DISEÑO DE FRAMEWORK ACOPLADO A ANDROID STUDIO PARA SOPORTAR LA GESTIÓN DE LA INFORMACIÓN DE LAS ENTIDADES QUE HACEN PARTE DE LA LÓGICA DEL NEGOCIO

FRAMEWORK DESIGN INTEGRATED WITH ANDROID STUDIO TO SUPPORT THE INFORMATION MANAGEMENT OF ENTITIES LIKE A PART OF THE BUSINESS LOGIC

Jorge Albeiro Rivera Rosero [1]
Héctor Andrés Mora Paz [2]

Resumen

El objetivo central del artículo es establecer las características esenciales de un framework para el desarrollo de aplicaciones móviles que soporten la recolección y almacenamiento de datos a través de formularios mediante operaciones *Create, Read, Update y Delete (CRUD)*. Este proceso se realizó mediante el diseño, aplicación y análisis de resultados de un instrumento de recolección de información, con lo cual se abordó la definición estructural del framework. Como resultados, se logró definir las relaciones entre las representaciones de los tipos de datos en las 3 capas de software (Presentación, Lógica de Negocio y Acceso a Datos). Adicionalmente, se consolida una tabla de equivalencias y finalmente, se modeló la operación del futuro plugin. Se concluye que la aplicación del instrumento muestra la diversidad de configuraciones requeridas en la tabla de equivalencias dependiendo de las necesidades del contexto, por lo tanto, la herramienta a construir debe soportar esta flexibilidad.

Palabras clave: Android, automatización CRUD, Gestión de información, lógica de negocios.

Abstract

The central objective of the paper is to establish the essential characteristics of a framework for the development of mobile applications that support the collection and storage of data through forms using Create, Read, Update and Delete (CRUD) operations. This process was carried out through the design, application and analysis of results of an information collection instrument, with which the structural definition of the framework was addressed. As results, it was possible to define the relationships between the representations of the data types in the 3 software layers (Presentation, Business Logic and Data Access). Additionally, an equivalence table is consolidated and finally, the operation of the future plugin was modeled. It is concluded that the application of the instrument shows the diversity of configurations required in the equivalence table depending on the needs of the context, therefore, the tool to be built must support this flexibility.

Recepción: 10 de septiembre de 2022/ Evaluación: 20 de octubre de 2022 / Aprobado: 15 de noviembre de 2022

[1] Programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad CESMAG, San Juan de Pasto Colombia. Miembro del grupo de investigación Tecnofilia. Email: jarivera1@unicesmag.edu.co.
. ORCID: https://orcid.org/0000-0002-0092-8226.
[2]Programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad CESMAG, San Juan de Pasto Colombia. Miembro del grupo de investigación Tecnofilia. Email: hamora@unicesmag.edu.co. ORCID: https://orcid.org/0000-0003-3097-4757.

Keywords: Android, CRUD automation, information management, business logic.

Introducción

La demanda de desarrollo de aplicaciones que se ejecuten sobre el sistema operativo Android está directamente ligada al impacto de las TIC que incluye en sus herramientas de acceso principal a los dispositivos móviles(Dadgar & Joshi, 2018), de los cuales a nivel mundial en su mayoría utilizan el sistema operativo ya nombrado. Es importante resaltar que los dispositivos móviles, por su fácil manejo, portabilidad, sus altas capacidades de procesamiento y almacenamiento entre otras características, se han convertido en una herramienta de uso frecuente en diferentes entornos y actividades (Chandrashekar et al., 2021) como: bancarias, redes sociales, comunicación, servicios sanitarios, encuestas , la caracterización de población, la aplicación de encuestas, entre otras, (Ozgur et al., 2021) (Cucciniello et al., 2021) (Huang et al., 2022), que tienen como proceso principal, la recolección de información a través de formularios.

Estos formularios son construidos mediante un proceso de desarrollo monótono, el cual causa alto consumo de recursos cuando no se usan herramientas para el apoyo a la generación automática de componentes. No obstante, las herramientas existentes generan componentes bajo una sola configuración y disposición, que pueden no estar relacionadas con el flujo de trabajo de una entidad en concreto. Este problema ha sido tratado por (Fujita & Herrera Viedma, 2018) (Núñez et al., 2020), donde se generan componentes de interfaz de usuario y base de datos a través de clases tipo entidad, basados en las técnicas propuestas por la herramienta *Android SQLite Creator* y la metodología de Modelo conducido por ingeniería (Dar & Iqra, 2016). En estos estudios se generan interfaces CRUD asociadas a un modelo descrito por una clase, con una disposición de interfaces fijas muy similar a las que se proponen en frameworks web y móviles (Fujita & Herrera Viedma, 2018)(Andersen et al., 2019). El problema de estas disposiciones es que requieren de una serie de pasos con un costo importante en las curvas de aprendizaje de los desarrolladores(Espinoza Pereda, 2020).

La presente investigación propone una tabla de equivalencias configurable para flexibilizar el framework, como mejora para sortear la disposición fija de interfaces y aplanar la curva de aprendizaje a la que se enfrentan los desarrolladores. Esto, permite generar componentes de interfaz gráfica de usuario y almacenamiento persistente, desde un solo punto de configuración por tipo de datos, incrementando la posibilidad de generar diferentes disposiciones de interfaces gráficas de usuario y entidades de base de datos.

