
**UNIVERSIDAD EAN**

# Desarrollo de una aplicación móvil para la gestión y acceso a la información de actividades extracurriculares para instituciones de educación superior

**Lizette Nicolle Hernández Arévalo**
**Laura valentina Pérez Velásquez**

**Docente:**
**Johanna Karina Solano Meza**

**Facultad de Ingeniería**
**BOGOTÁ D.C., 2024.**

## Índice de Contenido

## Índice de Figuras



## Índice de Tablas

**Resumen ejecutivo**

Las instituciones de educación superior enfrentan un desafío debido a la falta de un sistema centralizado para la difusión de los eventos extracurriculares. A pesar de contar con aplicaciones institucionales y el correo electrónico, estos métodos han sido poco eficaces. La saturación del correo institucional y la integración de múltiples funciones en las aplicaciones existentes dificultan en la visibilidad de los eventos, resultando en desinformación y baja participación.

Para abordar esta deficiencia, se desarrollará una aplicación móvil dedicada exclusivamente a la gestión de eventos extracurriculares. Esta aplicación proporcionará una plataforma centralizada que permite a los estudiantes acceder fácilmente a información acerca de los eventos. Además, incluirá funciones clave como el registro y autenticación mediante credenciales
universitarias, una lista actualizada de eventos que estará disponible en tiempo real, permitiendo a los estudiantes visualizar cambios inmediatos en la programación, así como una interfaz de búsqueda avanzada para facilitar el acceso a la información.

La implementación de esta aplicación tiene como objetivo mejorar la comunicación entre organizadores y participantes. Al centralizar la información y facilitar el acceso, se espera incrementar la participación estudiantil y se fortalecer el vínculo entre estudiantes, profesores y la institución de educación superior.

*Palabras clave: Instituciones de Educación Superior, eventos extracurriculares,*

*difusión de eventos, aplicación móvil.*

## Introducción

En las instituciones de educación superior, los eventos extracurriculares son componentes esenciales en la formación integral de los estudiantes. Estos eventos incluyen talleres, actividades culturales y lúdicas, que son fundamentales para el desarrollo de habilidades blandas, como el trabajo en equipo, la comunicación efectiva y el liderazgo. Estas experiencias complementan la educación académica, permitiendo a los estudiantes explorar sus intereses personales y fortalecer competencias que son vitales para su vida profesional.

Además, los eventos extracurriculares facilitan el desarrollo de competencias como la resolución de problemas y la adaptabilidad, proporcionando a los estudiantes experiencias únicas que enriquecen su experiencia universitaria. Estas actividades permiten a los estudiantes explorar intereses personales y profesionales que pueden no estar incluidos en el plan de estudios académico y ofrecen una interacción e intercambio de ideas con compañeros y profesores. Asimismo, permiten a los estudiantes construir una red de contactos para su futuro.

A pesar de la importancia de estos eventos, la gestión y difusión de la información relacionada presenta algunos desafíos significativos. Las instituciones de educación superior suelen depender del correo electrónico y aplicaciones institucionales para comunicar los eventos. Sin embargo, estos métodos no garantizan una comunicación efectiva. El correo electrónico, por ejemplo, se ve saturado debido a la gran cantidad de mensajes diarios que los estudiantes reciben, dificultando la visibilidad de los

anuncios importantes sobre eventos extracurriculares.

Además, la falta de priorización en los correos electrónicos agrava el problema, haciendo que la información relevante sobre los eventos se pierda o pase desapercibido. Por otro lado, las aplicaciones institucionales suelen integrar diferentes funcionalidades, lo que puede diluir el enfoque de la sección de eventos y reducir su visibilidad. La dispersión de la información en diferentes canales de comunicación contribuye a la falta de accesibilidad, impidiendo que los estudiantes aprovechen estas oportunidades. Esto reduce la participación estudiantil y limita el impacto potencial de estas experiencias en la formación integral de los estudiantes.

Es esencial reconocer que los espacios extracurriculares son fundamentales para una formación integral de los estudiantes. Estos espacios no solo enriquecen la formación académica al ofrecer experiencias que van más allá del aula de clase, sino que también motivan a los estudiantes y facilitan una comprensión más profunda de los temas. Según Chaparro Africano (2022) “Es, entonces, indudable la importancia de los espacios extracurriculares en el proceso de formación de los estudiantes, considerando que ellos aportan en termino de potenciación motivacional, de fluidez en los contenidos relacionados con la experiencia cotidiana y de activación de un dialogo y negociación de saberes” (p.39). La implementación de un sistema centralizado que facilite el acceso a estos espacios puede mejorar la experiencia educativa.

Como primera etapa en la solución de esta problemática, se ha desarrollado una aplicación móvil dedicada que está específicamente dedicada a la gestión de eventos

extracurriculares. Esta aplicación ofrece una plataforma centralizada y accesible para la difusión de eventos, superando las limitaciones de los métodos actuales.

En segunda instancia, esta aplicación busca mejorar la comunicación entre organizadores y estudiantes al centralizar la información y proporcionar recordatorios puntuales y accesibles. Esto se espera que incremente la participación estudiantil y enriquezca la experiencia universitaria.

Finalmente, la implementación de esta aplicación no solo facilita el acceso a oportunidades extracurriculares, sino que también contribuye al desarrollo integral de los estudiantes y al fortalecimiento de la comunidad académica, ayudando a crear una red sólida de contactos y fomentando una educación más integral y accesible para todos los miembros de la comunidad educativa.

**Objetivo general**

Desarrollar un aplicativo móvil utilizando el lenguaje de programación Python que permita acceder a la información de eventos extracurriculares en instituciones de educación superior.

**Objetivos específicos**

1. Realizar un análisis exploratorio de la gestión y comunicación de información referente a actividades extracurriculares en las instituciones de educación superior.
2. Prototipar una base de datos local simulada que replique el funcionamiento de una solución en la nube, utilizando Firebase.
3. Desarrollar una versión piloto del aplicativo, incluyendo la interfaz del usuario y las diferentes funcionalidades para su prueba y evaluación.

**Definición del problema**

En el contexto de la gestión y la difusión de información relacionada con eventos extracurriculares en instituciones de educación superior, las instituciones enfrentan un desafío significativo debido a la falta de un sistema centralizado. Actualmente, la información sobre los eventos se difunde a través de diversos canales, como el correo y aplicaciones institucionales. Sin embargo, estos métodos de difusión han demostrado ser poco eficaces debido a que los eventos no se muestran en tiempo real.

El uso del correo electrónico se enfrenta a un problema de saturación, debido a la gran cantidad de correos electrónicos que los estudiantes reciben diariamente, lo que provoca que los mensajes importantes relacionados con actividades extracurriculares se pierdan entre las demás comunicaciones. Como resultado, muchos estudiantes no están al tanto de los eventos que podrían enriquecer su experiencia universitaria.

Como menciona Almenara (2007) “la incorporación de las TIC a las instituciones educativas permitirá nuevas formas de acceder, generar, y transmitir información y conocimientos, lo que nos abrirá las puertas para poder flexibilizar, transformar, cambiar, extender” (p.8). Esta flexibilidad se manifiesta no solo en el aspecto temporal y espacial en cuanto a la interacción y recepción de la información, sino también en la posibilidad de adaptar y personalizar las estrategias y técnicas utilizadas en la formación educativa. Por lo tanto, el desarrollo de una aplicación móvil centralizada, que aproveche las ventajas de las TIC para mejorar la difusión y el acceso a la

información de actividades extracurriculares, se presenta como una solución necesaria para superar estas limitaciones actuales.

Por otro lado, aunque algunas instituciones de educación superior disponen de una aplicación institucional que incluye una sección para eventos, esta no logra captar de manera efectiva la atención de los estudiantes. Al integrar múltiples funciones, como la consulta de notas, el horario y/o el carnet estudiantil, lo que dificulta la visibilidad de los eventos extracurriculares. Los estudiantes tienden a utilizar la aplicación únicamente para fines específicos y rara vez exploran otras secciones, lo que minimiza el impacto de la información sobre los eventos.

Este enfoque fragmentado de comunicación contribuye a la baja participación en las actividades extracurriculares, afectando negativamente la integración y el desarrollo personal de los estudiantes. Además, la falta de centralización de la información limita la inclusión de nuevos miembros en la comunidad universitaria. Los estudiantes nuevos o los que no están familiarizados con los diversos canales de comunicación pueden encontrar difícil mantenerse al día con los eventos, lo que reduce su capacidad para integrarse en la comunidad. Este problema no solo afecta a los estudiantes, sino que también afecta en la capacidad de la institución para demostrar su compromiso con el bienestar de sus estudiantes y la inclusión de nuevos participantes.

Un estudio de Bayerlein et al. (2021) demostró como el uso de tecnologías digitales en

la educación universitaria puede aumentar la participación de los estudiantes y mejorar su compromiso. Estas herramientas no solo facilitan la comunicación, sino que también hacen la información más accesible, creando un entorno donde los estudiantes se sienten más conectados y motivados. Al aplicar estos hallazgos al ámbito de los eventos extracurriculares, una aplicación móvil podría cumplir un papel similar, brindando mayor visibilidad a las activades y alentando a más estudiantes a participar. De este modo, la tecnología se convierte en una pieza fundamental para mejorar la forma en que se comunican y gestionan estos eventos, facilitando una experiencia más enriquecedora para los estudiantes.

Considerando los desafíos actuales de dispersión de la información y la baja visibilidad de los eventos, es esencial explorar soluciones innovadoras que faciliten el acceso y la participación de los estudiantes. Por lo tanto, surge la siguiente pregunta: ¿Cómo pueden las instituciones de educación superior mejorar la difusión y accesibilidad de la información sobre actividades extracurriculares a través de una herramienta tecnológica, para incrementar la participación estudiantil y fomentar una mayor integración en la comunidad universitaria?

**Justificación**

En el ámbito de la educación superior, las actividades extracurriculares desempeñan un papel esencial en la formación integral de los estudiantes, complementando su educación académica con oportunidades para el desarrollo de habilidades blandas, la exploración de intereses personales y la integración social. Estas actividades no solo enriquecen la experiencia universitaria de los estudiantes, sino que también fomentan habilidades críticas como el trabajo en equipo, la comunicación efectiva y el liderazgo.

Estas habilidades son fundamentales para el desarrollo integral de los estudiantes y su preparación para el mundo laboral. Además, los estudiantes que participan en diferentes actividades extracurriculares aprenden a administrar su tiempo de manera que no interfiera con sus estudios, lo cual les permite desarrollar una disciplina más sólida y una mejor capacidad de organización.

Complementando lo anterior, Almalki et al. (2017) mencionan que se encontró que la capacidad de liderazgo, el pensamiento crítico, la autoconfianza social y las habilidades de resolución de conflictos eran mayores en los estudiantes que se ofrecían como voluntarios. La participación voluntaria de los estudiantes universitarios puede afectar positivamente los resultados educativos, como la obtención de títulos superiores, lo cual se debe a que, al involucrarse en actividades extracurriculares, los estudiantes aumentan su motivación y autoconfianza, lo cual contribuye a una experiencia educativa más exitosa.

Además, es importante resaltar que la participación de los estudiantes en estas actividades apoya los procesos de acreditación de alta calidad de las universidades. De esta forma, las actividades extracurriculares no solo potencian habilidades personales, sino que también contribuyen a fortalecer el perfil institucional de las universidades.

Normalmente, estas actividades suelen difundirse a través de carteleras, redes sociales o mediante el correo institucional, sin embargo, estos canales no siempre son eficientes para captar la atención de los estudiantes, ya que no todos están pendientes de estas comunicaciones o no les prestan la atención que merecen. Por esa razón, se propone el desarrollo de una aplicación móvil para la gestión y acceso a la información de actividades extracurriculares.

Aunque diferentes instituciones de educación superior cuentan con una aplicación en la cual se incluye el carnet digital de los estudiantes y algunas actividades, se ha detectado que muchos estudiantes no están muy pendientes de esta aplicación. Para abordar este problema, la nueva aplicación incluirá una lista de eventos programados durante el mes, con información detallada como la descripción de la actividad, sala o lugar donde se llevará a cabo, y el estado del evento, ya sea en curso o finalizado, junto con una cuenta regresiva para los eventos próximos.

En el desarrollo de aplicaciones móviles, especialmente aquellas orientadas a la gestión de información en tiempo real, los usuarios, tales como estudiantes y docentes de instituciones de educación superior, demandan un acceso inmediato y preciso a la

información sobre actividades extracurriculares, lo cual resulta esencial para la planificación académica y la gestión de su tiempo (Díaz y Díaz, 2018). Tal como lo señalan Enriquez Ayala y Villagómez Bardellini (2021) los sistemas en tiempo real deben interactuar con el entorno respondiendo a los estímulos en un tiempo limitado, ya que cualquier retraso en el procesamiento de la información podría comprometer la experiencia del usuario y, en este caso, la correcta organización de las actividades.

Por lo tanto, el desarrollo de esta aplicación móvil no solo debe enfocarse en la correcta entrega de la información, sino también en garantizar que dicha entrega ocurra dentro de un tiempo que permita la toma de decisiones en un contexto académico dinámico. La correcta gestión de estas interacciones en tiempo real no solo mejora la usabilidad y funcionalidad de la aplicación, sino que también asegura que se mantenga operativa y confiable incluso en situaciones de alta demanda o fallos temporales, cumpliendo con los estándares que los usuarios y las instituciones.

Estas mejoras pueden marcar la diferencia, ya que permitirá a los estudiantes estar más informados y motivados a participar en las diversas actividades planeadas por las instituciones, lo cual podría resultar en una mayor asistencia y en la creación de un espacio más agradable y amigable para toda la comunidad universitaria.

Mackaway y Chalkley (2021) destacan la importancia de hacer que los estudiantes se sientan incluidos y participen en sus programas educativos. En el caso de los eventos extracurriculares, una aplicación móvil podría ser la herramienta perfecta para mejorar

la comunicación entre las instituciones de educación superior y sus estudiantes. Al facilitar el acceso a la información sobre actividades y eventos, esta aplicación ayudaría aumentar la visibilidad y la participación, contribuyendo con el compromiso de los estudiantes y enriqueciendo su experiencia universitaria.

**Análisis de requerimientos**

Para el desarrollo de este proyecto, se empleará el lenguaje de programación Python, junto con frameworks adecuados para la construcción de la aplicación. Asimismo, es fundamental comprender el funcionamiento de las bases de datos NoSQL, en especial con Firebase, que servirá como herramienta principal para el almacenamiento de la información de manera organizada, segura y en tiempo real. La aplicación permitirá que los estudiantes se registren e inicien sesión de manera segura mediante sus credenciales universitarias, protegiendo así sus datos personales.

Una vez autenticados, los usuarios tendrán acceso a una lista actualizada de eventos extracurriculares organizados por la institución. El desarrollo del proyecto se realizará utilizando Visual Studio Code como editor, seleccionado por su compatibilidad con múltiples lenguajes y su integración con herramientas de control de versiones. Para la gestión del código y la colaboración en equipo, se empleará un repositorio en GitHub, lo cual permitirá el seguimiento de cambios y la colaboración en tiempo real.

Se implementará una función de búsqueda avanzada que permita filtrar eventos por palabras clave, fechas o categorías, y que permita a los usuarios visualizar los eventos en los que se han inscrito, facilitando la organización de su tiempo.

El diseño de la aplicación será intuitivo y fácil de utilizar, con una interfaz atractiva y una navegación fluida. Se desarrollarán wireframes y prototipos para evaluar el diseño antes de la implementación final. En términos de seguridad, se adoptarán medidas para

garantizar la autenticación y protección de la información, incluyendo la estructura de datos en Firebase y la creación de un modelo de datos adecuado.

La aplicación deberá ofrecer un rendimiento óptimo, respondiendo con rapidez a las acciones de los usuarios y siendo escalable para soportar un gran número de usuarios simultáneamente. Será compatible con los principales sistemas operativos móviles y funcionará correctamente en una variedad de dispositivos. Además, para su funcionamiento adecuado, las universidades deberán contar con una infraestructura tecnológica que garantice el almacenamiento y procesamiento de datos de manera segura y eficiente.

Esta infraestructura incluye capacidad de almacenamiento y soporte para bases de datos escalables que gestionen el registro de estudiantes y eventos. Asimismo, se requiere un equipo de soporte técnico que mantenga y actualice continuamente la plataforma, garantizando su rendimiento y seguridad. Será fundamental desarrollar políticas de privacidad y protección de datos conforme a la normativa vigente de seguridad de la información de los usuarios, además de asegurar la integración con el sistema de la universidad. Asimismo, se requerirá la colaboración del área de bienestar universitario, que será responsable de distribuir las actividades en la aplicación.

**Marco de referencia**

El presente marco teórico tiene como objetivo brindar una visión integral de los conceptos clave relacionados con el desarrollo de una aplicación móvil para la gestión y acceso a la información de actividades extracurriculares en instituciones de educación superior. Según Bond et al. (2020), la tecnología digital se ha convertido en un elemento fundamental en la educación superior, influyendo significativamente en la participación conductual, afectiva y cognitiva de los estudiantes.

Este impacto se complementa con la teoría organizacional, la cual, según Manning (2017), permite entender las dinámicas en los campus universitarios a través de diversos modelos y enfoques. A su vez, el aprendizaje combinado ha demostrado ser una estrategia efectiva para ayudar a los estudiantes de educación superior, mejorando la experiencia de aprendizaje mediante la integración de recursos en línea con métodos tradicionales (Serrano et al., 2019).

En este contexto, los dispositivos móviles juegan un rol esencial, ya que facilitan la planificación de la investigación, la adquisición de recursos y la gestión de tareas académicas (Ambriz, 2011), algo que también ha sido destacado por la UNESCO en sus investigaciones. Sin embargo, muchas instituciones aún enfrentan desafíos en la gestión de actividades extracurriculares, ya que dependen de procesos manuales que ralentizan las operaciones y aumentan la carga administrativa, lo que puede generar errores de registro.

Alias Mendoza y Santos del Carpio (2013) mostraron que la implementación de soluciones tecnológicas, como un sistema automatizado en el Instituto Técnico de Tuxtla Gutiérrez, puede aumentar significativamente la eficiencia en la gestión de eventos extracurriculares. Además, Torres y Silva (2017) subrayan que estos eventos permiten a los estudiantes aplicar y reforzar habilidades aprendidas en el aula, lo que resalta su importancia en el proceso formativo de los estudiantes.

***Actividades extracurriculares***

Las actividades extracurriculares son aquellas ofrecidas por las instituciones educativas, pero no están incluidas directamente en los programas académicos. Sin embargo, Porto y Gardey (2023) mencionan que desempeñan un papel crucial en el desarrollo integral de los estudiantes, ya que permiten y contribuyen al proceso de enseñanza y aprendizaje de forma divertida, creativa o entretenida.

Balseca Pallasco (2017) menciona que existen diferentes tipos de actividades extracurriculares, los cuales incluyen actividades artísticas, como el canto, la danza, el dibujo y la fotografía; además de actividades deportivas como el fútbol, tenis, tae kwon do. Estas actividades ofrecen a los estudiantes oportunidades de desarrollo en múltiples áreas, fomentando la creatividad y el bienestar físico y mental.

