Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los de Recursos Naturales No Renovables

Maestría en Ingeniería en Software

Diseño de arquitectura para el desarrollo y despliegue de aplicaciones móviles transaccionales, dirigido a la Cooperativa de Ahorro y Crédito 15 de abril Ltda.

Trabajo de Titulación previo a la obtención del título de Magíster en Ingeniería en Software

AUTOR:
Emilio José Vera Meza

DIRECTOR:
Ing. Wilman Patricio Chamba Zaragocín Mg. Sc.

Loja - Ecuador
2023

Certificación

Loja, 21 de abril de 2023

Ing. Wilman Patricio Chamba Zaragocín, Mg.Sc.

DIRECTOR DEL TRABAJO DE TITULACIÓN

CERTIFICO:

Que he revisado y orientado todo proceso de la elaboración del Trabajo de Titulación denominado: **Diseño de arquitectura para el desarrollo y despliegue de aplicaciones móviles transaccionales, dirigido a la Cooperativa de Ahorro y Crédito 15 de abril Ltda.**, previo a la obtención del título de **Magíster en Ingeniería en Software**, de autoría del estudiante **Emilio José Vera Meza**, con cédula de identidad Nro. **1312598269**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja para el efecto, autorizo la presentación para la respectiva sustentación y defensa.

Ing. Wilman Patricio Chamba Zaragocín, Mg.Sc.

DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Emilio José Vera Meza**, declaro ser autor del Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación del Trabajo de Titulación en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma: 

Cédula de Identidad: 1312598269

Fecha: 03/05/2023

Correo electrónico: emilio.vera@unl.edu.ec

Teléfono: 0992837583

Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica de texto completo, del Trabajo de Titulación

Yo, **Emilio José Vera Meza**, declaro ser autor del Trabajo de Titulación denominado: **Diseño de arquitectura para el desarrollo y despliegue de aplicaciones móviles transaccionales, dirigido a la Cooperativa de Ahorro y Crédito 15 de abril Ltda.** como requisito para optar el título de **Magíster en Ingeniería en Software**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los tres días del mes de mayo de dos mil veintitrés.

Firma: Firmado electrónicamente por: EMILIO JOSE VERA MEZA

Autor: Emilio José Vera Meza

Cédula de identidad: 1312598269

Dirección: Portoviejo, Cdla Forestal, calle San Cayetano

Correo electrónico: emilio.vera@unl.edu.ec

Teléfono: 0992837583

DATOS COMPLEMENTARIOS:

Director del Trabajo de Titulación: Ing. Wilman Patricio Chamba Zaragocín Mg. Sc.

Dedicatoria

Dedico este Trabajo de Titulación a mis padres, que gracias a su buen ejemplo y buenas enseñanzas forjaron de mi la persona que soy hoy por hoy.

Dedico también este Trabajo de Titulación a mi familia, a mis hijos, y de manera especial a mi esposa, por todo el empuje y apoyo incondicional brindado para conseguir esta meta en mi vida.

Emilio José Vera Meza

Agradecimiento

Agradezco a todos quienes de una u otra manera me brindaron su apoyo y conocimientos para conseguir esto logro, a mi familia, mis amigos, mis compañeros de trabajo.

A la Universidad Nacional de Loja por abrirme sus puertas y permitirme crecer a nivel profesional, a el cuerpo de docentes que impartió sus conocimientos durante todo el año de estudio, y muy particularmente al Ing. Wilman Chamba Zaragocín quien con sus conocimientos y paciencia me acompañó como director de mi Trabajo de Titulación.

Emilio José Vera Meza

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de anexos:

1. Título

Diseño de arquitectura para el desarrollo y despliegue de aplicaciones móviles transaccionales, dirigido a la Cooperativa de Ahorro y Crédito 15 de abril Ltda.