Materiales y métodos

A continuación se exponen las herramientas utilizadas en el desarrollo de la presente investigación seguido de las fases metodológicas.

Materiales

Para el desarrollo de este estudio se examinó la documentación de SQLite(Datatypes In SQLite, s/f), los lenguajes de programación Kotlin y Java(Basic types | Kotlin, s/f; Primitive Data Types (The JavaTM Tutorials > Learning the Java Language > Language Basics), s/f), y los controles de Interfaz Gráfica de Usuario (GUI) soportados por el sistema Operativo Android(Interfaz de usuario y navegación | Desarrolladores de Android | Android Developers, s/f) para determinar las representaciones de los atributos de una entidad. Seguido a ello, se identificaron los sectores económicos DANE(categoria economicas, s/f) donde generalmente se requiere gestión de información a través de formularios desplegados en

aplicaciones móviles, para detectar preferencias en el diseño de GUI acordes al flujo de trabajo, procesos y funciones específicas de cada sector. Para ello se diseñaron dos propuestas utilizando los softwares Microsoft Vicio y Balsamiq Mockups.

Conforme con lo anterior, se implementó y aplicó una encuesta usando Google Forms[3], como instrumento de recolección de información dirigida a desarrolladores, cuyos resultados, analizados con el apoyo de hojas de cálculo, orientaron el diseño del framework objeto de este estudio.

A continuación se detalla cada una de las fases de la metodología empleada en esta investigación.

Metodología

Para el desarrollo de la fase de diseño estructural del framework, se tomó como guía los criterios expuestos en (Quijano Vodniza, 2009). Estos criterios han configurado 4 fases de investigación, que consisten en primera instancia de un diseño del instrumento donde se seleccionó el tipo de instrumento y se estructuraron las preguntas, en segunda instancia se sometió el instrumento, a la validación por parte de expertos en áreas relacionadas al propósito de la investigación; en tercera instancia se seleccionaron los grupos focales y se aplicó el instrumento, y en última instancia se analizaron los datos recolectados y se diseñó el framework.

Cada una de las fases enunciadas se detallan en las siguientes secciones.

Diseño de instrumento

En esta fase se seleccionó como instrumento de recolección a la encuesta debido a que (Casas Anguita et al., s/f; Córdoba & 2006, s/f)muestra a este instrumento capaz de ofrecer una comprensión más precisa e imparcial del problema a resolver. Luego, por tratarse del desarrollo de un framework se tomó como grupo focal a programadores de software que trabajan en la región. Al tener estos enfoques, se diseñó el instrumento teniendo en cuenta la arquitectura en 3 capas a saber: Capa de presentación, Lógica de Negocios y Acceso a datos (Valle et al., s/f). La tabla I muestra una relación entre las capas, número de preguntas por capa, preguntas relacionadas y objetivo de las preguntas. El instrumento completo de este estudio puede visualizarse en [4].

Capa	Número de preguntas	Preguntas	Objetivo de las preguntas
Presentación	16	[11 - 12, 29 - 42]	Determinar los componentes de interfaz gráfica, su distribución y los tipos de datos que deben soportar para recolección de información
Lógica de Negocios	32	[11 - 42]	Determinar el comportamiento de las interfaces y la interacción en el intercambio de datos entre la GUI y el almacenamiento de los datos
Acceso a datos	30	[13 - 42]	Determinar las representaciones más frecuentes de atributos y sus tipos de datos

Tab. I Relación de capas y preguntas de la encuesta.
Fuente: Esta investigación

Es importante tener en cuenta que diferentes preguntas aportan a establecer requerimientos en más de una capa. Nótese que la tabla I no relaciona las preguntas con las

etiquetas 1 a 10, puesto que tienen fines de verificación de autorización (pregunta 2), obtención de datos personales básicos (preguntas 1,3,4) y caracterización de los participantes (preguntas 5 a 10), incluidas con el fin de cumplir con lo establecido en el componente ético del proyecto.

Validación de instrumentó

El instrumento de recolección de información fue construido en dos iteraciones, en primera instancia un banco de preguntas, el cual fue refinado y consolidado para obtener la versión inicial del instrumento, esta versión fue implementada en Google Forms, que se sometió a la evaluación por parte de 6 expertos en desarrollo de aplicaciones, diseño y administración de bases de datos e Ingeniería de Software, pertenecientes al cuerpo de docentes del programa de Ingeniería de Sistemas de la Universidad CESMAG, Universidad de Nariño y Universidad Mariana, de los cuales, 4 presentaron recomendaciones de ajuste al instrumento y 2 aprobaron el instrumento en primera instancia. En la siguiente iteración se implementaron los ajustes recomendados y posterior a ello se realizó una segunda validación con los 6 expertos quienes aprobaron el instrumento de recolección.

Aplicación

Una vez validado y consolidado el instrumento de recolección de información, se identificó un grupo focal, consultando la voluntad de participación en esta etapa de la investigación, finalmente se procedió a compartir la encuesta mediante mensaje de correo electrónico.