Además, Balseca Pallasco (2017) menciona el impacto en la educación, refiriéndose a las actividades de ocio como una valiosa adición al conjunto de actividades. El tiempo libre se utiliza para la formación integral del estudiante, tanto como ser individual como

social, desarrollando su potencial artístico, deportivo, cultural y educativo. Los niños las desarrollan desde el principio y logran generar expectativas claras sobre lo que quieren en los próximos años.

Por otro lado, Morrison (2023) destaca que participar en actividades extracurriculares ofrece beneficios para el desarrollo personal y profesional de los estudiantes. Estas actividades contribuyen a adquirir habilidades blandas como el liderazgo, el trabajo en equipo y la comunicación, fundamentales en el ámbito laboral.

***Desarrollo de aplicaciones móvil***

Introducción al desarrollo de aplicaciones móviles (IBM., s. f.) menciona que el desarrollo de aplicaciones móviles se refiere al proceso de creación de software para teléfonos inteligentes, tabletas y asistentes digitales. Este tipo de aplicaciones generalmente se desarrolla para sistemas operativos específicos como Android y iOS. Por otro lado, los lenguajes de programación más populares en el desarrollo de aplicaciones móviles son Java y Kotlin (Platzi, n.d.) ya que se determinan por tener un sólido historial y las características que ofrecen; sin embargo, cada uno depende de las necesidades específicas y la experiencia del desarrollador.

Además, los frameworks permiten completar proyectos de programación de manera rápida y eficiente, utilizando menos tiempo y generando un código más limpio y consistente. Simplifican el proceso de desarrollo porque se pueden reutilizar herramientas, ya que se tiene un "esqueleto" que se puede utilizar. Escribir código o

desarrollar aplicaciones de forma más sencilla puede ayudar a organizar y controlar mejor todo el código creado para su uso futuro (UNIR FP, n.d.). Por otro lado, las tendencias actuales en el desarrollo de aplicaciones móviles están en constante evolución. Nandaniya (2024) señala que entre las tendencias actuales se encuentran la centralización de portales, el aumento de plataformas de código abierto y la creciente integración de servicios y aplicaciones.

Además, el desarrollo de la aplicación se basó en el uso de Python como lenguaje de programación principal y Kivy como framework para la creación de interfaces gráficas de usuario (GUI). En donde Python es un lenguaje de programación de alto nivel, conocido por su sintaxis clara y legibilidad, lo cual es una buena opción para proyectos de desarrollo ágil. A lo largo de los años, Python ha ganado una enorme popularidad en diversas áreas del desarrollo de software, incluidos el desarrollo web, la ciencia de datos, la automatización y el desarrollo de aplicaciones de escritorio.

Según Van Rossum (2020), creador de Python, uno de los principales beneficios del lenguaje es su enfoque en la simplicidad y la legibilidad, lo que facilita la colaboración entre desarrolladores y la mantenibilidad del código. Además, Python es ampliamente utilizado en el desarrollo de aplicaciones debido a su rica colección de bibliotecas y frameworks, que permiten a los desarrolladores implementar funcionalidades complejas de manera eficiente.

Por otro lado, Kivy es un framework open-source para Python, diseñado para el

desarrollo de aplicaciones con interfaces gráficas interactivas, particularmente aquellas que requieren soporte multitáctil. Kivy se distingue por su capacidad de permitir el desarrollo de aplicaciones multiplataforma, es decir, aplicaciones que pueden ejecutarse en sistemas operativos como Windows, macOS, Linux, iOS y Android utilizando un único código base (Kivy, 2023). Esto hace que Kivy sea una herramienta poderosa para proyectos que buscan compatibilidad con múltiples plataformas sin necesidad de escribir código específico para cada sistema operativo.

Además, Kivy ofrece una gran variedad de widgets y componentes predefinidos que facilitan el diseño de interfaces complejas sin necesidad de desarrollar elementos desde cero, también permite integrar fácilmente otras bibliotecas de Python, lo que posibilita agregar funcionalidades avanzadas como el acceso a bases de datos, procesamiento de imágenes o integración con APIs externas.

Para que una aplicación móvil tenga éxito, es fundamental considerar factores clave como la experiencia del usuario (UX), la funcionalidad intuitiva, el rendimiento eficiente, y la seguridad robusta. (Lee et al., 2017). Además, el diseño de interfaz de usuario (UI) es un proceso utilizado por los diseñadores para crear interfaces en software, centrándose en la apariencia o el estilo, donde el objetivo es crear una interfaz que los usuarios encuentren fácil de usar y agradable (Debernardi, 2021).

En cuanto a los principios de diseño de la interfaz de usuario (UI), estos incluyen la consistencia, simplicidad, visibilidad del estado del sistema y retroalimentación. Johnson

(2020) destaca que los principios ayudan a crear interfaces que sean intuitivas y que tengan usabilidad, mejorando así la experiencia del usuario. Además, Levy (2015) menciona que la experiencia del usuario (UX) se centra en la satisfacción del usuario final con el diseño del producto, el cual debe ser intuitivo, accesible y atractivo.

La usabilidad y accesibilidad son aspectos clave en el desarrollo de UI. Como menciona Krug (2014), estos factores son esenciales, ya que aseguran que los productos digitales sean fáciles de usar y accesibles a todo el público, generando más audiencia en la aplicación.

***Base de datos***

Las bases de datos es una recopilación organizada para almacenar, gestionar y recuperar información de forma eficiente. Según Oracle (2020), los datos se almacenan en tablas, las cuales contienen filas y columnas que organizan la información de manera lógica, mejorando así el procesamiento de datos y la eficiencia de las consultas. A partir de esta definición, se explicarán los modelos relacionales, en los que la información se organiza en tablas con filas y columnas, donde cada tabla representa una entidad y las relaciones entre ellas se definen mediante claves.

Por otro lado, los modelos no relacionales permiten almacenar datos en estructuras más flexibles y no siguen el modelo tradicional de base de datos relacionales. Estas están diseñadas para manejar grandes volúmenes de datos, como documentos, grafos, pares clave-valor o columnas anchas. Khan et al. (2023) mencionan que las bases de datos

relacionales son mejores para transacciones complejas, mientras que las no relacionales son más adecuadas para datos estructurados y escalabilidad.

En cuanto a las estrategias de almacenamiento de datos en aplicaciones móviles, Santos et al. (2023) señalan que el almacenamiento de datos puede ser local (SQLite) o remoto (servidores backend), dependiendo del tipo de aplicación y del uso de los datos. Además, Dimitrios (2022) menciona que la seguridad en el almacenamiento de datos en aplicaciones móviles se centra en la encriptación de datos, autenticación segura y protección contra accesos no garantizados.

En el desarrollo del proyecto, se utilizó Firebase como solución para la gestión de bases de datos en tiempo real. Firebase es una plataforma proporcionada por Google que facilita el almacenamiento y la sincronización de datos entre usuarios de manera eficiente y escalable (Firebase, s. f.-b). A través de su servicio Firebase Realtime Database, que es una base de datos NoSQL basada en la nube, se logró almacenar y sincronizar datos en tiempo real entre los usuarios. Esta capacidad permitió la actualización instantánea de la información entre usuarios, mejorando la experiencia interactiva de la aplicación.

Gracias a su integración sencilla con plataformas móviles y web, así como su soporte para manejar operaciones en tiempo real, Firebase Realtime Database se presentó como la opción ideal para el almacenamiento y sincronización de datos del proyecto. Además, su estructura de datos en JSON, junto con las reglas de seguridad configurables, garantizó la integridad y la privacidad de la información en todo momento

(Firebase Realtime Database, s. f.).

Sin embargo, es importante considerar las pruebas de software, como lo mencionan Spillner y Linz (2021), donde deben incluir pruebas de funcionalidad, usabilidad, rendimiento y seguridad, garantizando así la calidad del producto. De acuerdo con Chiu (2015), este tipo de prueba se enfoca en verificar la correcta implementación de los requisitos del cliente en el sistema. Estas pruebas aseguran que el software funcione como se esperaba y comparan los resultados obtenidos con los resultados esperados. Se realizan desde la perspectiva del usuario y pueden incluir pruebas de módulo, integración y aceptación.

Según Verona Marcos et al. (2016), las pruebas de software están diseñadas para evaluar la velocidad del sistema y la eficiencia de utilización de recursos bajo ciertas condiciones. Además, permiten probar características de calidad como escalabilidad, confiabilidad y eficiencia de recursos. Diaz Diaz (2014) menciona que las pruebas son el conjunto de actividades diseñadas para identificar errores y debilidades en las aplicaciones web, con el fin de reducir el impacto de los ataques y la pérdida de información importante. El objetivo es garantizar la confidencialidad e integridad de los datos desde la etapa de desarrollo inicial.

***Efectos de la falta de centralización***

En cuanto a las soluciones tecnológicas en el desarrollo de aplicaciones móviles, Singh et al. (2023) explican que se trata de un enfoque integral que abarca el análisis, la

creación de aplicaciones, las pruebas correspondientes, la depuración y el mantenimiento, cubriendo todos los aspectos necesarios para que las aplicaciones funcionen de manera efectiva. Por otro lado, la computación en la nube ofrece un espacio flexible y adaptable en la educación, permitiendo almacenar, organizar y compartir recursos educativos de manera eficiente, según Sarmiento y Guerrero (2021). En cuanto a los modelos de servicios en la nube, Huawei (2023) menciona que se trata de una tecnología que abarca tanto los conceptos básicos como las arquitecturas y aplicaciones comunes.

Según Parast et al. (2022), la seguridad en la nube se enfrenta a desafíos importantes a pesar de tener varios beneficios que ofrece la computación en la nube. Los autores destacan como la virtualización y la multitenencia introducen nuevas vulnerabilidades y presentan los problemas de seguridad del estado actual. Y Abdulsalam y Hedabou (2021) mencionan que las normativas de seguridad en la computación en la nube se centran en abordar los riesgos asociados a la externalización de datos y aplicaciones a entornos de terceros. Estas normativas buscan proteger la privacidad y la integridad de la información mediante enfoques técnicos para enfrentar amenazas y garantizar un entorno seguro en la nube.

Por otro lado, Pan et al. (2021) explica la importancia de la sincronización de los datos en tiempo real en sistemas de logística. La integración de tecnologías como la computación en la nube permite un monitoreo y control efectivos para manejar las variaciones en tiempo real y optimizar el rendimiento del sistema.

En cuanto a las APIs para la gestión de eventos, González Granadillo et al. (2021) mencionan que los sistemas SIEM son esenciales para detectar y responder a ciberataques, ya que tienen una visión de los riesgos y ayudan a reducir costos y tiempos de respuesta mediante la integración con herramientas de big data.

**Análisis de restricciones**

***Ambientales***

Uno de los aspectos más relevantes a considerar en el proyecto son las restricciones ambientales, especialmente en el consumo de energía. Durante el desarrollo de la aplicación, se anticipa un alto consumo energético debido a las múltiples pruebas y simulaciones necesarias para garantizar su correcto funcionamiento. Además, el procesamiento de datos requerirá un almacenamiento considerable y grandes cantidades de energía para operar y enfriar los equipos.

Sin embargo, se espera que el impacto ambiental se vea mitigado, ya que el proyecto se llevará a cabo en la universidad, que utiliza en gran medida energía renovable. Esto contribuirá al uso responsable de los recursos energéticos durante el desarrollo.

***Económicas***

Es crucial verificar las licencias de cualquier software o biblioteca utilizada en el desarrollo de la aplicación, especialmente si esta se comercializa. Esto implica que el presupuesto será una restricción importante al momento de utilizar los recursos para el desarrollo. Sin embargo, en este proyecto no se anticipan grandes costos en licencias de software, ya que se utilizará Visual Studio Code, lo que permite reducir significativamente los costos asociados con el desarrollo. Además, al optar por software de código abierto y herramientas sin costo, se optimiza el uso de los recursos financieros, manteniendo un desarrollo de calidad sin incurrir en gastos adicionales.

***Legales***

Al desarrollar una aplicación móvil para la gestión de eventos extracurriculares en instituciones de educación superior en Colombia, es fundamental considerar diversas restricciones que pueden influir en su implementación. En primer lugar, se debe cumplir con las regulaciones de protección de datos personales establecidas en la Ley 1581 de 2012, que regula la recolección, almacenamiento, uso y tratamiento de los datos sensibles, como nombres o correos electrónicos. Esto implica obtener el consentimiento informado de los usuarios, informándoles sobre la finalidad de la recolección de los datos y garantizando la seguridad de la información recopilada mediante medidas adecuadas.

Además, se deben considerar los derechos de autor y la propiedad intelectual. Si la aplicación incluye contenido visual, como videos o imágenes relacionados con eventos, se debe contar con los permisos necesarios de los titulares de los derechos. En cuanto a las normativas sobre telecomunicaciones y servicios de Internet, La Ley 1341 de 2009 (Ley TIC) regula el uso de tecnologías de la información y las comunicaciones en Colombia, garantizando estándares de accesibilidad, calidad del servicio e interoperabilidad del sistema.

***Salud y seguridad***

Salud y seguridad La aplicación va a estar diseñada con una interfaz intuitiva y fácil de usar, cumpliendo con las pautas de accesibilidad al contenido web (WCAG 2.1). Esto garantiza que la aplicación sea accesible para personas con discapacidad y usable

para todos los usuarios, independientemente de sus capacidades. En términos de ciberseguridad, el Decreto 338 de 2022 establece lineamientos que deben ser considerados ya que ayudarían a proteger la información de los usuarios contra posibles ataques cibernéticos, manteniendo la confidencialidad, integridad y disponibilidad en los datos. Dado que el proyecto no implica riesgos o impactos directos en la salud este aspecto no es aplicable. Dado que el proyecto no implica riesgos o impactos directos en la salud, este aspecto no es aplicable.

***Socioculturales***

Es importante tener la aceptación de los usuarios para alcanzar los objetivos de la aplicación. Por lo tanto, la aplicación debe ser diseñada para ser inclusiva y respetuosa, teniendo en cuanta la diversidad cultural de los usuarios. Sin embargo, no se espera que la aceptación sea inmediata, ya que será necesario realizar campañas de sensibilización y formación para familiarizar a los usuarios con la plataforma y maximizar su uso.

**Metodología para la selección y desarrollo de la solución**

A lo largo del desarrollo de este proyecto, se han explorado varias formas de mejorar la difusión de eventos extracurriculares. Sin embargo, algunas de ellas opciones consideradas no cumplen con las expectativas que se tienen para el sistema que se quiere implementar. Por ejemplo, colocar pantallas de televisión en zonas comunes de la institución podría parecer una buena idea, pero tiene varios problemas.

En primer lugar, no permiten actualizaciones en tiempo real, lo que significa que los estudiantes podrían quedarse desinformados si hay cambios de última hora. Además, tendrían que estar en un lugar específico para visualizar la información, lo que limita su accesibilidad. También hay que considerar los costos asociados, ya que la compra e instalación de estas pantallas requeriría un presupuesto considerable, sin mencionar el alto consumo de energía que implicaría mantenerlas en funcionamiento constante.

Por otro lado, las carteleras físicas en los pasillos también tienen sus inconvenientes. Al igual que las pantallas, no permiten actualizaciones inmediatas, lo que puede generar confusión cuando hay cambios repentinos en los eventos. Además, requieren un mantenimiento constante y solo son visibles para aquellos que asisten de manera presencial. También es importante considerar que este método implica un uso continuo de papel y otros materiales de impresión, lo cual no contribuye para el cuidado del medio ambiente.

Si bien una página web permitiría a los usuarios acceder a la información desde cualquier

dispositivo conectado a Internet, su efectividad tiene sus limitaciones, especialmente en cuanto a interactividad y actualizaciones instantáneas. A menudo, los estudiantes tienen que entrar al sitio de manera activa para consultar los eventos, lo que puede generar una desconexión y disminuir su interés en la información.

También se consideró el uso de aplicaciones como WhatsApp, Telegram o el correo electrónico para difundir los eventos. Aunque estas plataformas permiten enviar información en tiempo real, no ofrecen una experiencia organizada ni personalizada. Los mensajes pueden perderse fácilmente entre otras notificaciones, y navegar entre los diferentes eventos y fechas puede resultar complicado. Además, no están diseñadas para gestionar eventos de manera eficiente, ni para ofrecer funciones como la inscripción o recordatorios.

Las redes sociales como Instagram, Facebook o Twitter pueden ser una buena manera de llegar a más estudiantes, pero existe el riesgo de que la información se pierda entre otros contenidos. Aunque permiten cierta interacción, no ofrecen herramientas específicas para gestionar eventos. Finalmente, enviar mensajes de texto para notificar a los estudiantes sobre eventos es una alternativa, pero tiene sus limitaciones.

El envío masivo de SMS puede implicar costos adicionales y, al igual que otras soluciones, carece de interactividad. Además, no permite una experiencia personalizada ni facilita la gestión eficiente de inscripciones o recordatorios. En cuando a la comparación con proyecto similares, el desarrollo de la aplicación móvil para la gestión

y acceso a la información de actividades extracurriculares para instituciones de educación superior será evaluado frente a cuatro iniciativas relacionadas.

En el trabajo de De La Riva et al. (2012), el objetivo principal es utilizar dispositivos móviles como una estrategia educativa para mejorar la disponibilidad y el acceso al conocimiento. Para lograr esto, utilizan tecnologías como plataformas de desarrollo de aplicaciones móviles para Android e iOS, adaptando las interfaces de usuario para que sean más amigables pantallas pequeñas.

Al comparar este enfoque con el nuestro, se encontró que ambos proyectos comparten la intención de mejorar la comunicación y facilitar el acceso a la información mediante aplicaciones móviles. Sin embargo, nuestro proyecto se distingue por el uso de tecnologías de computación en la nube para el almacenamiento y gestión de datos, lo que nos permite ser más eficientes. También con la implementación de un sistema de búsqueda avanzada, mejora significativamente la experiencia del usuario. Gracias a estas características, nuestra solución es más escalable, accesible y económica, aspectos esenciales para manejar grandes volúmenes de datos en tiempo real, especialmente considerando el número de usuarios que se espera.

Por otro lado, en el proyecto de Machuca et al. (2024) busca desarrollar una solución tecnológica para la gestión de eventos universitarios, priorizando la organización y asistencia a eventos dentro de la universidad. Aunque los autores mencionan el uso de bases de datos, no ofrecen detalles sobre si se emplearán tecnologías en la nube o que

lenguajes de programación se utilizarán.

Al comparar ambos proyectos, se evidencia que ambos facilitan el registro de usuarios y la gestión de eventos, pero nuestro enfoque va más allá al ofrecer funcionalidades orientadas que mejoran la experiencia del usuario, como la sincronización en tiempo real. Estas características brindan un enfoque más dinámico y moderno. Además, al incorporar tecnologías avanzadas como Firebase, posicionamos nuestra solución como una opción más sofisticada y escalable en comparación con el proyecto mencionado.