2. Resumen

Hoy en día, a nivel global existe un cambio a nivel social y cultural dirigido por el auge y el fácil acceso a los dispositivos móviles. Bajo esta situación las empresas buscan captar clientes y promover el acceso a sus servicios mediante el uso de aplicaciones que funcionen en estos entornos. El segmento bancario no es la excepción, pues siempre intenta innovar e ir a la vanguardia con la tecnología, ofreciendo soluciones prácticas motivadas por el buen servicio y fácil acceso a sus clientes.

A nivel tecnológico, la creación de estas aplicaciones se convierte en un desafío, pues, pese a que existen un sin número de tecnologías, metodologías y herramientas orientadas a este tipo de desarrollo, no existe una ruta marcada claramente que sirva de referencia para la construcción y publicación de estas aplicaciones. Y es precisamente bajo este contexto en donde el presente Trabajo de Titulación (TT) busca dar una solución a este problema. Considerando esto, surgió la siguiente pregunta de investigación: "¿La propuesta de una arquitectura que conjugue metodología de desarrollo y tecnologías, ayudará a las actividades de desarrollo y despliegue de aplicaciones móviles transaccionales, en la Cooperativa de ahorro y Crédito 15 de abril Ltda.?".

Para resolver esta pregunta, el presente TT partió evaluando metodologías de desarrollo actuales, haciendo uso de trabajos relacionados y encuestas, estableciendo como marco de trabajo la metodología Scrum. Posterior a esto se evaluó mediante entrevista la infraestructura actual de la Cooperativa y la manera en la que se despliegan sus aplicaciones para sugerir un modelo de despliegue basada en patrones arquitectónicos como Api Gateway, y Service Discovery, además de sugerir React y Flutter como frameworks de desarrollo. Y, Finalmente, con el objetivo de asegurar el canal de comunicación entre clientes y aplicaciones, se comparó una solución de seguridad basada en hardware contra una solución basada en software, recomendando mediante criterio inductivo la solución Json Web Token basada en software.

Palabras clave: *Microservicios, Patrones de arquitectura de software, Software bancario, Metodologías de desarrollo, Scrum, React, Flutter, JWT*

2.1. Abstract

Nowadays, at a global level, there is a social and cultural change driven by the boom and easy access to mobile devices. In this situation, companies seek to attract customers and promote access to their services through the implementation of applications that work in these environments. The banking segment is no exception, as it is always trying to innovate and be at the forefront of technology, by offering practical solutions motivated by good service and easy access to its customers.

At the technological level, the creation of these applications becomes a challenge, because, although there are countless technologies, methodologies, and tools oriented to this type of development, there is no clearly marked route that serves as a reference for the construction and publication of these applications. And it is precisely in this context that this Final Degree Project (FDP) aims to provide a solution to this problem. Considering this, the research question arose: "Will the proposal of an architecture that combines development methodology and technologies help the development and deployment activities of transactional mobile applications in the Cooperativa de Ahorro y Crédito 15 de Abril Ltda?".

To solve this question, this FDP started by evaluating current development methodologies, making use of related works and surveys, and establishing the Scrum methodology as a framework. After this, the actual infrastructure of the Cooperative and the way in which its applications are deployed were evaluated through an interview to suggest a deployment model based on architectural patterns such as Api Gateway, and Service Discovery, in addition to suggesting React and Flutter as development frameworks. Finally, with the objective of securing the communication channel between clients and applications, a hardware-based security solution was compared against a software-based solution, recommending by inductive criteria the software-based Json Web Token solution.

Keywords: *Microservices, Software architecture patterns, Banking software, Development methodologies, Scrum, React, Flutter, JWT*

3. Introducción

En el Ecuador, las instituciones financieras ofrecen dentro de su catálogo un sin número de servicios, considerando que son una herramienta fundamental en las estrategias de crecimiento para la industria bancaria [1], no obstante, el acceso a apps de tipo transaccional se encuentra limitado debido en gran parte a factores económicos. Sin embargo, según datos de la AsoBanca, en el año 2021 [2], el canal móvil fue el canal con mayor crecimiento, registrando 3.6 veces más usuarios que el año 2019.