Para definir el grupo focal de encuestados, se realizó una invitación a posibles interesados de diferentes contextos de la industria de desarrollo de software, procurando que, en su mayoría, tengan interés en desarrollo de aplicaciones móviles para el sistema operativo Android. Se obtuvo la participación de 31 desarrolladores que pertenecen a entidades como: Universidad de Nariño, Universidad CESMAG, Corporación Universitaria Autónoma dé Nariño, Universidad Mariana, Tekknika kor SAS, Tory tech, Collect Center y Chevrolet – Autodenar. Es importante tener en cuenta que la voluntad manifiesta para participar como encuestados en el proyecto de investigación, es directamente del desarrollador.

Analisis de datos y diseño del framework

Para el análisis de datos, se construyó una hoja de cálculo con la consolidación de los mismos, los cuales, se organizaron para generar diferentes gráficas estadísticas como: barras, pastel y barras apiladas. De igual manera, se cruzaron variables de preguntas que tienen capas relacionadas.

Una vez sintetizada la opinión del grupo focal, se inspeccionaron arquitecturas de frameworks orientados a la automatización en la generación de componentes que abordan la lógica de negocios de una aplicación, con lo que se encontraron elementos estructurales que favorecen esta tarea como es la generación de GUI y tablas de base de datos a partir de entidades de clase; y otros elementos desfavorables como la dificultad de flexibilizar los componentes de GUI, factor que fue fundamental como oportunidad de mejora al estado del arte, conceptualizando una tabla de equivalencias, propuesta en el presente estudio.

En la sección siguiente se detalla más a fondo los resultados de este análisis y diseño del framework para la generación de componentes de lógica de negocios.

Resultados

Diseño de la encuesta

La información obtenida con el instrumento de recolección de información permitió conocer el contexto de la población; su caracterización, sexo, ocupación y sector productivo; de igual forma en lo concerniente a las preguntas relacionadas con las capas de aplicación, lógica de negocios y acceso a datos se determinaron las tendencias de desarrollo tales como el tipo de desarrollo empleado, las tecnologías de desarrollo, la disposición típica empleada en la construcción para la gestión de formularios (CRUD), la frecuencia de uso de componentes de interfaz gráfica en aplicaciones Android, la cantidad de aplicaciones que necesitan almacenamiento persistente, los medios de almacenamiento empleado, los componentes de interfaz gráfica y los tipos de datos de atributos de tablas de bases de datos utilizados para extraer tipos de datos asociados a variables.

Aplicación de la encuesta

Al aplicar los filtros y gráficas de cada una de las encuestas, se observa que la mayoría de los encuestados trabajan en Universidades, donde el 71% son hombres, 6% mujeres, los demás no reportan. En cuanto a la ocupación generalmente es empleado, estudiante o profesor y el mayor sector para el cual se desarrollan aplicaciones, es el sector servicios con un 30%; seguido de los sectores comercio, comunicaciones y agropecuario, con 15% cada uno; y los sectores minero energético y financiero con 12% respectivamente.

En cuanto al tipo de desarrollo que realizan los programadores, 41% son aplicaciones nativas, 21% bridge, 10% híbridas y 15% aplicaciones web y web incrustadas; donde la tecnología empleada es generalmente realizada con un 47% en Android Studio, 22% React Native, 16% Angular y el restante en tecnologías como IONIC, Flutter y X-code. La herramienta de almacenamiento persistente mayormente empleada es SQLite con 54%, archivos planos 28%, seguido de Shared Preferences 16% y otros. En cuanto a la disposición de los modelos empleados para la gestión de formularios (ver figuras 1 y 2) los programadores ven más acorde el flujo dado por el modelo 1 el cual obtuvo una preferencia del 61%.