Los proyectos de Lis Santofimio (2021) y Blanco Espinosa y Madrid Plata (2014) comparten el objetivo de desarrollar aplicaciones para la gestión de eventos en instituciones educativas, aunque cada uno aborda este desafío desde enfoques y tecnologías diferentes. Lis Santofimio (2021) se centra en una aplicación web para la organización y registro de eventos en la Institución Universitaria Politécnico Grancolombiano. Su enfoque utiliza metodologías ágiles y tecnologías web para solucionar la falta de un sistema centralizado.

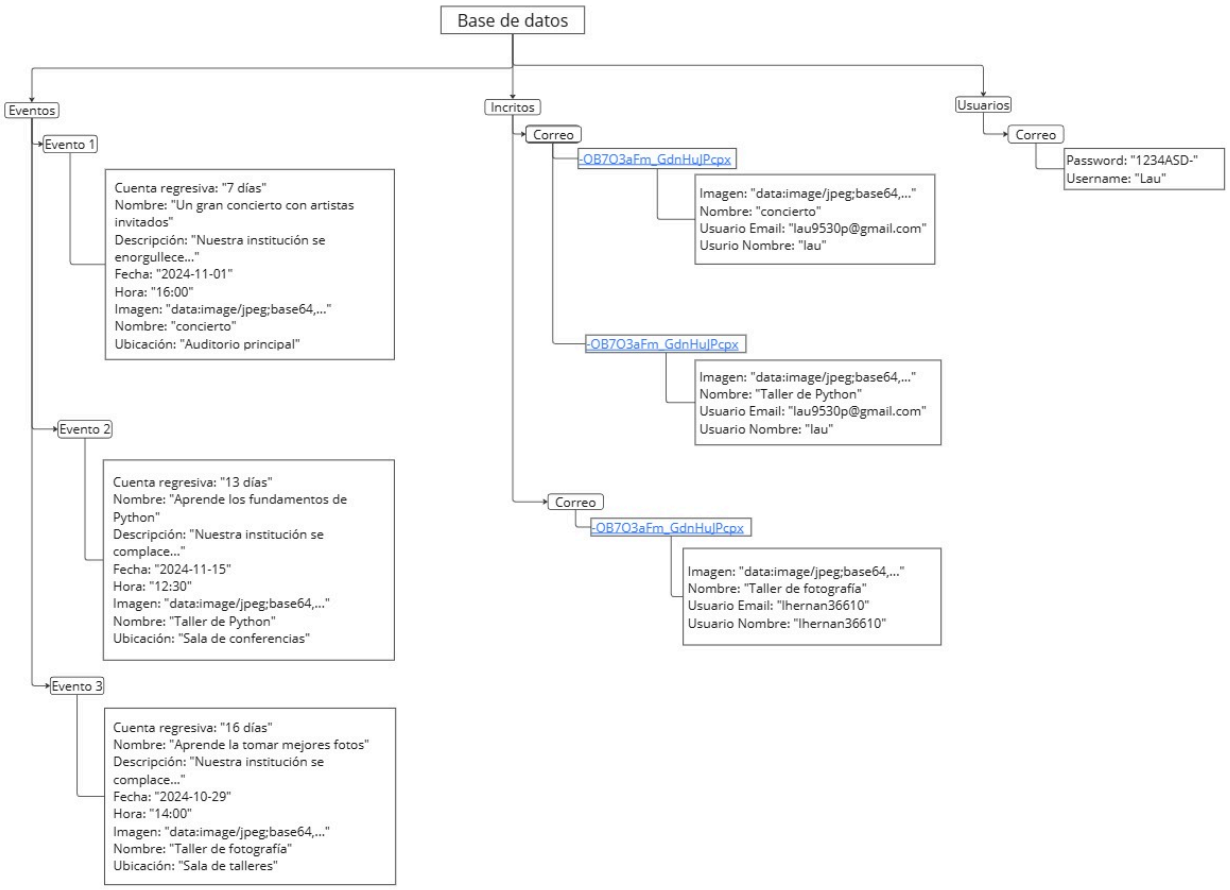
Por otro lado, Blanco y Madrid (2014) desarrollan una aplicación web para gestionar eventos académicos en la Universidad Pontificia Bolivariana, seccional Bucaramanga, aplicando también metodologías ágiles y tecnologías web. Ambos proyectos tienen el objetivo de mejorar la gestión y difusión de eventos académicos, aunque se enfocan en instituciones específicas.

Nuestro proyecto, en contraste, se dedica al desarrollo de una aplicación móvil que permita gestionar y acceder a la información de actividades extracurriculares en instituciones de educación superior. Además, busca mejorar la difusión y participación en eventos extracurriculares a través de una plataforma móvil centralizada que abarque varias instituciones. Para ello, se utilizó tecnologías móviles y en la nube, lo que permite ofrecer información en tiempo real, brindando así una experiencia más dinámica y accesible para los usuarios.

**Diagrama de base de datos NoSQL**

En el siguiente diagrama muestra la base de datos NoSQL:

*Figura 1. Representación de la base de datos en Firebase.*




La imagen muestra el esquema de la base de datos que organiza la información por medio de nodos, el primero es, eventos, usuarios e inscripciones para la aplicación. A continuación, se explica de manera general las secciones de este esquema:

1. **Base de datos principal**: Se trata de la estructura central de almacenamiento que contiene tres colecciones principales: *Eventos*, *Usuarios* e *Inscritos*.
2. **Eventos**: Esta colección almacena los detalles de los eventos. Cada evento tiene una cuenta regresiva, un nombre, una descripción, la fecha y hora, una imagen, el nombre del evento y la ubicación.

3. **Usuarios**: Aquí se almacena la información relacionada con los usuarios que pueden registrarse o inscribirse a los eventos. Se incluyen datos como el correo electrónico, el nombre de usuario y su contraseña.
4. **Inscritos**: Esta parte muestra qué usuarios están inscritos en cada evento. Los usuarios se asocian a los eventos en los que están registrados.

**Diagrama de casos de uso**

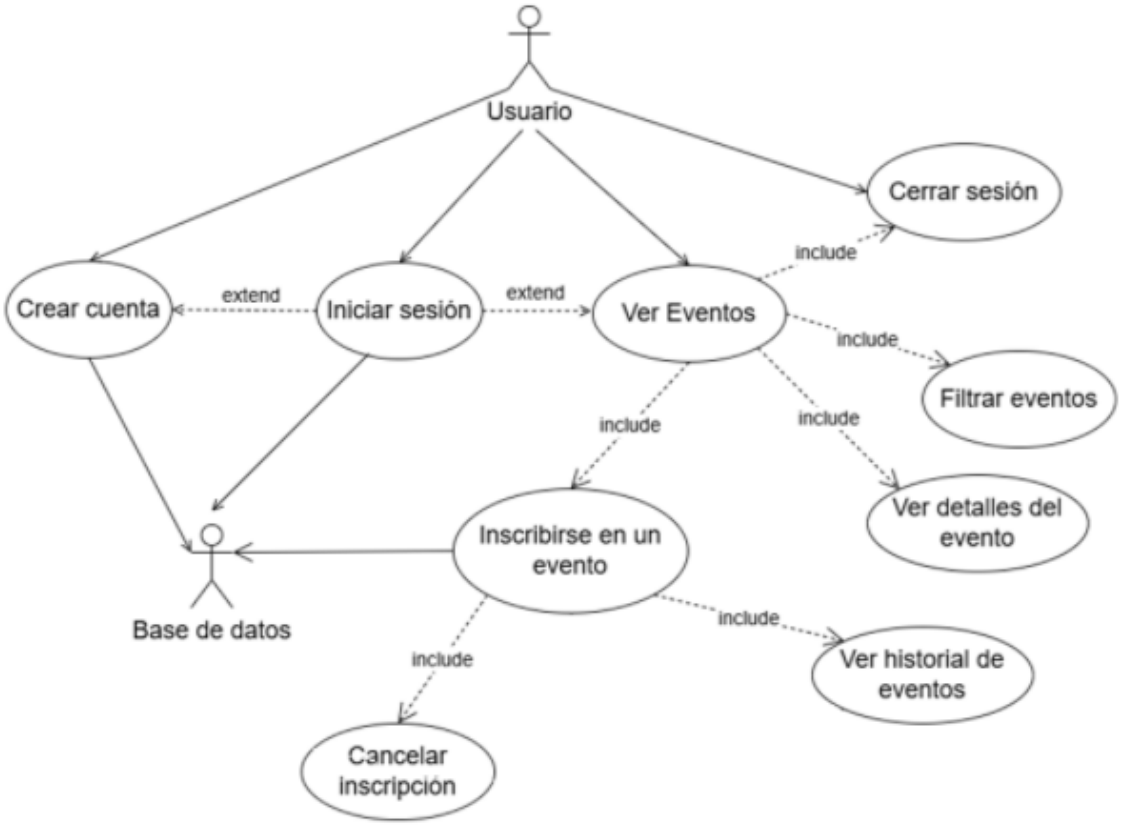
En el siguiente diagrama de casos de uso se muestra las principales funcionalidades disponibles para el usuario en la aplicación móvil. A continuación, se explican cada uno de los casos de uso y sus relaciones:

- **Crear cuenta**: Permite al usuario registrarse en el sistema y crear una cuenta, este es un requisito inicial para acceder a las funcionalidades de la aplicación móvil.
- **Iniciar sesión**: Una vez registrada una cuenta, el usuario puede iniciar sesión para acceder a las opciones personalizadas, y se requiere una autenticación, como la inscripción en eventos.
- **Ver eventos**: Este caso de uso permite al usuario consultar la lista de eventos disponibles.
- **Filtrar eventos**: Permite al usuario aplicar filtros para encontrar eventos específicos.
- **Ver detalles del evento**: Permite al usuario ver la información detallada de un evento específico.

- **Ver historial de eventos**: Muestra los eventos a los que el usuario se ha inscrito o participado previamente.
- **Inscribirse en un evento**: Este caso de uso permite al usuario registrarse en un evento de su interés, este proceso interactúa con la base de datos para almacenar la inscripción del usuario.
- **Cancelar inscripción**: Si el usuario decide no asistir a un evento en el que se inscribió, puede cancelar su inscripción mediante este caso de uso.
- **Cerrar sesión:** Este caso de uso permite al usuario finalizar su sesión en el sistema.

Además, para entender mejor el diagrama, es importante tener en cuenta que extend se utiliza como relaciones de extensión, como en "Iniciar sesión" que extiende al caso "Inscribirse en un evento", lo que implica que el usuario debe estar autenticado para poder inscribirse en eventos. Por otro lado, include se emplea para relaciones como "Filtrar eventos", "Ver detalles del evento" y "Ver historial de eventos" que están incluidos en "Ver eventos", como se muestra en la *Figura 2*.

*Figura 2. Diagrama de casos de uso*




***Nota***. *Creado en draw.io y es una elaboración propia.*

**Diagrama de contexto del modelo c4**

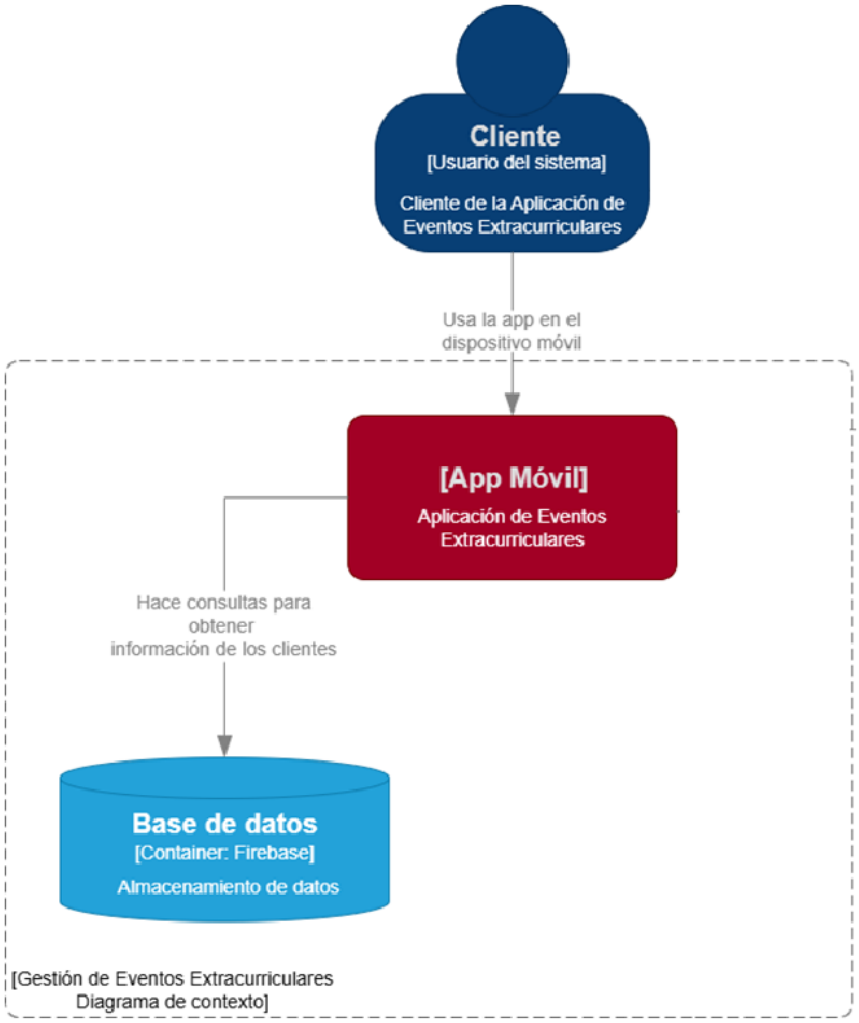
El diagrama de contexto del modelo c4 muestra la interacción entre los componentes principales del sistema de gestión de eventos extracurriculares como se muestran en la *Figura 3*. A continuación, se explican los elementos y sus relaciones:

- **Cliente (Usuario del sistema):** Representa al usuario de la aplicación, quien interactúa con el sistema a través de un dispositivo móvil, este usuario puede ver y registrarse en eventos, y recibe notificaciones relacionadas con sus actividades.
- **App Móvil (Aplicación de Eventos Extracurriculares):** Es la aplicación principal que los usuarios utilizan para acceder a la información sobre eventos extracurriculares, esta aplicación se conecta a la base de datos para obtener y

almacenar información de los eventos y de los usuarios.

- **Base de Datos (Firebase):** Actúa como el repositorio de información del sistema, la aplicación consulta esta base de datos para obtener información sobre los eventos y los usuarios, Firebase se utiliza como contenedor para gestionar el almacenamiento y acceso a los datos de manera eficiente.

*Figura 3. Diagrama de contexto del modelo c4*

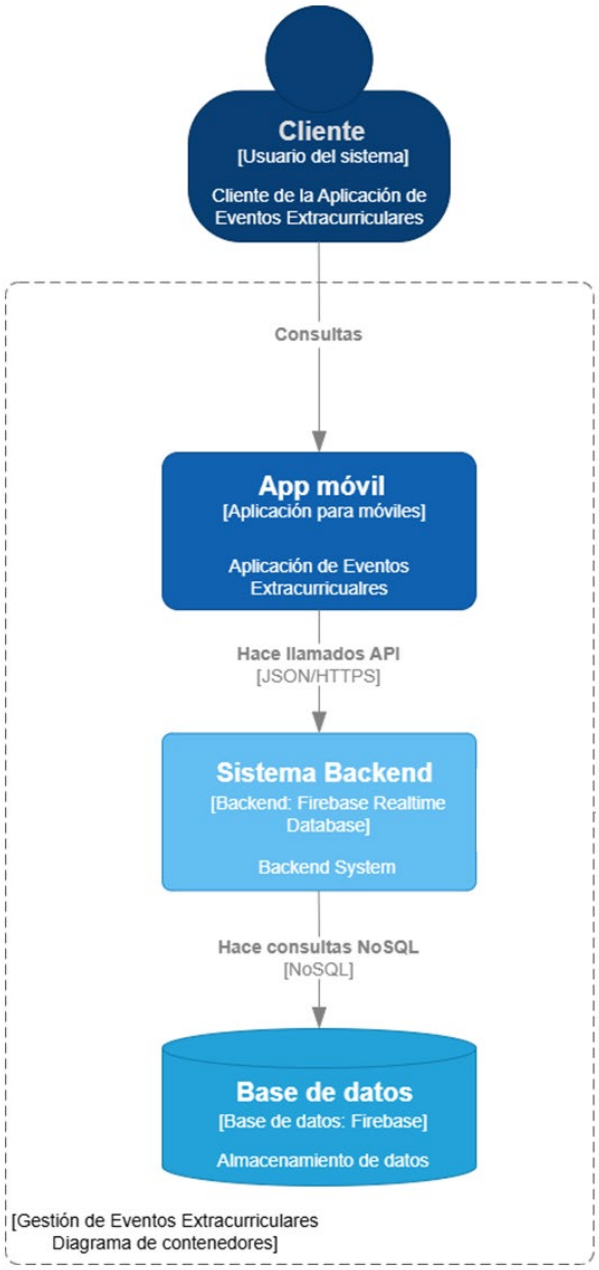



***Nota**. Creado en draw.io y es una elaboración propia.*

## Diagrama de contenedor del modelo c4

A continuación, se explica el diagrama de contenedor del modelo c4 que se encuentra en la Figura 4.

*Figura 4. Diagrama de contenedor del modelo c4*




***Nota**. Creado en draw.io y es una elaboración propia.*

- **Cliente (Usuario del sistema):** Representa al usuario que interactúa con la App

móvil para consultar o gestionar eventos extracurriculares.

- **App móvil:** Es la aplicación que el cliente utiliza en su dispositivo móvil, en donde realiza solicitudes como las consultas y actualizaciones a través de llamadas API utilizando el protocolo JSON/HTTPS para comunicarse con el sistema backend.
- **Sistema Backend:** Este componente se encarga de procesar las solicitudes provenientes de la aplicación móvil, donde utiliza Firebase Realtime Database como su base de datos principal para almacenar y recuperar información, la comunicación con la base de datos se realiza mediante consultas NoSQL.
- **Base de datos (Firebase Realtime Database):** Es una base de datos en la nube que almacena toda la información relacionada con los eventos extracurriculares, permite actualizaciones y sincronización en tiempo real con el backend.