"El gran impacto que están causando los smartphones en la actualidad se debe en gran medida a la evolución de sus sistemas operativos, los cuales cada vez son más estables y robustos, lo que permite a los desarrolladores de software crear aplicaciones móviles de mayor tamaño y complejidad" [3]. Esto ha abierto nuevas oportunidades para el acceso y prestación de los servicios financieros, debido a que es posible proporcionar a la población un mecanismo conveniente para efectuar transacciones entre diferentes instituciones financieras y pagos de servicios en general mediante el uso de apps transaccionales, no obstante, la capacidad para utilizar cualquier servicio financiero digital depende del acceso a la tecnología necesaria [4]. Es ahí donde la arquitectura planteada para orquestar los servicios y recursos tecnológicos se vuelven relevantes, justamente para satisfacer la necesidad de investigar y desarrollar un nivel de abstracción superior al del diseño, enmarcando todos los procedimientos y fundamentos tecnológicos necesarios para la correcta aplicación de patrones y metodologías [5].

Por una parte, tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados [6]. Bajo esta circunstancia, se concibe que la construcción de aplicaciones móviles que se integren transparentemente al core bancario supone desafíos arquitectónicos (en comparación con los productos de software tradicionales), pues, pese a existir una amplia variedad de patrones y metodologías para construir software no existe una arquitectura establecida que solucione este problema [7].

Por lo expuesto, el presente Trabajo de Titulación (TT), se presentó con el fin de dar respuesta a la pregunta de investigación: "¿La propuesta de una arquitectura que conjugue metodología

de desarrollo y tecnologías, ayudará a las actividades de desarrollo y despliegue de aplicaciones móviles transaccionales, en la Cooperativa de ahorro y Crédito 15 de abril Ltda.?". Planteando el objetivo principal "Diseñar una propuesta arquitectónica para el desarrollo y despliegue de aplicaciones móviles transaccionales, mediante la evaluación de frameworks y tecnologías orientadas a servicios, que servirá de base para desarrollos posteriores en la Cooperativa de ahorro y Crédito 15 de Abril Ltda.", y para lograrlo se desarrollaron tres objetivos específicos: "Definir la metodología necesaria para el desarrollo de aplicaciones móviles orientadas a sistemas operativos Android e iOS, mediante el análisis y comparación de metodologías agiles y tradicionales.", "Establecer una arquitectura para exponer y desplegar un api Gateway utilizable en aplicaciones móviles, analizando plataformas de despliegue y comparando frameworks orientados a estos desarrollos." y "Establecer un mecanismo de seguridad para asegurar la comunicación los dispositivos clientes y las apis expuestas, evaluando mecanismos de encriptación y de seguridad realizados por hardware y software.".

En lo que respecta al desarrollo de los objetivos, se partió de la revisión de obras grises, tomando como referencia estudios con propósitos similares para la selección de metodologías en función a criterios ya determinados. Posteriormente, se utilizaron encuestas y entrevistas para cumplir con el segundo objetivo. Por último, apoyados en la inducción y en el análisis de requisitos se pudo completar el objetivo final.

La estructura del presente informe, se compone de la siguiente manera: el **Marco Teórico** expone antecedentes y trabajos relacionados referentes al tema, incluye también los conceptos necesarios para la comprensión del tema principal; la **Metodología** presenta el área de estudio, los procedimientos y recursos utilizados para el desarrollo del presente TT; el apartado de **Resultados** presenta detalladamente las evidencias obtenidas durante el desarrollo de cada objetivo planteado; dentro de la **Discusión** se analizan los resultados obtenidos desde la perspectiva del autor; y para finalizar, se presentan la **Conclusiones y Recomendaciones** donde se abarcan los hechos más relevantes y sugerencias sobre el TT.