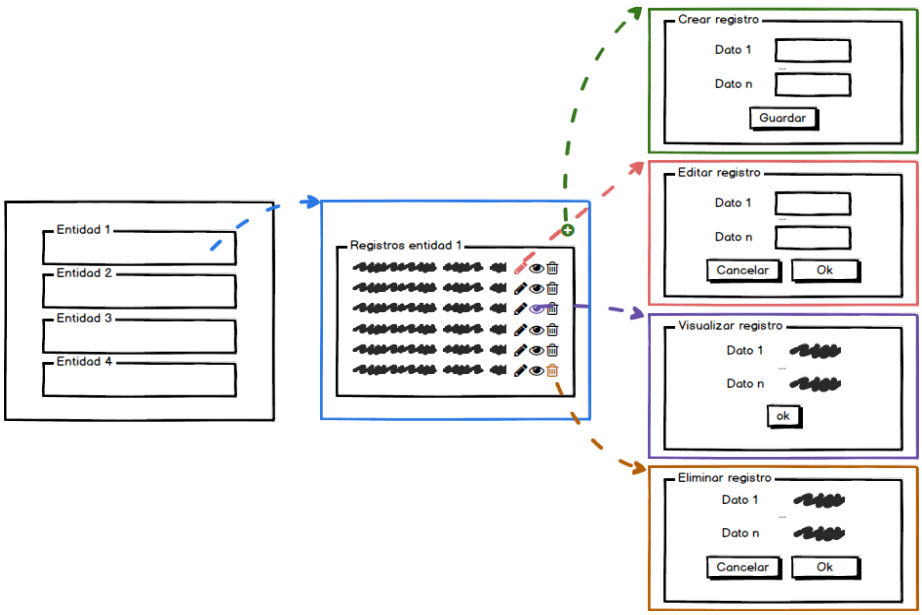

Fig. 1 Modelo 1 para la distribución de GUI.
Fuente: Esta investigación

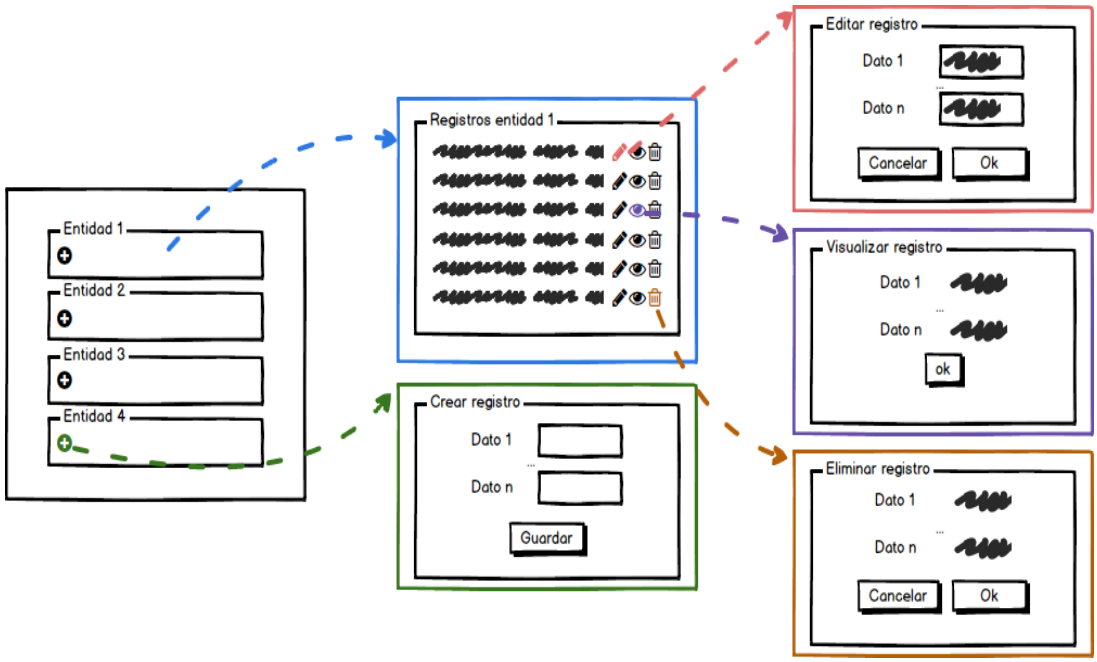

Fig. 2 Modelo 2 para la distribución de GUI.
Fuente: Esta investigación

En las figuras 1 y 2 se muestra el flujo de cada interfaz acorde a cada operación CRUD a realizarse siguiendo una secuencia de color. Los componentes que los desarrolladores más usan según la encuesta son los campos de edición de texto con un 32%, seguido de los botones y cajas de chequeo con un 13% cada uno, las vistas de texto o etiquetas 11%, los botones de radio 10%, los spinners y vista de lista con 8% respectivamente, y el objeto recyclerview con un 4%.

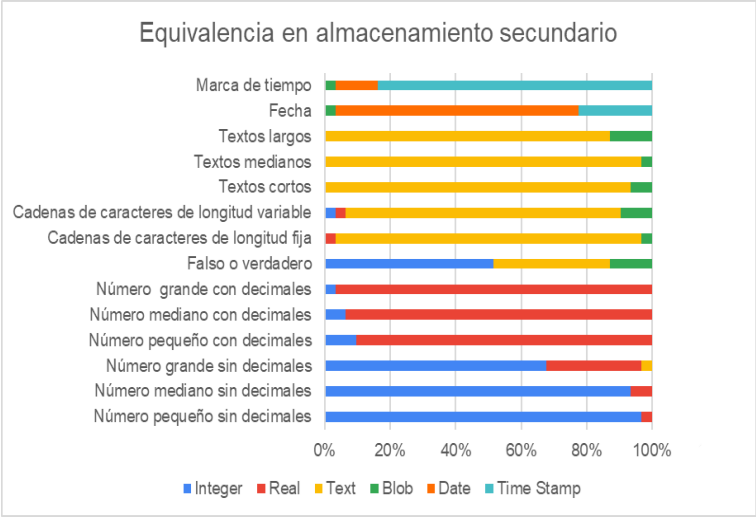

Fig. 3 Representación de atributos en base de datos.

Fuente: Esta investigación

En la figura 3 se observa que un atributo de una entidad es mayoritariamente representado por un tipo de datos en la base de datos en concreto, sin embargo existe un grado de flexibilidad para representar un tipo de datos. Un patrón similar es observado al cruzar los atributos de una entidad con los tipos de datos en el lenguaje de programación como se observa en la Figura 4.