**Diagrama de componentes del modelo c4**

El diagrama de contenedores del modelo C4 detalla los diferentes contenedores que componen el sistema de gestión de eventos extracurriculares, proporcionando una visión clara de cómo se agrupan y comunican entre sí las aplicaciones, bases de datos y servicios principales, como se observa en la Figura 5. A continuación, se describen los contenedores, sus responsabilidades y las interacciones que mantienen entre ellos:

- **Cliente (Usuario del sistema):** Representa al usuario final que utiliza la aplicación móvil para consultar, registrarse o gestionar eventos extracurriculares, interactuando con el sistema a través de una interfaz intuitiva y amigable.
- **App móvil**: Contenedor principal en el dispositivo del cliente, diseñado para permitir la consulta, inscripción y gestión de eventos extracurriculares. Se conecta

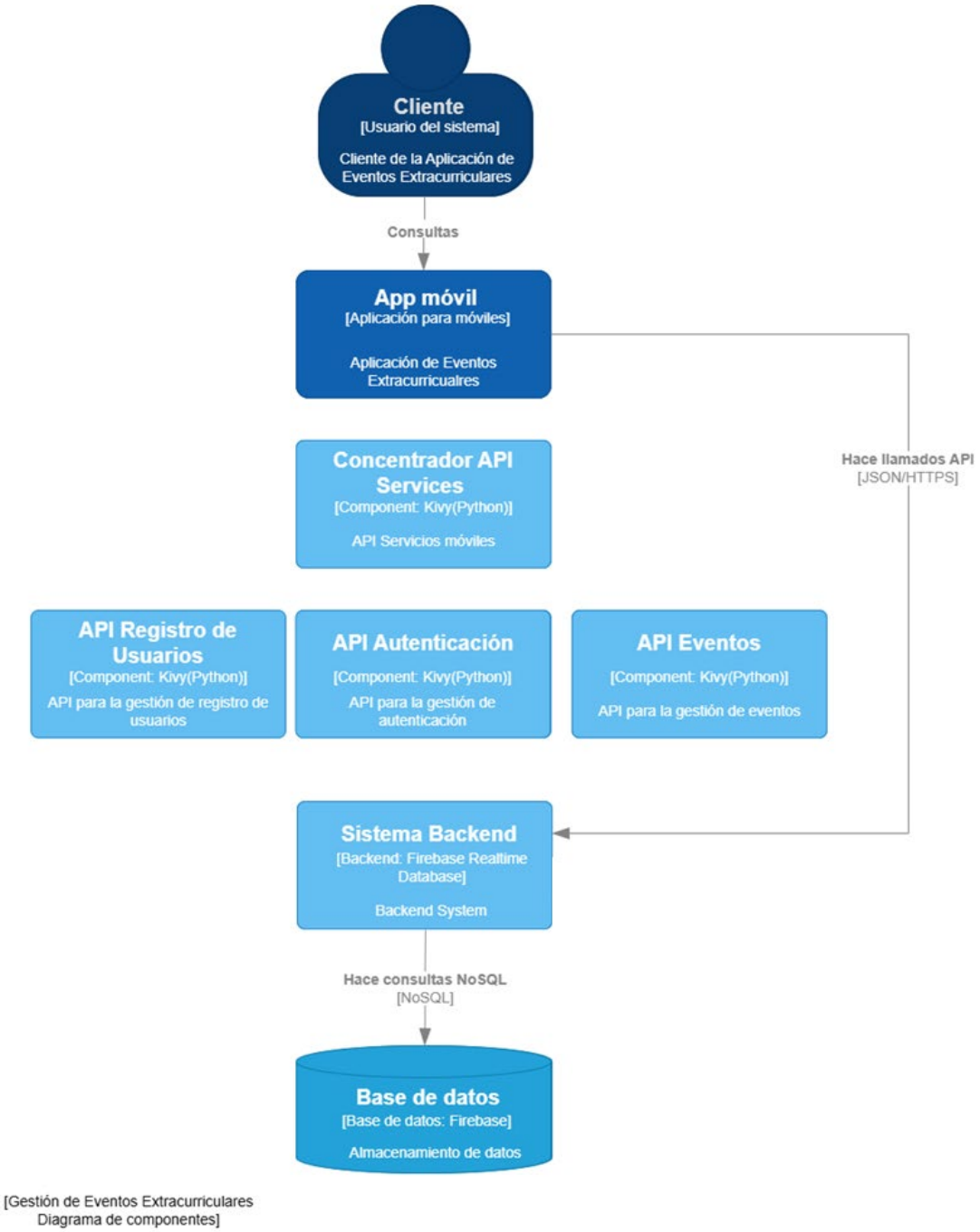
al sistema backend mediante llamadas API seguras utilizando el protocolo HTTPS y formato JSON para el intercambio de datos.

- **Sistema Backend**: Responsable de gestionar las solicitudes provenientes de la aplicación móvil, procesándolas y garantizando respuestas eficientes. Utiliza Firebase Realtime Database como base de datos principal para almacenar y recuperar datos relacionados con eventos, inscripciones y usuarios, empleando consultas NoSQL para la comunicación.
    - **Concentrador API Services**: Módulo principal del sistema backend desarrollado en Python, que actúa como un gestor de solicitudes. Su función es recibir las peticiones provenientes de la aplicación móvil y dirigirlas al componente correspondiente para su procesamiento, garantizando la correcta distribución de las tareas y la comunicación eficiente entre los diferentes servicios del backend.
    - **API Registro de Usuarios**: Encargado de la creación y gestión de cuentas de usuario, permitiendo a los clientes registrarse y almacenar su información de manera segura. Está diseñado para integrarse con Firebase Realtime Database, lo que asegura una sincronización eficiente de los datos en tiempo real.
    - **API Autenticación**: Responsable de validar las credenciales de los usuarios y gestionar la seguridad del sistema, utilizando un sistema de autenticación basado en tokens para proteger los datos y garantizar que solo los usuarios autorizados puedan acceder a las funcionalidades de la aplicación.

- **API Eventos**: Maneja todas las operaciones relacionadas con los eventos, como la creación, consulta, actualización y eliminación. Su integración con Firebase Realtime Database permite una gestión dinámica de los datos de eventos, asegurando que los cambios sean reflejados en tiempo real para todos los usuarios.

- **Base de datos (Firebase Realtime Database):** Es una base de datos en la nube que almacena toda la información relacionada con los eventos extracurriculares, permite actualizaciones y sincronización en tiempo real con el backend.

*Figura 5. Diagrama de componentes del modelo c4*



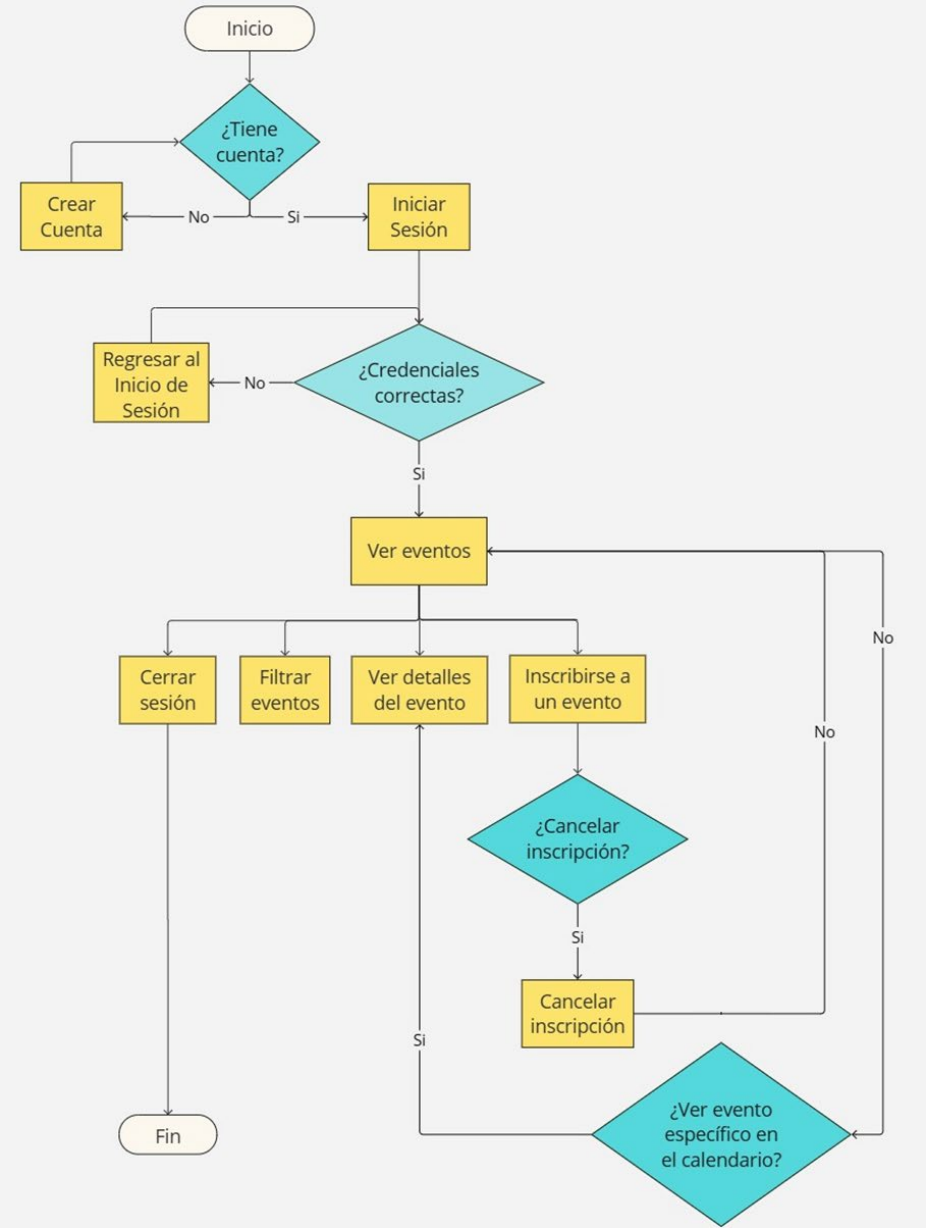

***Nota:*** *Creado en draw.io y es una elaboración propia.*

**Diagrama de flujo**

La figura 6 representa el proceso lógico que un usuario sigue dentro de la aplicación de gestión de eventos extracurriculares, desde el momento en que inicia su interacción hasta la finalización de sus tareas. El flujo comienza con la decisión de si el usuario tiene una cuenta existente. Si no la tiene, puede crear una nueva cuenta, si ya dispone de una, puede iniciar sesión. Una vez autenticado, el sistema valida las credenciales y, si son correctas, permite al usuario acceder a la lista de eventos disponibles. En caso contrario, se le redirige para reintentar el inicio de sesión.

Dentro de la aplicación, el usuario puede realizar diversas acciones, como filtrar eventos, inscribirse, cancelar inscripciones o consultar detalles específicos. En cualquier momento, el usuario puede cerrar sesión, lo que finaliza su interacción y lo redirige al inicio. Este flujo garantiza una navegación lógica, asegurando que cada decisión lleve al usuario hacia las funcionalidades clave de la aplicación.

*Figura 6. Diagrama de flujo*




***Nota**. Creado en miro y es una elaboración propia.*

### Diagrama de clases

En la figura 7 se muestra el diagrama de clases, la cual es la arquitectura de la aplicación móvil, utilizando KivyMD, donde cada clase cumple un rol específico dentro del sistema, desde la gestión de la autenticación y la navegación entre pantallas, hasta la carga y

visualización de eventos y sus detalles.

La clase principal, LoginApp, actúa como controlador central, coordinando la interacción entre las diferentes pantallas, mientras que el ScreenManager facilita la navegación fluida entre vistas como MainScreen, LoginScreen y InscritoScreen. Esta estructura modular no solo mejora la mantenibilidad del código, sino que también permite una fácil escalabilidad y la integración de servicios externos como bases de datos en tiempo real.

**LoginApp (MDApp)**

Es la clase principal de la aplicación, hereda de MDApp y actúa como el controlador central, en donde sus atributos son:

- **title**: Título de la aplicación.
- **url:** URL base para las operaciones de red.
- **events_url**: Endpoint para la gestión de eventos.
- **Auth**: Manejador de autenticación (token o credenciales de usuario).
- **sm (ScreenManager):** Responsable de gestionar la navegación entre las distintas pantallas.
- **all_events**: Cache local para almacenar la lista de eventos.
- **user_email y username**: Almacenan información del usuario autenticado.

Y sus métodos son:

- **build():** Construye la interfaz gráfica y configura las pantallas.
- **login() y signup():** Procesan el flujo de autenticación.
- **inscribirse_evento():** Permite a los usuarios registrarse en eventos.

- **cargar_inscripciones():** Carga las inscripciones existentes desde el backend.
- **logout_and_redirect():** Cierra la sesión y redirige a la pantalla de inicio.

**Pantallas Secundarias (Screens)**

Cada pantalla es una subclase de Screen, gestionada por el ScreenManager de LoginApp.

**LoginScreen**

En donde sus atributos son login_email y login_password, estos son campos para capturar las credenciales del usuario y el método on_login() quien maneja la lógica de autenticación y se conecta con el backend para validar credenciales y, si son correctas, navega a MainScreen.

**SignupScreen**

Sus atributos son signup_email y signup_password, son los campos encargados para registrar nuevos usuarios y su método es on_signup(), se trata de generar esa lógica para registrar un nuevo usuario, comunica los datos al backend y configura el entorno inicial del usuario.

**MainScreen**

Sus atributos son:

- events_box: Contenedor dinámico que lista los eventos disponibles.
- menu_button: Botón para acceder a opciones adicionales.

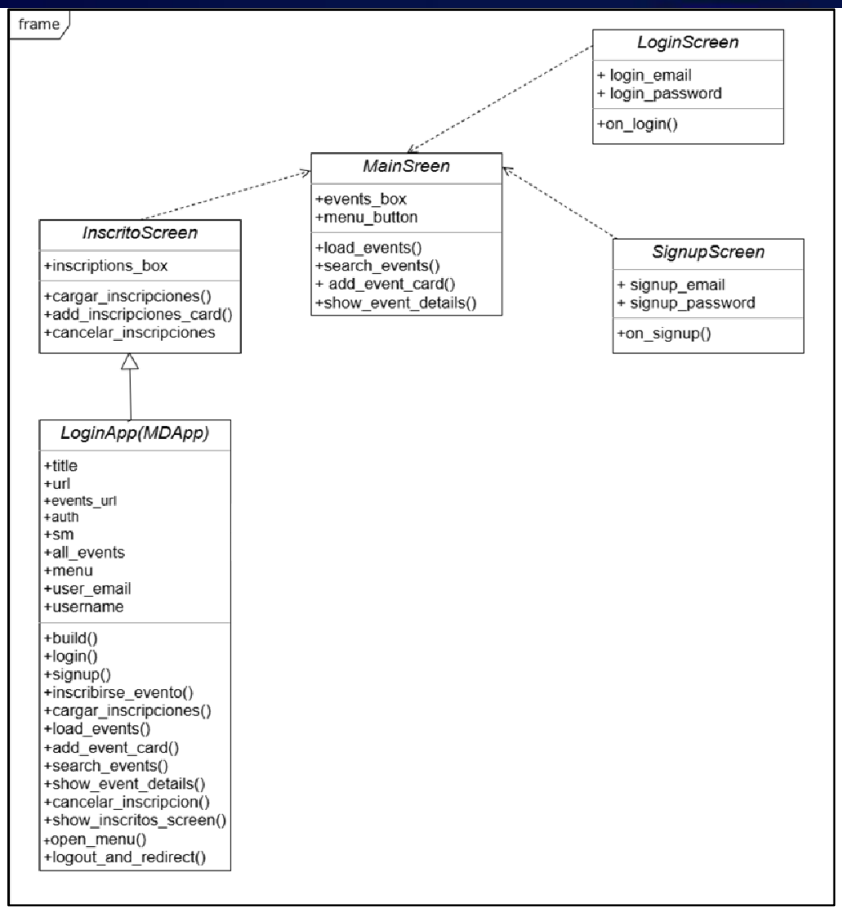
Y métodos:

- load_events(): Recupera eventos del backend y los muestra.
- search_events(): Permite realizar búsquedas en la lista de eventos.
- add_event_card(): Renderiza la representación visual de un evento.
- show_event_details(): Navega a una vista de detalle para un evento seleccionado.

**InscritoScreen**

Donde tiene este atributo de inscriptions_box, es un contenedor dinámico que lista los eventos en los que el usuario está inscrito y tiene el método de cargar_inscripciones() es el encargado de obtiene inscripciones del backend y las renderiza, tambien esta add_inscripciones_card(), añade una representación gráfica para cada inscripción y cancelar_inscripciones(), el encargado de eliminar una inscripción seleccionada.

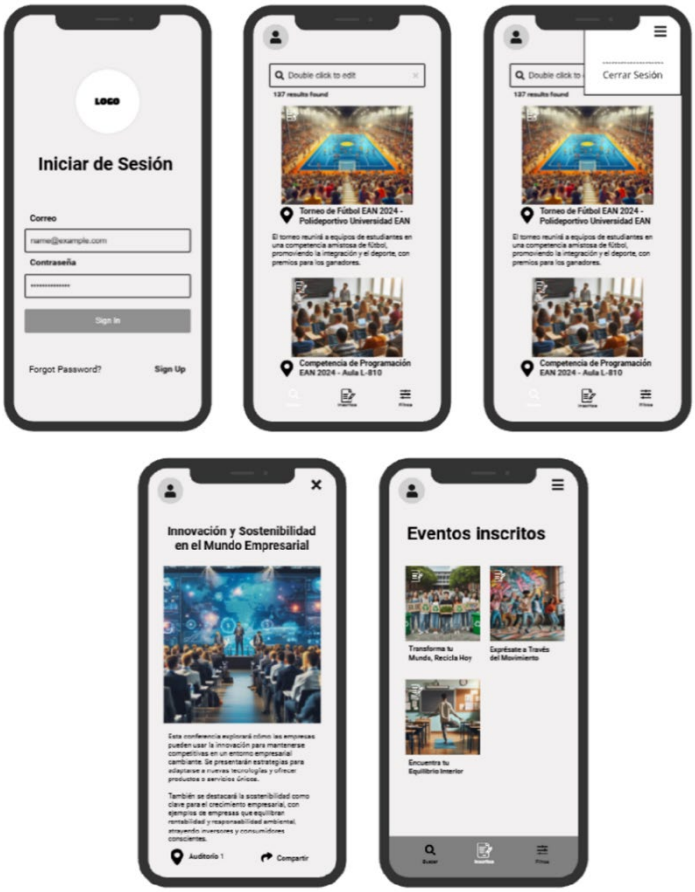
*Figura 7. Diagrama de clases*




***Nota**. Creado en draw.io y es una elaboración propia.*

## Diseño de mockup

*Figura 8.Diseño de mockup*




***Nota***. *Creado en miro y es una elaboración propia.*

Este mockup representa la aplicación móvil diseñada para gestionar y acceder a actividades extracurriculares en Instituciones de Educación Superior, la interfaz busca proporcionar una experiencia de usuario fluida y accesible, permitiendo a los usuarios registrarse, buscar actividades, ver detalles específicos e inscribirse. A lo largo de las diferentes pantallas, la navegación es intuitiva y visualmente atractiva, facilitando la interacción rápida con la información de las actividades disponibles. A continuación, se explicará cada pantalla:

**Inicio de sesión:** Es una interfaz simple y clara para que el usuario ingrese su correo electrónico y contraseña, también ofrece una opción para recuperar contraseña en donde garantiza que solo usuarios autorizados puedan acceder a la aplicación.

**Pantalla de eventos:** Incluye una barra de búsqueda en la parte superior para permitir al usuario buscar actividades por palabras clave, debajo de la barra de búsqueda, se muestran eventos y actividades destacadas, con imágenes y una breve descripción.

**Detalles de eventos:** Al seleccionar un evento de la lista, el usuario puede ver detalles ampliados, que incluyen una descripción detallada, el tiempo restante para el evento, imágenes relacionadas y opciones de interacción, como registrarse o compartir.

