es requerido.

Caso de Prueba	Descripción	Datos de Entrada	Resultado Esperado
4.1	Obtener listado de eventos	/api/events	Respuesta HTTP 200, Lista de eventos disponibles.
4.2	Obtener detalles de un evento con un <code>id</code> válido	/api/events/valid_event_id	Respuesta HTTP 200, Detalles del evento.
4.3	Obtener detalles de un evento con un <code>id</code> inválido	/api/events/invalid_event_id	Respuesta HTTP 404, Mensaje de error indicando que el evento no existe.
4.4	Obtener listado de eventos con un token de sesión no válido	/api/events (sin header de autorización válido)	Respuesta HTTP 401, Mensaje de error indicando que se requiere autenticación.

Caso de Prueba	Descripción	Datos de Entrada	Resultado Esperado
5.1	Obtener lista de semestres disponibles	/api/semesters	Respuesta HTTP 200, Lista de todos los semestres disponibles.
5.2	Obtener detalles de un semestre con un <code>id</code> válido	/api/semesters/valid_semester_id	Respuesta HTTP 200, Información detallada del semestre.
5.3	Obtener detalles de un semestre con un <code>id</code> inválido	/api/semesters/invalid_semester_id	Respuesta HTTP 404, Mensaje de error indicando que el semestre no existe.
5.4	Obtener lista de semestres con un token de sesión no válido	/api/semesters (sin header de autorización válido)	Respuesta HTTP 401, Mensaje de error indicando que se requiere autenticación.

Caso de Prueba	Descripción	Datos de Entrada	Resultado Esperado
6.1	Obtener perfil de un estudiante con un <code>id</code> válido	/api/students/valid_student_id	Respuesta HTTP 200, Detalles del perfil del estudiante.
6.2	Obtener perfil de un estudiante con un <code>id</code> inválido	/api/students/invalid_student_id	Respuesta HTTP 404, Mensaje de error indicando que el estudiante no existe.
6.3	Obtener perfil del estudiante con un token de sesión no válido	/api/students/valid_student_id (sin header de autorización válido)	Respuesta HTTP 401, Mensaje de error indicando que se requiere autenticación.

Caso de Prueba	Descripción	Datos de Entrada	Resultado Esperado
7.1	Obtener materias inscritas con <code>enrollment_id</code> válido	?enrollment_id=valid_uuid	Respuesta HTTP 200, Lista de materias en las que el estudiante está inscrito.
7.2	Obtener materias inscritas con <code>enrollment_id</code> inválido	?enrollment_id=invalid_uuid	Respuesta HTTP 404, Mensaje de error indicando que el curso no existe.
7.3	Obtener detalles de una materia con un <code>id</code> válido	/api/subject-registrations/valid_subject_id	Respuesta HTTP 200, Información detallada de la materia.
7.4	Obtener detalles de una materia con un <code>id</code> inválido	/api/subject-registrations/invalid_subject_id	Respuesta HTTP 404, Mensaje de error indicando que la materia no existe.
7.5	Obtener materias restantes con un <code>enrollment_id</code> válido	?enrollment_id=valid_uuid	Respuesta HTTP 200, Lista de materias