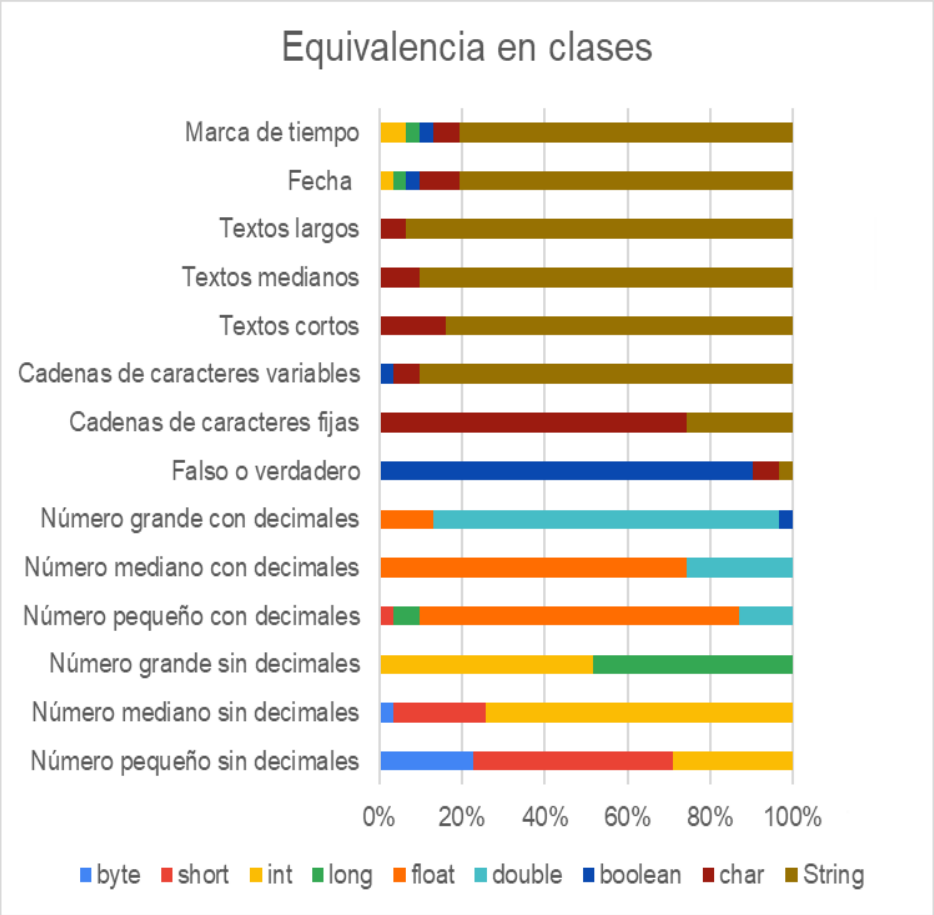

Fig. 4 Representación de atributos en lenguaje de programación.
Fuente: Esta investigación

Como se puede observar en la figura 4, aunque hay un grado de flexibilidad en la representación del atributo de entidad en el lenguaje de programación, en los tipos de datos de longitud mayor y compuestos hay una representación mucho más dominante en el uso de cadenas de texto (*String)*.

Diseño del Framework

Producto del análisis y consolidación de los resultados obtenidos en la aplicación de la encuesta, se extrajeron los componentes más representativos y disposición de gestión formularios *CRUD*. Seguido a ello, se realizó una relación entre el componente, tipos de datos de los atributos de clases tipo entidad y tipos de datos de columnas de tabla en SQLite, para finalmente estructurar una tabla de equivalencias. En la figura 5 puede observarse el proceso anteriormente descrito.