**Eventos inscritos:** Muestra una lista de eventos a los que el usuario ya se ha inscrito, con imágenes y descripciones.

***Implementación del Sistema***

En el siguiente fragmento de código se realizan las importaciones necesarias para garantizar la funcionalidad de la aplicación móvil. Estas herramientas permiten una experiencia de usuario fluida, atractiva y funcional. A continuación, se explica cada importación y cómo contribuye al funcionamiento de la aplicación:

- **Manejo de fechas y horas**

- from datetime import datetime: Facilita el manejo de fechas y horas, permitiendo registrar, mostrar o calcular información temporal.

- **Diseño de la interfaz gráfica**
  - from kivy.lang import Builder: Permite diseñar la interfaz de usuario de manera rápida y sencilla mediante un lenguaje propio de Kivy, ideal para organizar elementos visuales y pantallas.

- **Creación de la app**
  - from kivymd.app import MDApp: Esta clase sirve como base de la aplicación, integrando el diseño Material Design y gestionando las funcionalidades principales.

- **Gestión de pantallas**
  - from kivy.uix.screenmanager import ScreenManager, Screen: Facilitan la navegación entre diferentes pantallas dentro de la aplicación, proporcionando una experiencia de usuario fluida.

- **Conexión con el servidor**
  - import requests: Permite realizar consultas y obtener información desde servidores o bases de datos remotas.

- **Menús interactivos**
    - from kivymd.uix.menu import MDDropdownMenu: Ayuda a crear menús desplegables que ofrecen opciones al usuario de manera visualmente atractiva.

- **Desplazamiento en listas**
    - from kivy.uix.scrollview import ScrollView: Gestiona pantallas con gran cantidad de elementos, permitiendo al usuario desplazarse cómodamente.

- **Botones para interactuar**
    - from kivymd.uix.button import MDRaisedButton: Proporciona botones con un diseño destacado y elegante, mejorando la interacción del usuario con la aplicación.

- **Configuración de la ventana de la app**
    - from kivy.core.window import Window: Permite controlar aspectos como el tamaño de la ventana de la aplicación o su título.

- **Manejo de datos en formato JSON**
    - import json: Facilita la gestión de datos estructurados en formato JSON, ideal para la comunicación entre sistemas.

- **Diálogos para mostrar mensajes**
    - from kivymd.uix.dialog import MDDialog: Permite mostrar mensajes importantes al usuario en ventanas emergentes.

- **Organización de la interfaz**
    - from kivy.uix.boxlayout import BoxLayout: Ayuda a organizar elementos de la interfaz en filas o columnas, mejorando su estructura y claridad.

- **Acciones programadas**
    - from kivy.clock import Clock: Ayuda a organizar elementos de la interfaz en filas o columnas, mejorando su estructura y claridad.

- **Botones sencillos**
    - from kivymd.uix.button import MDFlatButton: Ofrece un diseño más sencillo, ideal para acciones menos destacadas o con un estilo limpio.

- **Tarjetas para mostrar información**
    - from kivymd.uix.card import MDCard: Permite mostrar información de manera organizada y atractiva, utilizando tarjetas visuales.

- **Visualización de texto**
    - from kivymd.uix.label import MDLabel: Gestiona la presentación de texto en la aplicación.

- **Visualización de imágenes**
    - from kivy.uix.image import Image: Permite incluir fotos o íconos en la interfaz.

- **Ventanas emergentes (pop-ups)**
    - from kivy.uix.popup import Popup: Muestra información importante que requiere la atención inmediata del usuario.

*Figura 9. Importación de las librerías del framework Kivy*

```
from datetime import datetime
from kivy.lang import Builder
from kivymd.app import MDApp
from kivy.uix.screenmanager import ScreenManager, Screen
import requests
from kivymd.uix.menu import MDDropdownMenu
from kivy.uix.scrollview import ScrollView
from kivymd.uix.button import MDRaisedButton
from kivy.core.window import Window
import json
from kivymd.uix.dialog import MDDialog
from kivy.uix.boxlayout import BoxLayout
from kivy.clock import Clock
from kivymd.uix.button import MDFlatButton
from kivymd.uix.card import MDCard
from kivymd.uix.label import MDLabel
from kivy.uix.image import Image
from kivy.uix.popup import Popup
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código, se definen las clases que representan cada una de las pantallas que conforman la estructura principal de la aplicación. Cada pantalla tiene una funcionalidad específica que contribuye a la experiencia integral del usuario.

- **LoginScreen:** Pantalla de inicio de sesión

  Esta clase gestiona la pantalla donde los usuarios ingresan sus credenciales (nombre de usuario y contraseña) para acceder a su cuenta. Es el punto de entrada principal a la aplicación.

- **SignupScreen:** Pantalla de registro

  Diseñada para que los nuevos usuarios puedan crear una cuenta. Aquí se recopila la información necesaria, como nombre, correo electrónico y contraseña, para registrarse en la aplicación.

- **MainScreen:** Pantalla principal

  Una vez que los usuarios inician sesión, acceden a esta pantalla principal. Sirve como el núcleo de la aplicación, donde se encuentran accesos a otras funcionalidades clave.

- **EventsScreen:** Pantalla de eventos

  Esta pantalla permite a los usuarios explorar los eventos disponibles, revisar detalles específicos de cada evento, y decidir si desean participar.

- **InscritosScreen:** Pantalla de eventos inscritos

  Representa el espacio donde los usuarios pueden consultar los eventos a los que ya se han registrado, manteniendo un control organizado de su participación.

*Figura 10. Definición de las clases de las diferentes pantallas*

```
class LoginScreen(Screen):
    pass

class SignupScreen(Screen):
    pass

class MainScreen(Screen):
    pass

class EventsScreen(Screen):
    pass

class InscritosScreen(Screen):
    pass
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código se define la clase “LoginApp”, que es la clase principal de la aplicación móvil. Su responsabilidad principal es configurar la aplicación, inicializar la interfaz de usuario y gestionar las interacciones de los usuarios a través de las diferentes pantallas disponibles.

- **Título de la aplicación**
  - **self.title:** Establece el título de la ventana de la aplicación. En este caso, el título definido es "Inicio sesión".

- **URLs de la base de datos**
    - **Self.url:** Especifica la URL de la base de datos de Firebase Realtime Database donde se almacenan los datos de los usuarios:'https://dbaplicacionmovil-default-rtdb.firebaseio.com/usuarios/.json'.

    - **self.events_url:** Indica la URL de la base de datos donde están almacenados los eventos disponibles:'https://dbaplicacionmovil-default-rtdb.firebaseio.com/eventos/.json'.

- **Autenticación de la API**
    - **Self.auth:** Contiene la clave de autenticación necesaria para acceder a la base de datos Firebase Realtime Database:'QwwWZy39FBqnILNqy1nfvNcwvWyTYSQAhx8619XB'.

- **Configuración de la ventana**
    - **Window.size = (375, 667)**: Define el tamaño inicial de la ventana de la aplicación.

- **Gestión de pantallas con ScreenManager**
    - **self.sm = ScreenManager():** Crea un objeto ScreenManager que se encarga de la navegación entre las pantallas de la aplicación.

- **Cargar el archivo .kv**
    - **Builder.load_file('main.kv'):** Carga el archivo .kv para definir y mostrar los elementos visuales de la aplicación.

- **Agregar pantallas al ScreenManager**
    - **self.sm.add_widget(...):** Añade las pantallas previamente definidas al ScreenManager.

- **Almacenamiento de eventos**
    - **self.all_events = []**: Declara una lista vacía para almacenar los eventos obtenidos de la base de datos.

- **Configuración del menú desplegable**
    - **menu_items = [...]:** Define los elementos que estarán disponibles en el menú desplegable de la pantalla principal.

    - **self.menu = MDDropdownMenu(...)**: Configura el menú desplegable, especificando el botón de activación y las acciones correspondientes.

- **Devolver la pantalla principal**
    - **return self.sm:** Devuelve el objeto ScreenManager, que contiene todas las pantallas, para que sea mostrado como la interfaz principal de la aplicación.

*Figura 11. Clase Principal*

```python
class LoginApp(MDApp):
    def build(self):
        self.title = "Iniciar Sesión"
        self.url = 'https://dbaplicacionmovil-default-rtdb.firebaseio.com/usuarios/.json'
        self.events_url = 'https://dbaplicacionmovil-default-rtdb.firebaseio.com/eventos/.json'
        self.auth = 'QwwWZy39FBqnILNqy1nfvNcwvWyTYSQAhx8619XB'
        Window.size = (375, 667)
        self.sm = ScreenManager()

        # Cargar el archivo .kv
        Builder.load_file('main.kv')

        self.sm.add_widget(LoginScreen(name='loginscreen'))
        self.sm.add_widget(SignupScreen(name='signupscreen'))
        self.sm.add_widget(MainScreen(name= 'mainscreen'))
        self.sm.add_widget(InscritosScreen(name='inscritoscreen'))

        # Variable para almacenar todos los eventos
        self.all_events = []


        # Configurar el menú desplegable
        menu_items = [
            {"viewclass": "OneLineListItem", "text": "Cerrar Sesión", "on_release": self.on_option_1},
        ]
        self.menu = MDDropdownMenu(
            caller=self.sm.get_screen('mainscreen').ids.menu_button,
            items=menu_items,
            width_mult=4,
        )

        return self.sm
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

Como parte del diseño de la base de datos en Firebase Realtime Database, se han creado las siguientes colecciones principales:

- **Colección de Usuarios**
    - Contiene la información de los usuarios registrados en la aplicación, incluyendo detalles como credenciales de inicio de sesión y datos personales esenciales.

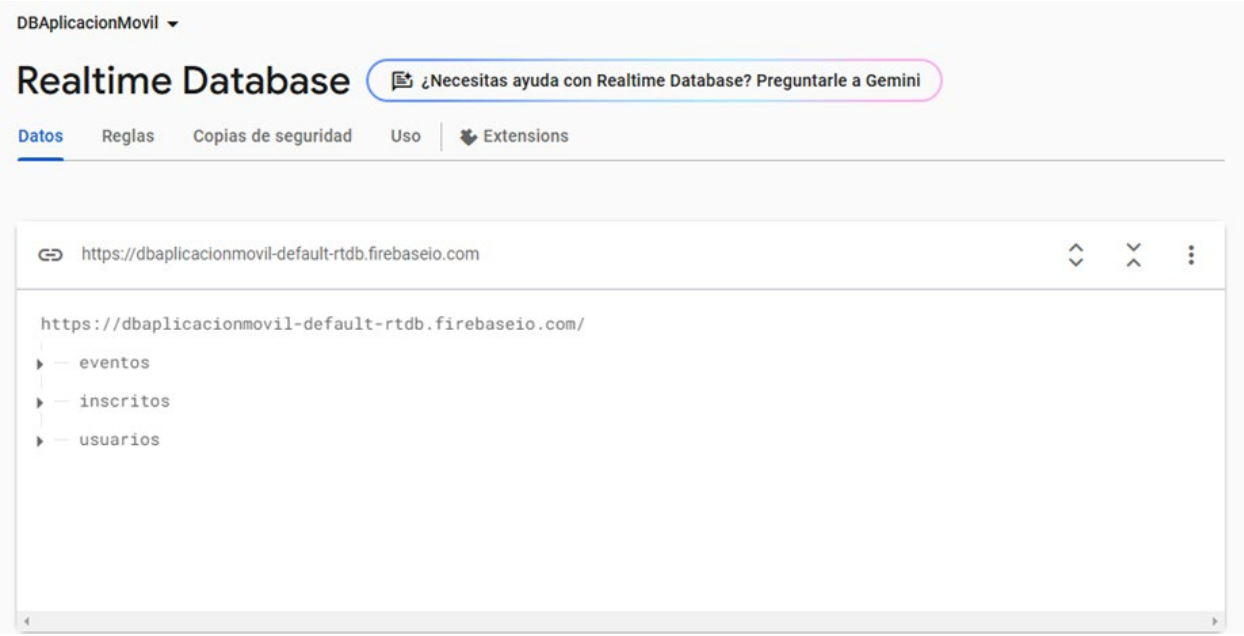
- **Colección de Eventos**

    - Almacena la información de los eventos disponibles, incluyendo títulos, descripciones, fechas, horarios y ubicaciones, lo que permite a los usuarios explorar y acceder a detalles relevantes.

- **Colección de Inscritos** (creada posteriormente)

    - Esta colección registra los eventos a los que los usuarios se han inscrito, estableciendo una relación entre los usuarios y los eventos, y facilitando la gestión de inscripciones dentro de la aplicación.

*Figura 12. Vista de la base de datos (Firebase Real Time)*




***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “login” se encarga de verificar las credenciales de acceso (correo electrónico y contraseña) proporcionadas por el usuario. Si las credenciales son válidas, el usuario es redirigido a la pantalla principal de la aplicación. En caso contrario. se registra un mensaje en la consola indicando que el usuario no existe.

- **Obtener el correo y la contraseña ingresados:**
    - **loginEmail = self.sm.get_screen('loginscreen').ids.login_email.text:** Recupera el correo electrónico ingresado en el campo correspondiente de la pantalla de inicio de sesión.
    - **loginPassword = self.sm.get_screen('loginscreen').ids.login_password.text:** Recupera la contraseña ingresada en el campo correspondiente.

- **Preparar las credenciales para la validación**
    - **supported_loginEmail = loginEmail.replace('.','-'):** Dado que Firebase no admite puntos (.) en las claves, se reemplazan por guiones (-) para garantizar compatibilidad con la base de datos.
    - **supported_loginPassword = loginPassword.replace('.','-'):** Aplica el mismo proceso a la contraseña para facilitar la comparación en la base de datos.

- **Realizar una solicitud para verificar las credenciales**
    - **request = requests.get(self.url+'?auth='+self.auth):** Envía una solicitud GET a la URL de la base de datos para obtener los registros de usuarios. Incluye el token de autenticación necesario.
    - data = request.json(): Convierte la respuesta de la solicitud en un formato JSON para procesar los datos.

- **Procesar los datos obtenidos de Firebase**
    - **emails = set():** Se inicializa un conjunto para almacenar los correos electrónicos registrados en la base de datos.

    - **for key, value in data.items(): emails.add(key):** Itera sobre los registros en la base de datos y agrega las claves (correos electrónicos) al conjunto.

- **Validar las credenciales**
    - **if supported_loginEmail in emails and supported_loginPassword == data[supported_loginEmail]['Password']:** Verifica que el correo electrónico ajustado exista en la base de datos y que la contraseña coincida con la almacenada.

- **Si el inicio de sesión es exitoso**

- **self.user_email = loginEmail:** Guarda el correo del usuario para futuras referencias.

- **self.username = data[supported_loginEmail]['Username']:** Guarda el nombre de usuario asociado al correo.

- **self.login_check = True:** Establece un indicador para confirmar que el inicio de sesión fue exitoso.

- **self.sm.current = 'mainscreen':** Cambia la vista a la pantalla principal de la aplicación.

- **self.load_events():** Llama a la función que carga los eventos disponibles para el usuario.

- **Si el inicio de sesión falla**
  - **else:** Si las credenciales no coinciden con los datos almacenados, registra un mensaje en la consola indicando que el usuario no existe.

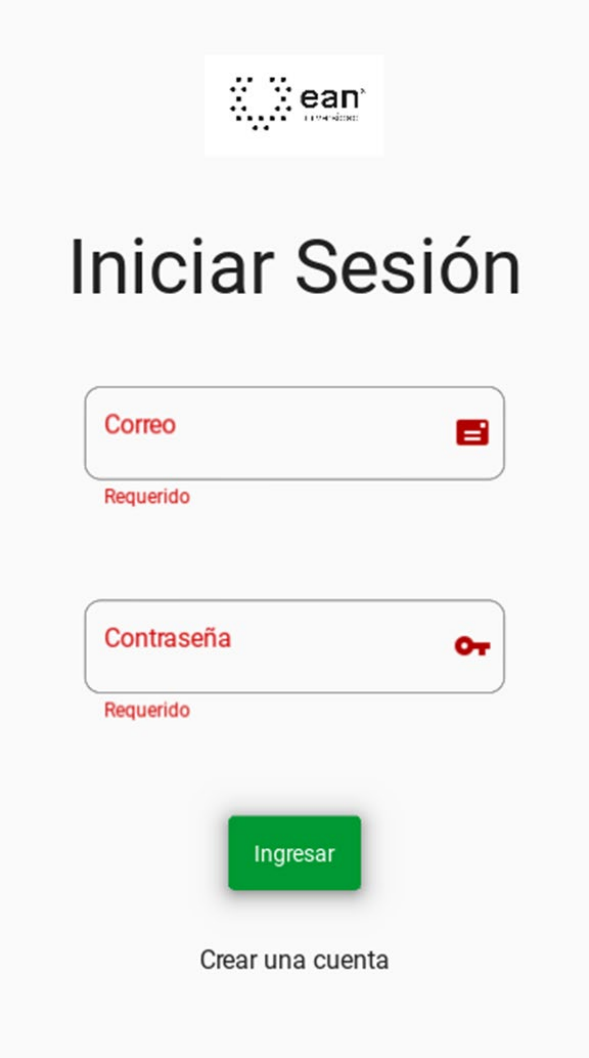
*Figura 13. Función de login*

```
def login(self):
    loginEmail = self.sm.get_screen('loginscreen').ids.login_email.text
    loginPassword = self.sm.get_screen('loginscreen').ids.login_password.text

    self.login_check = False
    supported_loginEmail = loginEmail.replace('.','-')
    supported_loginPassword = loginPassword.replace('.','-')
    request  = requests.get(self.url+'?auth='+self.auth)
    data = request.json()
    emails= set()
    for key,value in data.items():
        emails.add(key)
    if supported_loginEmail in emails and supported_loginPassword == data[supported_loginEmail]['Password']:
        self.user_email = loginEmail
        self.username = data[supported_loginEmail]['Username']
        self.login_check=True
        self.sm.current = 'mainscreen'
        self.load_events()
    else:
        print("user no longer exists")
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

Como resultado de la pantalla LoginScreen, la función login y los diferentes diseños para los elementos de Kivy, se visualiza de la siguiente manera:

*Figura 14. Resultado para la pantalla LoginScreen*




***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “signup” está diseñada para gestionar el proceso de registro de un nuevo usuario en la aplicación.

- **Obtención de datos del formulario**

  Se recogen los datos ingresados en los campos de correo electrónico (signup_email), contraseña (signup_password) y nombre de usuario (signup_username) desde la pantalla de registro (signupscreen).

- **Validación de entrada**
  - Si alguno de los campos está vacío, se muestra un diálogo de error (MDDialog) pidiendo al usuario que ingrese un valor válido.
    - Se utiliza el método split() para verificar si hay texto en los campos, ya que un campo vacío devuelve una lista vacía.
  - Si el nombre de usuario contiene espacios (más de un valor en el campo), también se muestra un diálogo de error indicando que el nombre de usuario no debe contener espacios.