4. Marco teórico

El éxito de un producto de software es el resultado de buenas decisiones durante cada una de las etapas de su desarrollo, iniciando desde el levantamiento de requerimientos, hasta la certificación y puesta en producción. Para asegurar rendimiento y escalabilidad son necesarias las actividades de arquitectura de software.

4.1. Arquitectura de software

La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones [8].

"Una arquitectura es el conjunto de decisiones importantes sobre la organización de un sistema de software, la selección de los elementos estructurales y sus interfaces mediante las cuales se compone el sistema" [9].

La arquitectura de software, ha emergido como una disciplina de gran importancia dentro de la ingeniería de software. Una arquitectura adecuada es pieza clave para lograr tanto los requerimientos funcionales como no funcionales de un sistema. Por otro lado, una arquitectura no adecuada puede ser catastrófica [10].

De la misma manera, se vuelve necesario encontrar soluciones "genéricas" a los problemas comunes que se presentan en el desarrollo de software, y aquí intervienen los patrones de diseño.

4.2. Patrones de Diseño Arquitectónico

Un patrón de diseño arquitectónico define una familia de sistemas en base a un esquema de organización estructural. Bajo esta perspectiva, los patrones arquitectónicos pueden ser vistos como los bloques constructivos básicos de las arquitecturas de software. Esto es, definen y organizan un vocabulario de elementos basado en un conjunto de componentes y conectores asociados a un grupo de restricciones que indican la forma en la cual tales elementos pueden ser combinados. De acuerdo con esta perspectiva, los patrones de diseño arquitectónico describen pares de elementos {problema; solución} con el objetivo de brindar soluciones abstractas a problemas recurrentes en un dominio específico [7].

La arquitectura de una aplicación nos dirá que hacer, mientras que los patrones de diseño nos ayudan a saber cómo hacerlo. Consecuentemente con esto, es necesario organizar las actividades y tareas del proceso de desarrollo de software, siendo prácticamente obligatorio ajustarse a una guía de desarrollo para organizar estas tareas, agilizar los procesos y aportando a las mejores del producto final, hoy por hoy conocemos a estas directrices como metodologías.

4.3. Metodologías de desarrollo de software

"Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo" [11].

En sus inicios, las metodologías de desarrollo se concebían como procesos secuenciales, iterativos o de mejora continua (espirales), pero, hoy en día la tendencia está siendo marcada por metodologías que se centran en el trabajo en equipo, la adaptabilidad y la colaboración dentro del equipo de desarrollo, y que tienen como objetivo acelerar la producción de una entrega de software (conocidas como metodologías ágiles) [12]. La TABLA I [13] presenta una comparativa entre las metodologías tradicionales y las metodologías ágiles.

TABLA I
Comparación entre metodologías tradicionales y metodologías ágiles

Metodologías ágiles	**Metodologías tradicionales**
Se basan en heurísticas provenientes de prácticas de producción de código	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo
Preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente por el equipo	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso muy controlado, numerosas normas
Contrato flexible e incluso inexistente	Contrato prefijado
El cliente es parte del desarrollo	Cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10)	Grupos grandes
Pocos artefactos	Más artefactos
Menor énfasis en la arquitectura del software	La arquitectura del software es esencial

A continuación, según su categorización se presentan las metodologías mayormente populares.

4.3.1. Metodologías tradicionales

Las metodologías de desarrollo tradicionales o clásicas son también llamados modelos de proceso prescriptivo, y fueron planteadas originalmente para poner orden en el caos del

desarrollo de software que existía cuando se empezó a generar masivamente. La historia revela que estos modelos tradicionales que fueron presentados en la década de los 60, dieron cierta estructura útil al trabajo de la Ingeniería de software y constituyen un mapa razonablemente eficaz para los equipos de desarrollo [14].