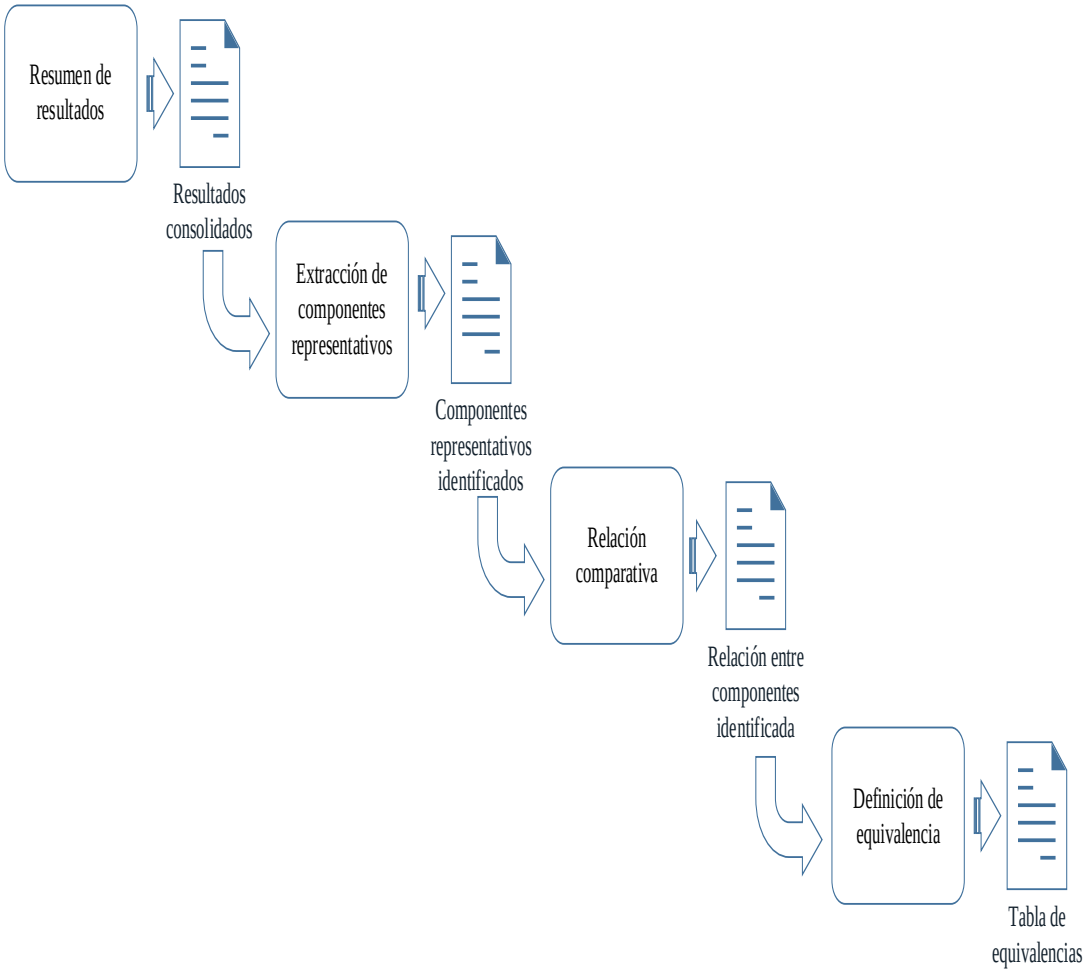

Fig. 5 Representación de atributos en lenguaje de programación.

Fuente: Esta investigación

En la figura 5, se observa como producto final a la tabla de equivalencias, la cual representa el factor diferencial frente a otros frameworks, ya que permite flexibilizar la representación de un componente por tipos de datos en las capas de lógica de negocios y acceso a datos. En la tabla ii se detalla el Tipo de datos (Tipo), su representación en una clase tipo entidad (T.Entity), su representación en SQLite (T SQLite) y componentes de interfaz de usuario Android (CIU) ordenados en orden de representatividad descendente.

Tipo	T. Entity	T. SQLite	CIU
NPSD	Integer	INTEGER	TextView PlainText [alfanumérico] Spinner Seekbar
NMSD	Integer	INTEGER	
NGSD	Integer	INTEGER	
NPCD	Real	REAL	
NMCD	Real	REAL	
NGCD	Real	REAL	

FV	Integer	INTEGER (0 = false, 1 = true)	CheckBox RadioButton
CCLF	Text	TEXT	TextView PlainText [alfanumérico]
CCLV	Text	TEXT	
TC	Text	TEXT	
TM	Text	TEXT	
TL	Text	TEXT	
F	Date	NUMERIC	
TS	TimeStamp	NUMERIC	

Tab. II Relación de capas y preguntas de la encuesta.
Fuente: Esta investigación

En columna Tipo de la tabla II, N significa Número, P pequeño, M Mediano, G Grande, SD Sin Decimales, CD con decimales, FV Falso o verdadero, CC cadena de caracteres, LF Longitud Fija, LV Longitud Variable, T Textos, F Fechas, TS Marca de Tiempo (Time Stamp).

La tabla de equivalencias (tabla II) se introduce como interfaz de configuración de la representatividad de un componente de GUI, con ella, cada atributo de tipo entidad, genera la representación en base de datos y clase entidad con la cual se producen los formularios CRUD para la creación de componentes a través de un Motor de Generación de Interfaces (MGI).

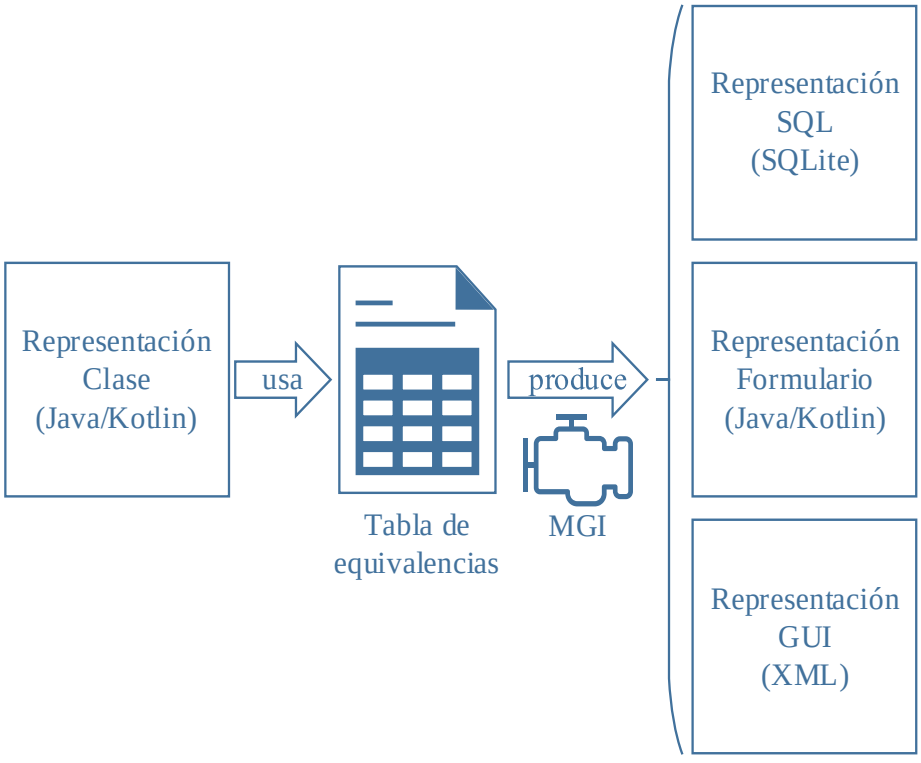

Fig. 6 Interacción componentes del framework.