- **Preparación y envío de datos**
  - Si las validaciones son superadas, los datos se preparan en un formato adecuado para la base de datos utilizando json.loads.
    - La dirección de correo electrónico es usada como clave, y la contraseña y el nombre de usuario se almacenan como valores asociados en el formato JSON.
  - Para asegurar compatibilidad con el sistema de almacenamiento, los puntos (.) en los correos electrónicos se reemplazan por guiones (-), ya que Firebase no permite puntos en las claves.

- **Envío a la base de datos**

  Los datos del usuario se envían a la base de datos utilizando una solicitud PATCH a la URL configurada (self.url).

- **Navegación**

  Tras enviar los datos, se redirige al usuario a la pantalla de inicio de sesión (loginscreen).

- **Autenticación**

  Se incluye un valor auth, que parece ser una clave de autenticación utilizada para interactuar con la base de datos de Firebase Real Time.

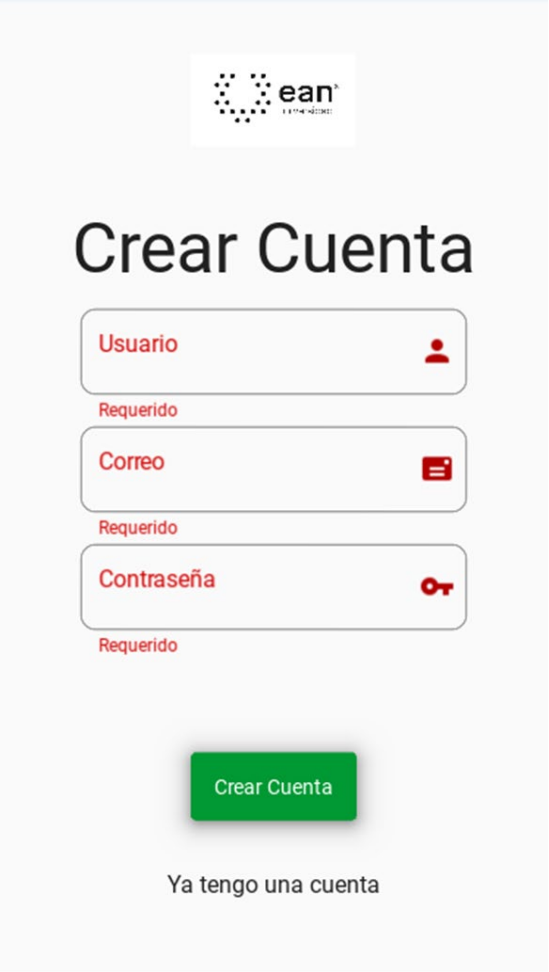
*Figura 15. Función de signup*

```
def signup(self):
    signupEmail = self.sm.get_screen('signupscreen').ids.signup_email.text
    signupPassword = self.sm.get_screen('signupscreen').ids.signup_password.text
    signupUsername = self.sm.get_screen('signupscreen').ids.signup_username.text
    if signupEmail.split() == [] or signupPassword.split() == [] or signupUsername.split() == []:
        cancel_btn_username_dialogue = MDFlatButton(text = 'Retry',on_release = self.close_username_dialog)
        self.dialog = MDDialog(title = 'Invalid Input',text = 'Please Enter a valid Input',size_hint = (0.7,0.2),buttons = [cancel_btn_username_dialogue])
        self.dialog.open()
    if len(signupUsername.split())>1:
        cancel_btn_username_dialogue = MDFlatButton(text = 'Retry',on_release = self.close_username_dialog)
        self.dialog = MDDialog(title = 'Invalid Username',text = 'Please enter username without space',size_hint = (0.7,0.2),buttons = [cancel_btn_username_dialogue])
        self.dialog.open()
    else:
        print(signupEmail,signupPassword)
        signup_info = str({f'\"{signupEmail}\":{{"Password":\"{signupPassword}\","Username":\"{signupUsername}\"}}'})
        signup_info = signup_info.replace(".","-")
        signup_info = signup_info.replace("\'","")
        to_database = json.loads(signup_info)
        print((to_database))
        requests.patch(url = self.url,json = to_database)
        self.sm.get_screen('loginscreen').manager.current = 'loginscreen'
auth = 'QwwWZy39FBqnILNqy1nfvNcwvWyTYSQAhx8619XB'
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

Como resultado de la pantalla SignupScreen, la función signup y los diferentes diseños para los elementos de Kivy se visualiza de la siguiente manera:

*Figura 16. Resultado de la pantalla SignupScreen*




***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “open_menu” está diseñada para abrir un menú cuando el usuario interactía con un botón específico en la interfaz.

- **Apertura del menú**

  La función llama al método open() de un objeto denominado menu, lo que permite desplegar un menú interactivo en la pantalla.

- **Interacción con el botón**

  Esta función se ejecuta cuando se hace clic en el botón correspondiente en la interfaz de usuario. Al llamar a “self.menu.open()”, el menú se muestra en la pantalla, revelando las opciones que contiene para que el usuario pueda interactuar con ellos.

*Figura 17. Función de open_menu*

```
def open_menu(self):
    self.menu.open()
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “logout” tiene como propósito cerrar la sesión del usuario, eliminando cualquier asociada a la sesión activa.

- **Limpiar los datos de sesión**
  - **self.username = None**

    Este comando elimina el nombre de usuario almacenado durante la sesión activa, garantizando que no quede información del usuario previamente autenticado.

  - **self.login_check = False**

Cambia el valor de la variable login_check a False, indicando que el usuario ya no está autenticado en la aplicación.

*Figura 18. Función de logout*

```
def logout(self):
    self.username = None
    self.login_check = False
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código, la función “logout_and_redirect” tiene como objetivo cerrar la sesión del usuario y redirigirlo a la pantalla de incio de sesión “LoginScreen”.

- **Cerrar sesión (limpiar datos)**
  - **self.logout()**

    Llama a la función logout, que elimina los datos de sesión del usuario, como el nombre de usuario y el estado de autenticación, asegurando que no queden rastros del inicio de sesión anterior.

- **Cerrar el menú si está abierto**
  - **if self.menu**

    Comprueba si el menú desplegable está abierto.

  - **self.menu.dismiss()**

Si el menú está abierto, lo cierra para garantizar que la interfaz esté limpia antes de redirigir al usuario.

- **Limpiar los campos de correo y contraseña en el LoginScreen**
  - **login_screen = self.sm.get_screen('loginscreen')**

    Obtiene la referencia a la pantalla de inicio de sesión (LoginScreen).

  - **login_screen.ids.login_email.text**

    Limpia el campo de correo electrónico en el formulario de inicio de sesión.

  - **login_screen.ids.login_password.text**

    Limpia el campo de contraseña, asegurando que no quede información sensible almacenada.

- **Redirigir al LoginScreen**
  - **self.sm.current = loginscreen**

    Cambia la pantalla activa a loginscreen, redirigiendo al usuario a la pantalla de inicio de sesión.

*Figura 19. Función de logout_and _redirect*

```
def logout_and_redirect(self):
    """Cerrar sesión y redirigir al LoginScreen."""
    self.logout()

    if self.menu:
        self.menu.dismiss()

    login_screen = self.sm.get_screen('loginscreen')
    login_screen.ids.login_email.text = ''
    login_screen.ids.login_password.text = ''

    self.sm.current = 'loginscreen'
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “on_option_1” maneja la lógica para cerrar sesión cuando se selecciona la opción 1.

- **Mostrar mensaje de consola**
    - **print("Opción 1 seleccionada: Cerrando sesión")**

      Imprime un mensaje en la consola indicando que el usuario seleccionó la opción 1 y que se procederá con el cierre de sesión. Esto sirve como registro para el desarrollador o para la depuración.

- **Cerrar sesión y redirigir al LoginScreen**
    - **self.logout_and_redirect()**

      Llama a la función logout_and_redirect, que se encarga de cerrar la sesión del usuario eliminando cualquier dato asociado a la sesión activa.

Posteriormente, redirige al usuario a la pantalla de inicio de sesión (LoginScreen), donde puede iniciar sesión nuevamente si lo desea.

*Figura 20. Función on_option_1*

```
def on_option_1(self):
    print("Opción 1 seleccionada: Cerrando sesión")
    self.logout_and_redirect()
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “search_events” permite filtrar los eventos mostrados en la aplicación según un término de búsqueda específico proporcionado por el usuario. El propósito es ayudar al usuario a encontrar eventos relevantes de manera rápida, basándose en el nombre del evento.

- **Filtrar eventos por nombre**
    - **filtered_events = {event_id: event_info for event_id, event_info in self.all_events.items() if query.lower() in event_info.get("nombre", "").lower()}**

Esta línea filtra los eventos almacenados en self.all_events.

- Convierte el término de búsqueda (query) y los nombres de los eventos a minúsculas usando lower() para realizar una búsqueda insensible a mayúsculas/minúsculas.

- Si el término de búsqueda aparece en el nombre del evento, ese evento se incluye en el conjunto filtered_events.

- **Limpiar los widgets actuales**
  - **events_box = self.sm.get_screen('mainscreen').ids.events_box**

    Obtiene el contenedor de eventos de la pantalla principal, donde se mostrarán los eventos filtrados.

  - **events_box.clear_widgets()**

    Limpia el contenedor de eventos eliminando todas las tarjetas de eventos previamente mostradas. Esto asegura que solo se muestren los eventos que coinciden con el término de búsqueda.

- **Crear tarjetas para los eventos filtrados**
  - **for event_id, event_info in filtered_events.items()**

    Recorre todos los eventos filtrados en filtered_events, obteniendo el ID y la información de cada evento.

  - **self.add_event_card(events_box, event_info)**

    Llama a la función add_event_card para crear una tarjeta visual con los detalles de cada evento filtrado. Luego, esta tarjeta se agrega al

contenedor events_box, permitiendo que el usuario vea los resultados de la búsqueda en la interfaz.

*Figura 21. Función de search_events*

```
def search_events(self, query):
    filtered_events = {event_id: event_info for event_id, event_info in self.all_events.items() if query.lower() in event_info.get("nombre", "").lower()}

    events_box = self.sm.get_screen('mainscreen').ids.events_box
    events_box.clear_widgets()

    for event_id, event_info in filtered_events.items():
        self.add_event_card(events_box, event_info)
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “load_events” se encarga de obtener la lista de eventos desde la base de datos de Firebase Real Time, limpiar la pantalla principal y luego mostrar cada evento en la interfaz de usuario.

- **Obtener los eventos desde Firebase**
    - **response = requests.get(self.events_url + '?auth=' + self.auth)**

      La función realiza una solicitud HTTP GET a la URL de la base de datos de Firebase para obtener los datos de los eventos almacenados.

        - El parámetro auth se incluye en la solicitud para autenticar al usuario, asegurando que solo los usuarios autorizados puedan acceder a los datos.

- **Convertir la respuesta en formato JSON**

- **events_data = response.json()**

  La respuesta obtenida de Firebase se convierte en formato JSON, conteniendo los eventos disponibles en la base de datos.

- **Limpiar el contenedor de eventos**
  - **events_box = self.sm.get_screen('mainscreen').ids.events_box**

    Obtiene el contenedor donde se mostrarán las tarjetas de los eventos en la pantalla principal de la aplicación.

  - **events_box.clear_widgets()**

    Limpia el contenedor de eventos para eliminar cualquier tarjeta de eventos previamente mostrada, asegurando que la pantalla se actualice con los nuevos eventos obtenidos.

- **Guardar los eventos en una variable de clase**
  - **self.all_events = events_data**

    Los datos de los eventos se almacenan en la variable self.all_events, lo que permite a la aplicación acceder a la lista completa de eventos y realizar filtrados si es necesario.

- **Crear tarjetas para cada evento**
  - **for event_id, event_info in events_data.items()**

Recorre los eventos obtenidos desde Firebase, obteniendo el event_id (ID del evento) y event_info (información asociada al evento).

- **self.add_event_card(events_box, event_info)**

  Para cada evento, se llama a la función add_event_card, que crea una tarjeta visual con la información del evento y la agrega al contenedor de eventos en la interfaz de usuario.

*Figura 22. Función de load_events*

```
def load_events(self):

    response = requests.get(self.events_url + '?auth=' + self.auth)
    events_data = response.json()

    events_box = self.sm.get_screen('mainscreen').ids.events_box
    events_box.clear_widgets()

    self.all_events = events_data

    for event_id, event_info in events_data.items():
        self.add_event_card(events_box, event_info)
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “add_event_card” se encarga de crear y mostrar una tarjeta de información para un evento en la aplicación. Cada tarjeta presenta detalles clave sobre el evento, como su imagen, nombre, ubicación, descripción, y una cuenta regresiva que muestra el tiempo restante para que el evento ocurra.

- **Crear tarjeta para el evento**
    - Se crea un contenedor de tipo MDCard, que es el widget que contendrá toda la información del evento.
        - La tarjeta se configura con un diseño vertical, un padding de 10dp y un tamaño fijo de 300dp por 300dp.
        - Además, se le da una elevación para crear un efecto de profundidad en la interfaz y se ajusta para que se centre en la pantalla.
- **Imagen del evento**
    - Se obtiene la URL de la imagen del evento de los datos proporcionados (event_info). Si existe, se crea una imagen con un tamaño ajustado (60% de la altura de la tarjeta).
        - La imagen se ajusta para que mantenga su proporción y se adapta al tamaño del contenedor.
        - Si no hay imagen disponible, se mostrará un espacio vacío.
- **Título del evento**
    - Se agrega un título del evento en la tarjeta, que es el nombre del evento (event_info.get("nombre")).

- Si no se encuentra un nombre, se mostrará el texto "Evento sin título".
- El nombre se coloca en una etiqueta (MDLabel), con un estilo de fuente H6 y alineación centrada.

- **Ubicación del evento**
  - Se incluye la ubicación del evento, con un texto que indica "Ubicación: " seguido de la ubicación proporcionada.
    - Si no se proporciona una ubicación, se mostrará "Ubicación desconocida".
    - Esta información también se presenta en un widget MDLabel centrado.

- **Descripción del evento**
  - Se agrega una breve descripción del evento.
    - Si no se encuentra descripción, se mostrará "Sin descripción".
    - La descripción es colocada en otro MDLabel centrado.

- **Cuenta regresiva**

    - Se muestra un texto que dice "Calculando tiempo restante...", para indicar que la cuenta regresiva está en proceso.

        - La fecha y hora del evento se extraen de los datos (event_info.get('fecha') y event_info.get('hora')), y se combinan para crear un objeto datetime.

- **Actualizar la cuenta regresiva**

    - Se calcula el tiempo restante hasta el evento y se actualiza cada segundo.

        - Esto se logra usando el “Clock.schedule_interval”, que ejecuta la función “update_countdown” cada segundo para actualizar el tiempo restante en la etiqueta de cuenta regresiva.

- **Mostrar detalles del evento**

    - Al hacer clic en la tarjeta del evento, se dispara un evento que llama a la función “show_event_details”, mostrando una vista con más detalles del evento.

- **Añadir tarjeta al contenedor**

    - La tarjeta que contiene toda la información del evento se agrega al contenedor en la pantalla, lo que permite que se vea en la interfaz de usuario.

*Figura 23. Función de add_event_card*

```python
def add_event_card(self, container, event_info):
    event_card = MDCard(
        orientation="vertical",
        padding="10dp",
        size_hint=(None, None),
        size=("300dp", "300dp"),
        elevation = 4,
        pos_hint = {"center_x": 0.5}
    )

    image_source = event_info.get("imagen", "")
    image = Image(source=image_source, size_hint_y=0.6, allow_stretch=True, keep_ratio=True)
    event_card.add_widget(image)

    name_label = MDLabel(
        text=event_info.get("nombre", "Evento sin título"),
        font_style="H6",
        size_hint_y=None,
        height="30dp",
        halign="center"
    )
    event_card.add_widget(name_label)

    location_label = MDLabel(
        text="Ubicación: " + event_info.get("ubicacion", "Ubicación desconocida"),
        size_hint_y=None,
        height="20dp",
        halign="center"
    )
    event_card.add_widget(location_label)

    description_label = MDLabel(
        text=event_info.get("descripcion", "Sin descripción"),
        size_hint_y=None,
        height="40dp",
        halign="center"
    )
    event_card.add_widget(description_label)

    countdown_label = MDLabel(
        text="Calculando tiempo restante...",
        size_hint_y=None,
        height="20dp",
        halign="center"
    )
    event_card.add_widget(countdown_label)

    event_datetime_str = f"{event_info.get('fecha')} {event_info.get('hora')}"
    event_datetime = datetime.strptime(event_datetime_str, "%Y-%m-%d %H:%M")

    Clock.schedule_interval(lambda dt: self.update_countdown(countdown_label, event_datetime), 1)

    event_card.bind(on_release=lambda x: self.show_event_details(event_info))

    container.add_widget(event_card)
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función update_countdown muestra a los usuarios el tiempo restante hasta el evento. Al desglosar el tiempo en días, horas, minutos y segundos, la función proporciona una cuenta regresiva clara y precisa.

- **Obtener la fecha y hora actuales}**
    - La función empieza obteniendo la fecha y hora actual utilizando “datetime.now()”.
        - Este valor representa el momento en el que la función es ejecutada, lo que nos permite calcular cuánto tiempo falta hasta el evento.

- **Calcular la diferencia de tiempo**
    - Se calcula la diferencia de tiempo entre el “target_datetime” (la fecha y hora del evento) y la fecha y hora actuales (now).
        - El resultado es un objeto “timedelta” que contiene la diferencia en días, segundos, y microsegundos.

- **Comprobar si el evento todavía está por ocurrir**
    - Si la diferencia en segundos es mayor que cero (remaining.total_seconds() > 0), significa que el evento aún no ha ocurrido, por lo que se continúa con el cálculo del tiempo restante.

- **Desglosar el tiempo restante**
    - Se extraen los días, horas, minutos y segundos de la diferencia de tiempo (remaining).
        - Para calcular las horas, minutos y segundos, se realiza una operación de división y módulo:
            - hours se obtiene dividiendo los segundos totales entre 3600 (el número de segundos en una hora).
            - minutes se obtiene dividiendo el residuo de los segundos entre 60 (el número de segundos en un minuto).
            - seconds es el residuo final después de calcular horas y minutos.

- **Actualizar el texto del label**
    - El label que se pasa como parámetro se actualiza con el tiempo restante en el formato Faltan: {días}d {horas}h {minutos}m {segundos}s.
        - Este texto se actualizará dinámicamente cada segundo gracias al uso de “Clock.schedule_interval”.

- **Evento ya comenzado o finalizado**

    - Si el evento ya ha comenzado o ha finalizado (es decir, si el tiempo restante es menor o igual a cero), el texto del label se actualiza para mostrar "¡El evento ya ha comenzado o finalizado!".

*Figura 24. Función de update_coutdown*

```
def update_countdown(self, label, target_datetime):
    now = datetime.now()
    remaining = target_datetime - now
    if remaining.total_seconds() > 0:
        days, seconds = remaining.days, remaining.seconds
        hours = seconds // 3600
        minutes = (seconds % 3600) // 60
        seconds = seconds % 60
        label.text = f"Faltan: {days}d {hours}h {minutes}m {seconds}s"
    else:
        label.text = "¡El evento ya ha comenzado o finalizado!"
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

Como resultado de la pantalla MainScreen, y las funciones de open_menu, search_events, add_event_card y update_countdown; y los diferentes diseños para los elementos de Kivy se visualiza de la siguiente manera:

*Figura 25. Resultado de la pantalla EventsScreen*


***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “show_inscritos_screen” es responsable de cambiar la vista actual de la aplicación a la pantalla de "Inscritos" y, al mismo tiempo, carga las inscripciones asociadas al usuario que está autenticado.

- **Cambiar la vista de la interfaz**
    - Para hacer esto, primero se cambia el estado de la interfaz de usuario utilizando self.sm.current, lo que permite mostrar la pantalla correspondiente a la vista de "Inscritos".
        - self.sm hace referencia al Screen Manager, y al establecer self.sm.current a la pantalla de "Inscritos", la aplicación muestra esa vista a los usuarios.

- **Cargar las inscripciones del usuario**
    - Después de cambiar a la pantalla de "Inscritos", se llama a la función cargar_inscripciones.
        - Esta función se encarga de obtener y mostrar todas las inscripciones previas que el usuario ha realizado a los eventos.
        - Esto asegura que la información de inscripciones esté actualizada y se refleje correctamente en la interfaz de usuario, proporcionando una experiencia dinámica y coherente al usuario autenticado.

*Figura 26. Función de show_inscritos_screen*

```
def show_inscritos_screen(self):
    self.sm.current = 'inscritoscreen'
    self.cargar_inscripciones()
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “show_event_details” está diseñada para mostrar los detalles completos de un evento cuando un usuario hace clic en una tarjeta de evento.

- **BoxLayout para la organización de la interfaz**
    - Se utiliza un **BoxLayout** con una orientación vertical para organizar los elementos de la interfaz, garantizando que los elementos se apilen de manera ordenada.
        - Este **BoxLayout** está contenido dentro de un **ScrollView**, lo que permite el desplazamiento si el contenido excede el tamaño visible de la pantalla.

- **Detalles del evento**
    - En la vista de detalles, se presentan los siguientes datos relacionados con el evento:
        - **Título:** muestra el nombre del evento.
        - **Imagen:** inserta una imagen asociada al evento (si está disponible).
        - **Descripción:** muestra una descripción detallada del evento (una versión extendida, descripcion2).
        - **Ubicación, fecha y hora:** proporciona información sobre la ubicación, fecha y hora del evento.

- **Botón de inscripción**
    - Al final de la vista de detalles, se incluye un botón con el texto **"Inscribirme"**.

- Este botón permite que el usuario se inscriba en el evento al hacer clic en él, lo que invoca la función **inscribirse_evento** para completar la inscripción.

- **Popup para mostrar los detalles**
    - Todo el contenido de los detalles del evento se muestra dentro de un Popup. Esto permite que los detalles se presenten de manera emergente sin que sea necesario cambiar la pantalla actual de la aplicación.

*Figura 27. Función de show_event_details*

```
def show_event_details(self, event_info):
    content = BoxLayout(orientation="vertical", padding=20, spacing=15, size_hint_y=None)
    content.bind(minimum_height=content.setter('height'))

    title_label = MDLabel(
        text=f"{event_info.get('nombre', 'Sin nombre')}",
        halign="center",
        size_hint_y=None,
        height="30dp"
    )
    content.add_widget(title_label)

    image = Image(
        source=event_info.get("imagen", ""),
        size_hint_y=None,
        height="200dp",
        allow_stretch=True
    )
    content.add_widget(image)

    desc_label = MDLabel(
        text=f"{event_info.get('descripcion2', 'Sin descripción')}",
        halign="center",
        size_hint_y=None,
        height="250dp"
    )
location_label = MDLabel(
    text=f"{event_info.get('ubicacion', 'Desconocida')}",
    halign="center",
    size_hint_y=None,
    height="30dp"
)
date_label = MDLabel(
    text=f"{event_info.get('fecha', 'Sin fecha')}",
    halign="center",
    size_hint_y=None,
    height="30dp"
)
time_label = MDLabel(
    text=f"{event_info.get('hora', 'Sin hora')}",
    halign="center",
    size_hint_y=None,
    height="30dp"
)

content.add_widget(desc_label)
content.add_widget(location_label)
content.add_widget(date_label)
content.add_widget(time_label)

scroll_view = ScrollView(size_hint=(1, None), size=(Window.width * 0.8, Window.height * 0.7))
scroll_view.add_widget(content)

    main_layout = BoxLayout(orientation="vertical", padding=10, spacing=10)
    main_layout.add_widget(scroll_view)

    inscribirme_button = MDRaisedButton(
        text="Inscribirme",
        size_hint=(None, None),
        size=("150dp", "40dp"),
        pos_hint={"center_x": 0.5}
    )
    inscribirme_button.bind(on_release=lambda x: self.inscribirse_evento(event_info))
    main_layout.add_widget(inscribirme_button)

    popup = Popup(title="Detalles del Evento", content=main_layout, size_hint=(0.9, 0.9))
    popup.open()
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código la función “inscribirse_evento” se encarga de gestionar el proceso de inscripción de un usuario a un evento dentro de la aplicación. Esta función recopila la información del evento y la del usuario, y luego guarda los datos de la inscripción en una base de datos en la nube (Firebase Real Time).

- **Obtención de datos del usuario**
    - **user_email = self.user_email:** esta línea obtiene el correo electrónico del usuario que está actualmente logueado en la aplicación.

    - **user_name = self.username:** similar al correo electrónico, aquí obtenemos el nombre del usuario, que también se asociará con su inscripción al evento.

- **Creación del objeto de inscripción**
    - **inscription_data = {...}**: Se crea un diccionario que contiene toda la información relevante de la inscripción, la cual incluye:
        - **evento_id:** El ID del evento, único para cada evento.
        - **nombre_evento:** El nombre del evento, para identificar el evento al que el usuario se inscribe.
        - **imagen_evento:** La imagen del evento, que se puede usar para mostrar visualmente el evento.
        - **usuario_email:** El correo electrónico del usuario, para asociar la inscripción con su cuenta.

- **usuario_nombre:** El nombre del usuario, para personalizar su inscripción.

- **Envío de la inscripción a la base de datos:**
  - **inscription_url = f"https://dbaplicacionmovil-default-rtdb.firebaseio.com/inscritos/{user_email.replace('.', '-')}.json":** Se define la URL de la base de datos de Firebase donde se guardará la inscripción. El correo electrónico del usuario se usa como clave única en la base de datos, pero los puntos (.) se reemplazan por guiones (-) para evitar problemas en la URL.
  - **response = requests.post(inscription_url, json=inscription_data):** Se envía la solicitud POST a la base de datos de Firebase Real Time, guardando los datos de la inscripción en la URL especificada.

- **Verificación del éxito de la inscripción**
  - **if response.status_code == 200::** Después de realizar la solicitud, se verifica si la respuesta de la base de datos fue exitosa.
    - Si la inscripción fue exitosa:
      - **print("Inscripción exitosa."):** Se imprime un mensaje en la consola indicando que el proceso de inscripción fue exitoso.

- **self.show_inscritos_screen():** Se llama a la función show_inscritos_screen() para cambiar a la pantalla donde el usuario puede ver todos los eventos a los que se ha inscrito.

- Si ocurre algún error durante el proceso:
  - **print("Error al inscribirse."):** En caso de que la inscripción no haya podido completarse, se imprime un mensaje indicando que hubo un problema.

*Figura 28. Función de inscribirse_evento*

```
def inscribirse_evento(self, event_info):
    """Guardar la inscripción del usuario al evento en la base de datos."""
    user_email = self.user_email
    user_name = self.username

    inscription_data = {
        "evento_id": event_info.get("id"),
        "nombre_evento": event_info.get("nombre"),
        "imagen_evento": event_info.get("imagen"),
        "usuario_email": user_email,
        "usuario_nombre": user_name,
    }

    inscription_url = f"https://dbaplicacionmovil-default-rtdb.firebaseio.com/inscritos/{user_email.replace('.', '-')}.json"
    response = requests.post(inscription_url, json=inscription_data)

    if response.status_code == 200:
        print("Inscripción exitosa.")
        self.show_inscritos_screen()
    else:
        print("Error al inscribirse.")
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

Como resultado de la pantalla EventsScreen, y las funciones de show_events_details, e inscribirse_evento; y los diferentes diseños para los elementos de Kivy se visualiza de la siguiente manera:

*Figura 29. Resultado de el detalle de los eventos*


***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código, la función cargar_inscripciones se encarga de recuperar y mostrar las inscripciones de un usuario a los eventos específicos desde la base de datos Firebase Real Time.

- **Verificación de la existencia del correo electrónico del usuario**
    - **if not self.user_email::** La función comprueba si el correo electrónico del usuario está disponible. Si no se encuentra un correo asociado al usuario logueado, se imprime un mensaje de error y se detiene la ejecución de la función:
    - **print("No hay correo de usuario."):** Este paso es crucial, ya que las inscripciones están asociadas al correo electrónico del usuario.

- **Construcción de la URL para la base de datos**
    - **email_key = self.user_email.replace('.', '-'):** Para asegurar que el correo electrónico sea compatible con la estructura de la URL en Firebase Real Time, se reemplazan los puntos (.) del correo por guiones (-), ya que la base de datos no permite ciertos caracteres en las claves.
    - **inscription_url = f"https://dbaplicacionmovil-default-rtdb.firebaseio.com/inscritos/{email_key}.json":** Se construye la URL completa que permite acceder a las inscripciones del usuario en la base de datos de Firebase Real Time.

- **Solicitud de los datos a Firebase**
    - **response = requests.get(inscription_url + '?auth=' + self.auth):** Se usa el método get de la librería requests para enviar una solicitud a la base de datos, solicitando los datos de las inscripciones del usuario.
        - **print("Respuesta de Firebase:", response.json()):** Se verifica que la solicitud fue exitosa, imprimiendo la respuesta de la base de datos, que contendrá las inscripciones del usuario en formato JSON.

- **Verificación de si hay inscripciones disponibles**
    - **if not inscripciones:** Después de recibir los datos de Firebase Real Time, la función comprueba si existen inscripciones. Si no hay datos, se muestra un mensaje indicando que no hay inscripciones disponibles:
        - **print("No hay inscripciones para mostrar."):** Si no hay inscripciones, se termina la ejecución de la función.

- **Preparación de la interfaz para mostrar las inscripciones**
    - **inscriptions_box = self.sm.get_screen('inscritoscreen').ids.inscriptions_box:** Se obtiene el contenedor donde se mostrarán las inscripciones en la interfaz de usuario. Esta caja se encuentra en la pantalla de inscripciones (inscritoscreen).
    - **inscriptions_box.clear_widgets():** Antes de agregar nuevas inscripciones, se limpia cualquier tarjeta de inscripción que pueda haber

quedado de una sesión anterior.

- **Creación de las tarjetas de inscripción**
  - **for inscription_id, inscription_info in inscripciones.items()::** Se recorre todas las inscripciones recuperadas de Firebase Real Time. Cada inscripción tiene un ID único (inscription_id) y un conjunto de datos asociados (inscription_info).
  - **self.add_inscription_card(inscriptions_box, inscription_info, inscription_id):** Para cada inscripción, se llama a la función add_inscription_card para crear una tarjeta de inscripción y agregarla al contenedor de inscripciones en la pantalla.

*Figura 30. Función de cargar_inscripciones*

```
def cargar_inscripciones(self):
    """Cargar las inscripciones del usuario desde Firebase y mostrarlas en la pantalla."""
    if not self.user_email:
        print("No hay correo de usuario.")
        return

    email_key = self.user_email.replace('.', '-')
    inscription_url = f"https://dbaplicacionmovil-default-rtdb.firebaseio.com/inscritos/{email_key}.json"
    print(f"Obteniendo datos de: {inscription_url}")

    response = requests.get(inscription_url + '?auth=' + self.auth)
    print("Respuesta de Firebase:", response.json())

    inscripciones = response.json()

    if not inscripciones:
        print("No hay inscripciones para mostrar.")
        return

    inscriptions_box = self.sm.get_screen('inscritoscreen').ids.inscriptions_box
    inscriptions_box.clear_widgets()

    for inscription_id, inscription_info in inscripciones.items():
        self.add_inscription_card(inscriptions_box, inscription_info, inscription_id)
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código, la función add_inscription_card se encarga de crear una tarjeta visualmente atractiva para cada inscripción de un usuario y agregarla a la pantalla de inscripciones en la aplicación.

- **Creación de la tarjeta**

  - La tarjeta se crea utilizando el componente MDCard, que es un widget de KivyMD (Kivy Material Design). La tarjeta tiene un diseño vertical (orientation="vertical") y un pequeño relleno interno para que el contenido no esté pegado a los bordes.

  - Se ajusta el tamaño de la tarjeta con las propiedades size_hint y size, asegurando que se vea correctamente en la pantalla. La propiedad elevation añade una sombra para darle un efecto de profundidad, lo que hace que la tarjeta resalte visualmente.

- **Imagen del evento**

  - **image_source = inscription_info.get("imagen_evento", ""):** Se obtiene la URL de la imagen asociada al evento desde la información de la inscripción.

- **image = Image(source=image_source, size_hint_y=0.6, allow_stretch=True, keep_ratio=True):** Si la imagen está disponible, se muestra en la parte superior de la tarjeta. Se ajusta el tamaño de la imagen a la mitad de la altura de la tarjeta, permitiendo que se estire pero manteniendo la relación de aspecto para que no se distorsione.

- **Nombre del evento**
  - **name_label = MDLabel(text=inscription_info.get("nombre_evento", "Evento sin nombre"), font_style="H6", halign="center"):** Se crea una etiqueta (MDLabel) para mostrar el nombre del evento. Si no se encuentra un nombre, se muestra un texto predeterminado como "Evento sin nombre". La etiqueta está alineada al centro de la tarjeta y se le da un estilo de texto H6 para asegurar que tenga un tamaño adecuado y se vea claro.

- **Botón para cancelar la inscripción**
  - **cancel_button = MDRaisedButton(text="Cancelar Inscripción", md_bg_color=(0.0, 0.6, 0.2, 1), on_release=lambda x: self.cancelar_inscripcion(inscription_info, inscription_card, inscription_id)):** Se agrega un botón MDRaisedButton que permite al usuario cancelar su inscripción al evento. Este botón tiene un color verde y, al presionarlo, se llama a la función cancelar_inscripcion para gestionar la cancelación.
  - **on_release=lambda x: self.cancelar_inscripcion(...)):** Al hacer clic en

el botón, se llama a la función cancelar_inscripcion pasando la información de la inscripción, la tarjeta y el ID de la inscripción para poder procesar la cancelación.

- **Añadir la tarjeta al contenedor**
    - **container.add_widget(inscription_card):** Esta línea agrega la tarjeta recién creada al contenedor en la pantalla, haciendo que se visualice al usuario en la interfaz de la aplicación.

*Figura 31. Función de add_inscription_card*

```
def add_inscription_card(self, container, inscription_info, inscription_id):
    """Añadir tarjeta de inscripción a la pantalla de inscritos."""
    inscription_card = MDCard(
        orientation="vertical",
        padding="5dp",
        size_hint=(None, None),
        size=("155dp", "155dp"),
        elevation=4,
        pos_hint={"center_x": 0.5}
    )

    image_source = inscription_info.get("imagen_evento", "")
    if image_source:
        image = Image(source=image_source, size_hint_y=0.6, allow_stretch=True, keep_ratio=True)
        inscription_card.add_widget(image)

    name_label = MDLabel(
        text=inscription_info.get("nombre_evento", "Evento sin nombre"),
        font_style="H6",
        size_hint_y=None,
        height="30dp",
        halign="center"
    )
    inscription_card.add_widget(name_label)

    cancel_button = MDRaisedButton(
        text="Cancelar Inscripción",
        size_hint_y=None,
        height="40dp",
        md_bg_color=(0.0, 0.6, 0.2, 1),
        on_release=lambda x: self.cancelar_inscripcion(inscription_info, inscription_card, inscription_id),
        pos_hint={"center_x": 0.5}
    )

    inscription_card.add_widget(cancel_button)

    container.add_widget(inscription_card)
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código, la función cancelar_inscripcion se encarga de eliminar una inscripción de un usuario a un evento, tanto de la interfaz de la aplicación como de la base de datos Firebase Real Time.

- **Eliminar la tarjeta de la interfaz**
    - **inscription_card.parent.remove_widget(inscription_card):** Esta línea elimina la tarjeta visual correspondiente a la inscripción de la interfaz de usuario, asegurando que la tarjeta ya no se muestre al usuario una vez que la inscripción ha sido cancelada. Esto hace que la interfaz sea más limpia y actualizada, reflejando el cambio inmediatamente.

- **Verificación del correo del usuario**
    - **if not self.user_email::** Antes de proceder con la cancelación, se verifica que el correo del usuario esté disponible. Si no se encuentra el correo, se muestra un mensaje en la consola indicando que la cancelación no puede realizarse sin la información del usuario. Esta verificación asegura que solo los usuarios autenticados puedan cancelar sus inscripciones.

- **Formatear el correo para Firebase**
    - **email_key = self.user_email.replace('.', '-'):** Dado que Firebase Real Time no permite el uso de puntos (.) en las claves de su base de datos, el

correo del usuario se ajusta reemplazando los puntos por guiones para que sea compatible con la estructura de la base de datos.

- **Construcción de la URL para eliminar la inscripción**
    - **inscription_url = f"https://dbaplicacionmovil-default-rtdb.firebaseio.com/inscritos/{email_key}/{inscription_id}.json":** Se construye la URL para acceder y eliminar la inscripción específica en la base de datos. Se utiliza el correo del usuario (formateado) como clave y el ID de la inscripción para apuntar directamente a la ubicación de la inscripción dentro de la base de datos.

- **Realizar la solicitud de eliminación**
    - **response = requests.delete(inscription_url + '?auth=' + self.auth):** Se realiza una solicitud HTTP DELETE a la URL de la base de datos para eliminar la inscripción del evento. Se incluye un token de autenticación (auth) en la solicitud, que asegura que solo el usuario autorizado pueda realizar esta acción.

- **Verificar la respuesta de la solicitud**
    - **if response.status_code == 200::** Si la solicitud de eliminación es exitosa (código de respuesta 200), se imprime un mensaje en la consola indicando que la inscripción ha sido cancelada correctamente.

- **else::** Si la solicitud falla, se imprime el código de error y el mensaje de respuesta de Firebase Real Time, lo que ayuda a identificar qué salió mal durante la operación de cancelación.