Fuente: Esta investigación

En la figura 6 se aprecia que la representación de formulario es construida a través de un Motor de Generación de Interfaces (MGI) el cual genera plantillas para formularios CRUD basado en las clases. El diseño contempla soportar una configuración dinámica, que dé soporte a las posibles variaciones de la tabla de equivalencias. La figura 7 muestra el despliegue propuesto para dicha herramienta.

Fig. 7 Interacción componentes del framework.
Fuente: Esta investigación

La figura 7 (a) muestra el estado inicial de un proyecto de aplicación en Android Studio, con dos contenedores principales, el contenedor de recursos (*res*) y el contenedor de los elementos de la lógica (j*ava/kotlin*), al realizar una interacción de creación de una nueva clase, se mostrará una opción más, la que corresponderá a la ejecución del plugin, el cual permitirá la creación de una clase junto con sus formularios de tipo CRUD, los artefactos producidos para el modelo M1 se muestran en la figura 7 (b), donde se evidencia dentro del contenedor (j*ava/kotlin*) la creación del contenedor *Models*, *Commons* y *ActivitiesM1*, este último almacenará las clases controladoras de las actividades del modelo M1, Actividades que a su vez, tendrán la distribución definida mediante el instrumento. En la figura 7 (c), se muestra una nueva iteración de la ejecución del plugin, donde se busca evidenciar, que algunos de los contenedores como *Models* y *Commons*, almacenan los artefactos del nuevo modelo Mn y a su vez, para el nuevo modelo, se crea un contenedor de actividades *ActivitiesMn*.

Discusión y análisis de los resultados

En la ejecución de la fase de definición estructural del generador de plantillas para formularios CRUD basado en clases en Android Studio se han logrado consolidar diferentes resultados relevantes.

En primera instancia, se ha producido un instrumento de recolección de información que permite contextualizar la población a la cual se aplica el instrumento de recolección, también, establecer las tendencias en cuanto a herramientas y tecnologías de desarrollo de los participantes en esta fase. Adicionalmente, este instrumento permitió definir los requerimientos y características principales a las que debe dar cobertura el framework. Estudios como (Cárdenas Villavicencio et al., s/f) establecen comparación entre el desarrollo

de tecnologías móviles pero no hace alusión a la representación de datos en las diferentes capas arquitectónicas.

Por otro lado, el framework propuesto establece una tabla de equivalencia intermedia que permite adquirir diferentes representaciones de atributos a nivel de capa de acceso a datos, lógica de negocios y presentación, a través de cambios a un archivo de configuración y no a un conjunto de archivos con diferentes plantillas como lo muestra (Lazetic et al., s/f). Igualmente, el Motor de Generación de Interfaces *(MGI)* se diferencia del motor propuesto por (Edalat et al., 2015) en que la generación de los archivos XML incluye solo un contenedor u organizador (Layout) al que desde Java o Kotlin se le adicionan los componentes de interfaz. Esto puede ser beneficioso para los desarrolladores *backend*, no obstante para desarrolladores *frontend* la curva de aprendizaje es más sencilla cuando se generan interfaces XML como el enfoque propuesto por (Edalat et al., 2015).

Conclusiones

Los desarrolladores de aplicaciones móviles utilizan diferentes componentes de interfaz de usuario, representación de tipos de datos de atributos de clases y tipos de bases de datos, que indica la importancia de instaurar un mecanismo con una configuración por defecto acorde con la tendencia, pero que también facilite la personalización de cada representación.

El motor de generación de interfaces gráficas favorece el desarrollo automático de formularios CRUD desde un enfoque *código primero* donde se generan tablas de base de datos, lógica de negocio e interfaces utilizando clases tipo entidad.

La implementación de la tabla de equivalencia en la herramienta, permitirá tener un plugin de configuración dinámica, la cual podrá ajustarse a diferentes configuraciones.

Trabajos futuros

La presente investigación ve fundamental validar el framework propuesto con un grupo de desarrollo que permita conocer la curva de aprendizaje, la usabilidad y la mejora en los tiempos de desarrollo.

Por otro lado, se plantea mejorar la arquitectura propuesta permitiendo la generación de código partiendo de la capa de acceso a datos o alguna de los formularios *CRUD* de la capa de presentación.

Igualmente, se requiere establecer mejoras arquitectónicas que permitan configurar la disposición del flujo de formularios CRUD generados, como también el estilo de cada componente.

Otro trabajo futuro que se propone es articular las nuevas perspectivas de la era digital acopladas al desarrollo de frameworks en contextos educativos, investigativos e industriales, dado que estas herramientas tienen mayor presencia con la evolución tecnológica.