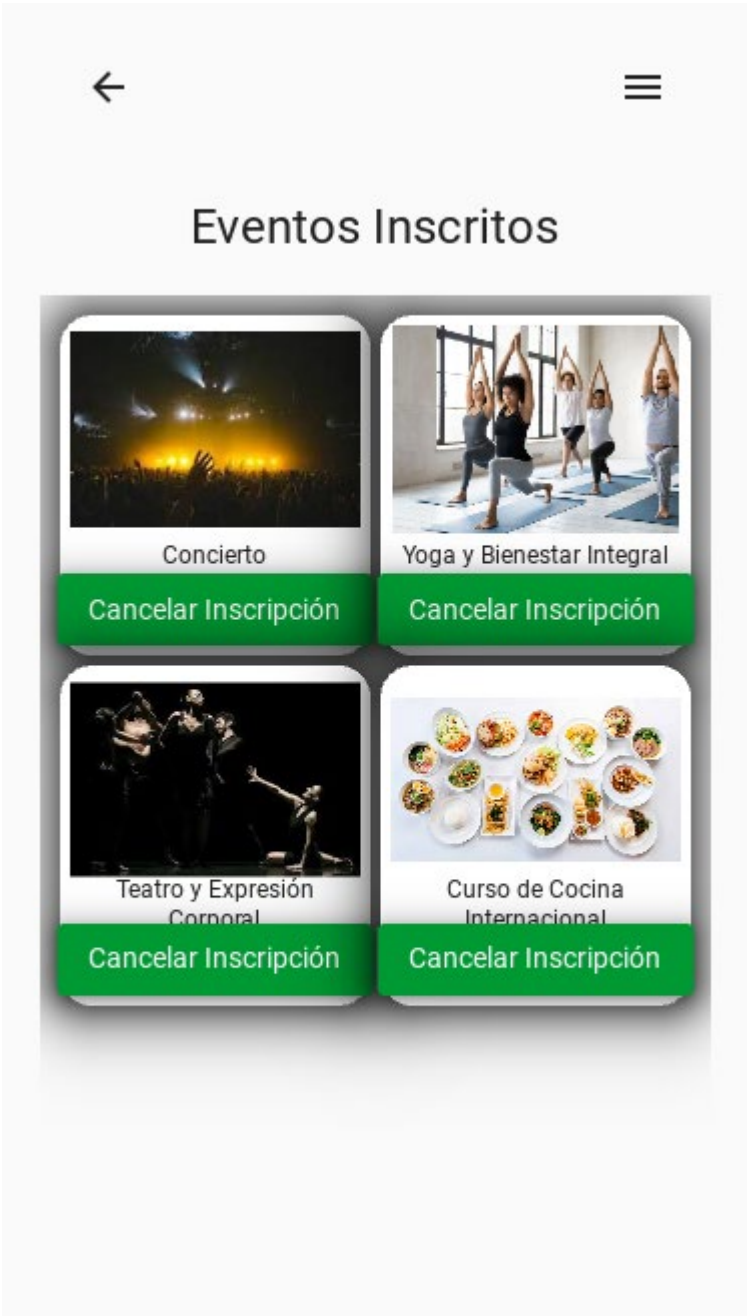
*Figura 32. Función de cancelar_inscripcion*

```
def cancelar_inscripcion(self, inscription_info, inscription_card, inscription_id):
    """Cancelar la inscripción: eliminar tarjeta y actualizar Firebase."""
    inscription_card.parent.remove_widget(inscription_card)

    if not self.user_email:
        print("No se puede cancelar inscripción. No se ha encontrado el correo del usuario.")
        return

    email_key = self.user_email.replace('.', '-')
    print(f"Correo formateado: {email_key}")

    inscription_url = f"https://dbaplicacionmovil-default-rtdb.firebaseio.com/inscritos/{email_key}/{inscription_id}.json"
    print(f"URL de eliminación: {inscription_url}")

    response = requests.delete(inscription_url + '?auth=' + self.auth)

    if response.status_code == 200:
        print(f"Inscripción del evento con ID {inscription_id} cancelada con éxito.")
    else:
        print(f"Error al eliminar inscripción. Código de respuesta: {response.status_code}")
        print(f"Respuesta de Firebase: {response.text}")
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

Como resultado de la pantalla InscritosScreen, y las funciones de cargar_inscripciones, add_inscription_card y cancelar_inscripcion; y los diferentes diseños para los elementos de Kivy se visualiza de la siguiente manera:

*Figura 33. Resultado de la pantalla InscritosScreen*




***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

En el siguiente fragmento de código, se inicia la ejecución de la aplicación LoginApp cuando el script es ejecutado directamente:

- **if __name__ == '__main__'**
    - Esta línea asegura que el bloque de código a continuación solo se ejecute si el script es ejecutado directamente. __name__ es una variable especial en Python que contiene el nombre del módulo actual. Si el script se ejecuta directamente (no es importado como módulo en otro script), su valor será '__main__'. Esto es útil para asegurarse de que ciertos bloques de código solo se ejecuten cuando se corre el archivo como programa principal y no cuando se importa como parte de otro programa.

- **LoginApp().run()**
    - **LoginApp():** Esta línea crea una instancia de la clase LoginApp, que es la clase principal de la aplicación Kivy. Esta clase define la interfaz de usuario, las pantallas, y la lógica que gobierna la aplicación.
    - **.run():** Luego, se llama al método run() de la clase LoginApp, el cual inicia el ciclo de eventos de Kivy. Este ciclo mantiene la aplicación en ejecución, gestionando las interacciones con la interfaz de usuario, las actualizaciones de la pantalla y las respuestas del usuario hasta que el usuario cierre la ventana de la aplicación.

*Figura 34. Ejecución de la Aplicación*

```
if __name__ == '__main__':
    LoginApp().run()
```

***Nota**. Para más información dirigirse al link adjunto en el anexo 1.*

## *Cronograma*

*Tabla 1. Cronograma de actividades*

| Objetivo | Actividades |
|---|---|
| **Realizar un análisis exploratorio de la gestión y comunicación de información sobre actividades extracurriculares en las instituciones de educación superior.** | Análisis comparativo de herramientas y plataformas. |
| | Revisión de la literatura sobre la gestión de eventos extracurriculares. |
| | Sistematización de resultados. |
| | |
| **Prototipar una base de datos local simulada que imite el funcionamiento de una solución en la nube en Firebase.** | Definición de tablas y relaciones. |
| | Creación y configuración de la instancia en Firebase. |
| | Inserción de datos de prueba. |
| | |
| **Desarrollar una versión piloto del aplicativo, incluyendo la interfaz de usuario para las diferentes funcionalidades para su prueba y evaluación.** | Diseño y construcción de pantallas clave. |
| | Implementación de funcionalidades. |
| | Lanzamiento piloto y recolección de retroalimentación. |

***Nota**. Elaboración propia.*

## Análisis de costos

Para asegurar la viabilidad económica y la rentabilidad del proyecto, se evaluarán los costos necesarios para fabricar el producto. A continuación, se presentan los distintos grupos de costos a considerar:

*Tabla 2. Análisis de costos*

| Categoría | Descripción | Tipo de costo | Precio (COP) | Total anual (COP) |
|---|---|---|---|---|
| **Costos Directos** | Mano de obra: Python | Variable | 50.000/hora | 72 millones (6 millones/mes) |
| | Mano de obra: Firebase | Variable | 50.000/hora | |
| | Mano de obra: Tester | Variable | 50.000/hora | |
| **Costos Fijos** | Hosting de servidores | Fijo | 20.000/mes * 3 personas | 43.747.900 |
| | 2 PC de alto rendimiento | Fijo | 5.000.000 * PC * 3 personas | |
| | Licencia Firebase | Fijo | 1.300.000 | |
| | Servicio de internet | Fijo | 150.000/ mes por persona * 3 personas | |
| | Servicio de luz | | 100.000/persona/2 meses * 3 personas | |
| | Licencias comerciales | Fijo | 7.900/anual | |
| | Seguros de equipo (contra todo riesgo) | Fijo | 5.000.000 | |

| | | | | |
|---|---|---|---|---|
| **Gastos Generales** | Administración del proyecto | Overhead | 80.000/día | 29.700.000 |
| | Demostraciones (4 eventos) | Overhead | 4.000.000/evento | |
| | Capacitación (4 sesiones) | Overhead | 1.500.000/evento | |
| | Imprevisto | Overhead | 5.000.000 | |

***Nota***. *Elaboración propia.*

## Totales Generales

*Tabla 3. Total de costos*

| Categoría | Costo Anual (COP) |
|---|---|
| **Costos Directos** | 72.000.000 |
| **Costos Fijos** | 43.747.900 |
| **Gastos Generales** | 29.700.000 |
| **Total General** | **145.447.900 (24 cuotas = 6.061.579 mensual) (estimando un total de 200 usuarios el valor de cada uno sería de 30.307 COP)** |

***Nota***. *Elaboración propia.*

**Conclusiones**

El presente proyecto de grado ha logrado cumplir satisfactoriamente con el objetivo general de desarrollar un aplicativo móvil utilizando el lenguaje de programación Python, con el fin de facilitar el acceso a la información sobre eventos extracurriculares en instituciones de educación superior. Esta herramienta busca optimizar la gestión y comunicación de estos eventos, promoviendo la participación estudiantil y mejorando la interacción entre los organizadores y los participantes.

A lo largo del desarrollo, se alcanzaron los objetivos específicos planteados. En primer lugar, se realizó un análisis exploratorio sobre la gestión de eventos extracurriculares en las instituciones educativas, lo cual permitió identificar las principales problemáticas y áreas de oportunidad en cuanto a la visibilidad y comunicación de los mismos. Esta fase fue fundamental para orientar el diseño y las funcionalidades del aplicativo.

Posteriormente, se llevó a cabo el prototipo de una base de datos local simulada, replicando el funcionamiento de una solución en la nube mediante Firebase. Esta base de datos permitió validar la integridad y sincronización de los datos dentro de la aplicación, garantizando una solución escalable y robusta a futuro.

Finalmente, se desarrolló una versión piloto del aplicativo móvil, que incluyó la creación de una interfaz de usuario intuitiva y la implementación de las funcionalidades necesarias para el acceso a la información de eventos y la interacción con los usuarios. La prueba y evaluación de esta versión piloto demostraron que el aplicativo cumple con las

expectativas iniciales, y su uso puede ser extendido en el futuro para cubrir una mayor cantidad de eventos y ofrecer nuevas funcionalidades.

## Referencias

1. Abdulsalam, Y. S., & Hedabou, M. (2021). Security and privacy in cloud computing: technical review. Future Internet, 14(1), 11.
2. Alias Mendoza, R., & Santos del Carpio, G. (2013). Página web y aplicación en Android para el módulo de inscripción y acreditación del departamento de actividades extraescolares del Instituto Tecnológico de Tuxtla Gutiérrez.
3. Almalki, S. A., Almojali, A. I., Alothman, A. S., Masuadi, E. M., & Alaqeel, M. K. (2017). Burnout and its association with extracurricular activities among medical students in Saudi Arabia. International journal of medical education, 8, 144.
4. Almenara, J. C. (2007). Las necesidades de las TIC en el ámbito educativo: oportunidades, riesgos y necesidades. Tecnología y comunicación educativas, 21(45), 5-19.
5. Ambriz, M. L. S. (2011). El uso del celular para desarrollar el pensamiento crítico, reflexivo y analítico. Etic@ net: Revista científica electrónica de Educación y Comunicación en la Sociedad del Conocimiento, (11), 196-212.
6. Anncode. (2018). Conoce los lenguajes de programación para Android. Platzi. https://platzi.com/blog/lenguajes-programacion-android/
7. Anncode. (2018). Conoce los lenguajes de programación para Android.
8. Balseca Pallasco, F. M. (2017). Actividades extracurriculares (Bachelor's thesis, Latacunga: Universidad Técnica de Cotopaxi; Facultad de Ciencias Humanas y

Educación; Licenciatura en Educación Básica).

9. Bayerlein, L., Dean, B. A., Perkiss, S., & Jeske, D. (2021). Using technology platforms for work-integrated learning. In Advances in research, theory and practice in work-integrated learning (pp. 239-248). Routledge.
10. Blanco Espinosa, C. O., & Madrid Plata, J. L. (2014). Implementación de una aplicación web para la gestión de eventos académicos de la Universidad Pontificia Bolivariana seccional Bucaramanga.
11. Bond, M., Buntins, K., Bedenlier, S., Zawacki-Richter, O., & Kerres, M. (2020). Mapping research in student engagement and educational technology in higher education: A systematic evidence map. International journal of educational technology in higher education, 17, 1-30.
12. Chaparro Africano, A. F. (2022). La compleja, difícil y necesaria articulación entre las actividades curriculares y extracurriculares.
13. CHIU, C. C. (2015). Las pruebas en el desarrollo de software. Universidad Nacional Autónoma de México.
14. De la Riva, D., Di Cicco, C., Montero, F., & Sottile, S. (2012). Proyecto UniMóvil: una aplicación móvil para Universidades. XVIII Congreso argentino deficiencias de la computación. Recuperado el.
15. Debernardi, F. (2021). ¿Qué es el diseño de la interfaz de usuario? https://es.linkedin.com/pulse/qu%C3%A9-es-el-dise%C3%B1o-de-la-interfaz-usuario-dise%C3%B1ador-ui-ux
16. Diaz Diaz, S. M. (2014). Pruebas de seguridad en aplicaciones web como imperativo en la calidad de desarrollo del software

17. Díaz, C. T., & Díaz, E. T. (2018). Uso y consumo de las aplicaciones móviles en el Smartphone como herramienta de apoyo académico. Espacios, 39(30), 18.
18. Dimitrios, T. (2022). Privacy and Data Protection in mobile applications.
19. Enriquez Ayala, N. M., & Villagómez Bardellini, E. J. (2021). “Desarrollo de una aplicación móvil informativa en tiempo real para el Centro de Propaganda y Comunicación Social de la Universidad Técnica de Cotopaxi Extensión La Maná” (Bachelor's thesis, Ecuador: La Maná: Universidad Técnica de Cotopaxi (UTC)).
20. Firebase Realtime database. (s. f.). Firebase. https://firebase.google.com/docs/database
21. Firebase. (s. f.). Firebase. https://firebase.google.com/
22. González-Granadillo, G., González-Zarzosa, S., & Diaz, R. (2021). Security information and event management (SIEM): analysis, trends, and usage in critical infrastructures. Sensors, 21(14), 4759. *https://marutitech.com/7-trends-of-mobile-application-development/*
23. Huawei. (2023). Cloud computing technology. *https://doi.org/10.1007/978-981-19-3026-3*
24. IBM. (s. f.). Introducción al desarrollo de aplicaciones móviles. *https://www.ibm.com/mx-es/topics/mobile-application-development*
25. Johnson, J. (2020). Designing with the mind in mind: simple guide to understanding user interface design guidelines. Morgan Kaufmann.
26. Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A. M., & Luo, B. (2023). SQL and NoSQL database software architecture performance analysis and assessments—

a systematic literature review. Big Data and Cognitive Computing, 7(2), 97.

27. Kivy: Cross-platform Python Framework for NUI. (s. f.-b)
28. Krug, S. (2014). Don't make me think, Revisited. A Common Sense Approach to Web and Mobile Usability.
29. Lee, S. M., Kim, N. R., & Hong, S. G. (2017). Key success factors for mobile app platform activation. Service Business, 11, 207-227.
30. Levy, J. (2015). UX strategy: How to devise innovative digital products that people want. " O'Reilly Media, Inc.".
31. Lis Santofimio, J. D. (2021). Desarrollo de una aplicación web para la organización y registro de eventos para la Institución Universitaria Politécnico Grancolombiano (Bachelor's thesis, Ingeniería de Sistemas).
32. Machuca, R., Sánchez, E., Romero, E., Feliz, D., Villanueva, A., Feliz, E., Rojas, S., & Alvarado, J. (2024). Propuesta de proyecto P Gestion de Eventos Universitarios - Asignatura: Proyecto Final TDS Docente: - Studocu. https://www.studocu.com/latam/document/instituto-tecnologico-de-las-americas/administracion-de-proyectos-de-software/propuesta-de-proyecto-p-gestion-de-eventos-universitarios/95234012
33. Mackaway, J., & Chalkley, T. (2021). Student access and equity in work-integrated learning: A work in progress. In Advances in Research, Theory and Practice in Work-Integrated Learning (pp. 227-238). Routledge.
34. Manning, K. (2017). Organizational theory in higher education. Routledge.
35. Morrison, M. (2023). The perceptions of adolescent girls of the relationship between sports based extra curricular activities and their wellbeing (Doctoral

dissertation, University of Birmingham).

36. Nandaniya, H. (2024). Top 17 Mobile App Development Trends to Know in 2024
37. Oracle. (2020). What Is a Database? https://www.oracle.com/co/database/what-is-database
38. Pan, Y. H., Qu, T., Wu, N. Q., Khalgui, M., & Huang, G. Q. (2021). Digital twin based real-time production logistics synchronization system in a multi-level computing architecture. Journal of Manufacturing Systems, 58, 246-260.
39. Parast, F. K., Sindhav, C., Nikam, S., Yekta, H. I., Kent, K. B., & Hakak, S. (2022). Cloud computing security: A survey of service-based models. Computers & Security, 114, 102580.
40. Porto, J. P., & Gardey, A. (2023). Extracurricular - Qué es, beneficios, definición y concepto. Definición.de. https://definicion.de/extracurricular/
41. Sammel, A. (2014). The pedagogical implications of implementing new technologies to enhance student engagement and learning outcomes. Creative Education, 5(02), 104.
42. Santos, N. M., Silva, L. M., Leitão, J., & Preguiça, N. (2023). Data Management for mobile applications dependent on geo-located data. In Proceedings of the 10th Workshop on Principles and Practice of Consistency for Distributed Data (pp. 70-76).
43. Sarmiento, P. A. Q., & Guerrero, C. S. (2021). La Computación en la Nube en el proceso formativo en Programación Web. RISTI: Revista Ibérica de Sistemas e Tecnologias de Informação, (42), 10-19.
44. Serrano, D. R., Dea-Ayuela, M. A., Gonzalez-Burgos, E., Serrano-Gil, A., &

Lalatsa, A. (2019). Technology-enhanced learning in higher education: How to enhance student engagement through blended learning. European Journal of Education, 54(2), 273-286.

45. Singh, J., Das, D., Kumar, L., & Krishna, A. (2023). Mobile Application Development: Practice and Experience. Springer, Singapore.
46. Spillner, A., & Linz, T. (2021). Software Testing Foundations: A Study Guide for the Certified Tester Exam-Foundation Level-ISTQB® Compliant. dpunkt. verlag.
47. *The Python tutorial*. (s. f.). Python Documentation. https://docs.python.org/3/tutorial/
48. Torres, R. E. M., & Silva, I. Z. (2017). IMPACTO DE COMPETENCIAS TRANSVERSALES EN LA FORMACIÓN DE LÍDERES CUANDO PARTICIPAN EN EVENTOS EXTRACURRICULARES. ANFEI Digital, (7).
49. UNIR FP. (s.f.). Framework: qué es, para qué sirve y algunos ejemplos. *https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/*
50. Verona-Marcos, S., Pérez-Díaz, Y., Torres-Pérez, L., Delgado-Dapena, M. D., & Yáñez-Márquez, C. (2016). Pruebas de rendimiento a componentes de software utilizando programación orientada a aspectos. Ingeniería Industrial, 37(3), 278-285.

**Anexos**

Se ha utilizado un repositorio en GitHub para gestionar el código y la documentación del proyecto. Actualmente, el repositorio es privado. Para obtener acceso, puede enviar una solicitud al propietario a través del siguiente enlace: https://github.com/LauraP30/Aplicaci-n-Eventos-Extracurriculares