Referencias bibliograficas

Andersen, M. S., Cameron, D., Lindemann, J., Wen, F., & Qiao, X. (2019). Template-based Web AR service rapid generation platform. IOP Conference Series: Materials Science and Engineering, 490(4), 042020. https://doi.org/10.1088/1757-899X/490/4/042020

Kotlin. (07/11/2023). Basic types. Disponible en https://kotlinlang.org/docs/basic-types.html

Cárdenas Villavicencio, O. E., Zea Ordóñez, M. P., Valarezo Pardo, M. R., & Ramón Ramón, R. A. (s/f). Comparativa de tendencias de desarrollo de software móvil - 3Ciencias. Recuperado el 10 de mayo de 2022, de

https://www.3ciencias.com/articulos/articulo/comparativa-tendencias-desarrollo-software-movil/

Casas Anguita, J., Repullo Labrador Donado Campos, J. J., & Casas Anguita, J. (s/f). La encuesta como técnica de investigación. Elaboración de cuestionarios y tratamiento estadístico de los datos (I). unidaddocentemfyclaspalmas.org.es. Recuperado el 4 de mayo de 2022, de http://www.unidaddocentemfyclaspalmas.org.es/resources/9+Aten+Primaria+2003.+La+Encuesta+I.+Custionario+y+Estadistica.pdf

DANE. (s/f). Categorías económicas. Disponible en https://www.dane.gov.co/index.php/categoria-economicas

Chandrashekar, A., Kumar, P. V., & Chandavarkar, B. R. (2021). Comparative Analysis of Modern Mobile Operating Systems. 2021 12th International Conference on Computing Communication and Networking Technologies, ICCCNT 2021. https://doi.org/10.1109/ICCCNT51525.2021.9580093

Córdoba, L. G.-E. para su diseño y análisis., & 2006, undefined. (s/f). Encuestas. academia.edu. Recuperado el 4 de mayo de 2022, de https://www.academia.edu/download/55006912/Metodologias_Grasso__Livio__Encuestas._Elementos_para_su_diseno_y_analisis__cap_2_y_5.pdf

Cucciniello, M., Petracca, F., Ciani, O., & Tarricone, R. (2021). Development features and study characteristics of mobile health apps in the management of chronic conditions: a systematic review of randomised trials. npj Digital Medicine, 4(1). https://doi.org/10.1038/S41746-021-00517-1

Dadgar, M., & Joshi, K. D. (2018). The Role of Information and Communication Technology in Self-Management of Chronic Diseases: An Empirical Investigation through Value Sensitive Design. Journal of the Association for Information Systems, 19(2). https://aisel.aisnet.org/jais/vol19/iss2/2

Dar, S. A., & Iqra, J. (2016). Synchronization of Data Between SQLite (Local Database) and SQL Server (Remote Database). The IUP Journal of Computer Sciences, 10, 7–2. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3065777

SQLite. (s/f). Datatypes In SQLite. Disponible en https://www.sqlite.org/datatype3.html

Edalat, E., Sadeghiyan, B., & Ghassemi, F. (2015). ConsiDroid: A Concolic-based Tool for Detecting SQL Injection Vulnerability in Android Apps. www.ietdl.org

Espinoza Pereda, J. C. (2020). Análisis de los frameworks javascript nativo y angular en la incidencia del tiempo de respuesta en una web MVC en el sector comercial. Disponible en https://repositorio.upn.edu.pe/handle/11537/24254

Fujita, H., & Herrera Viedma, E. (2018). New Trends in Intelligent Software Methodologies, Tools and Techniques. IOS Press.

Huang, Y., Emery, J., Naughton, F., Cooper, S., McDaid, L., Dickinson, A., Clark, M., Kinahan-Goodwin, D., Thomson, R., Phillips, L., Lewis, S., Orton, S., & Coleman, T. (2022). The development and acceptability testing of an app-based smart survey system to record smoking behaviour, use of nicotine replacement therapy (NRT) and e-cigarettes. BMC Research Notes, 15(1). Disponible en https://doi.org/10.1186/S13104-022-05983-8

Interfaz de usuario y navegación | Desarrolladores de Android | Android Developers. (s/f). Recuperado el 4 de mayo de 2022, de https://developer.android.com/guide/topics/ui

Lazetic, S., Savic, D., Vlajić, S., & Lazarević, S. (s/f). A Generator of MVC-based Web Applications. World of Computer Science and Information Technology Journal, 2, 741.

Disponible en https://www.researchgate.net/publication/235651622_A_Generator_of_MVC-based_Web_Applications

Núñez, M., Bonhaure, D., González, M., & Cernuzzi, L. (2020). A model-driven approach for the development of native mobile applications focusing on the data layer. Journal of Systems and Software, 161, 110489. Disponible en https://doi.org/10.1016/J.JSS.2019.110489

Ozgur, B., Dogru, I. A., Uctu, G., & Alkan, M. (2021). A Suggested Model for Mobile Application Penetration Test Framework. 14th International Conference on Information Security and Cryptology, ISCTURKEY 2021 - Proceedings, 18–21. Disponible en https://doi.org/10.1109/ISCTURKEY53027.2021.9654417

Primitive Data Types (The JavaTM Tutorials > Learning the Java Language > Language Basics). (s/f). Recuperado el 4 de mayo de 2022, de Disponible en https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Quijano Vodniza, A. J. (2009). Guía de investigación cuantitativa (1a ed.). [Institución Universitaria CESMAG].

Valle, J. del, Mare, J. G.- di, Rica, C., & 2007, undefined. (s/f). Programación en capas. di-mare.com. Disponible en http://www.di-mare.com/adolfo/cursos/2007-2/pp-3capas.pdf