En las metodologías tradicionales se concibe al proyecto como uno solo de grandes dimensiones y estructura definida; el proceso es de manera secuencial, en una sola dirección y sin marcha atrás; el proceso es rígido y no cambia; los requerimientos son acordados de una vez y para todo el proyecto, demandando grandes plazos de planeación previa y poca comunicación con el cliente una vez ha terminado ésta.

Al pasar el tiempo se empieza a detectar los principales problemas tales como la dificultad de responder a los requerimientos cambiantes del cliente.

Se presentan a continuación las metodologías tradicionales mayormente conocidas:

4.3.1.1. Cascada (waterfall)

Este modelo de desarrollo de software toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y, luego, los representa como fases separadas del proceso, tal como especificación de requerimientos, diseño de software, implementación, pruebas, etcétera. Representados en la Fig. 1 [15].

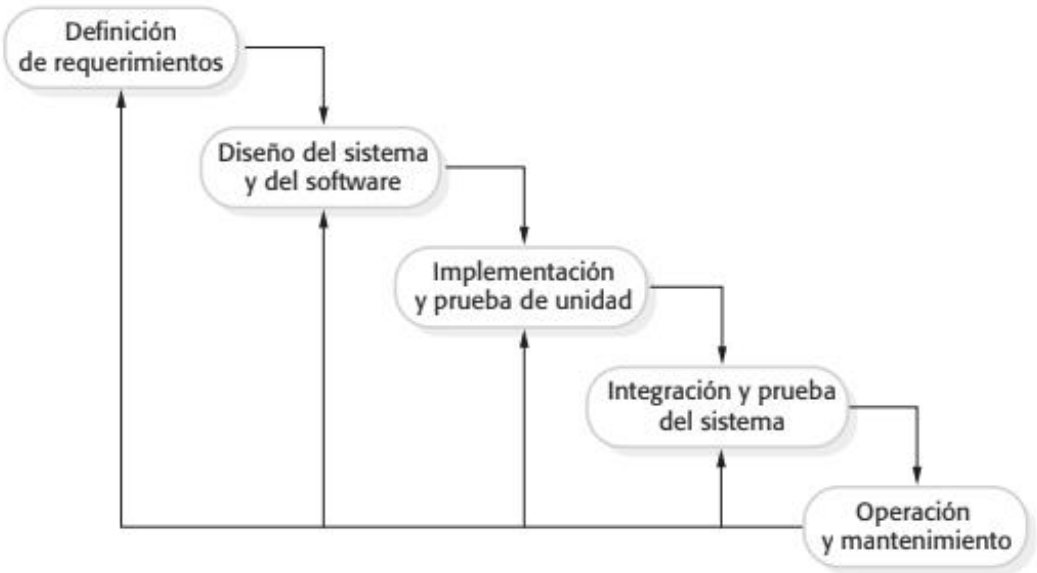


Fig. 1. Modelo de desarrollo en cascada

El modelo en cascada es un ejemplo de un proceso dirigido por un plan; en principio, usted debe planear y programar todas las actividades del proceso, antes de comenzar a trabajar con ellas. En principio, el modelo en cascada sólo debe usarse cuando los requerimientos se

entiendan bien y sea improbable el cambio radical durante el desarrollo del sistema. Sin embargo, el modelo en cascada refleja el tipo de proceso utilizado en otros proyectos de ingeniería. Como es más sencillo emplear un modelo de gestión común durante todo el proyecto, aún son de uso común los procesos de software basados en el modelo en cascada.

4.3.1.2. Incremental

Hay muchas situaciones en las que los requerimientos iniciales del software están razonablemente bien definidos, pero el alcance general del esfuerzo de desarrollo imposibilita un proceso lineal. Además, tal vez haya una necesidad imperiosa de dar rápidamente cierta funcionalidad limitada de software a los usuarios y aumentarla en las entregas posteriores de software. En tales casos, se elige un modelo de proceso diseñado para producir el software en incrementos [16]. En la Fig. 2 se presenta en resumen las actividades que forman parte del modelo de desarrollo incremental [15]

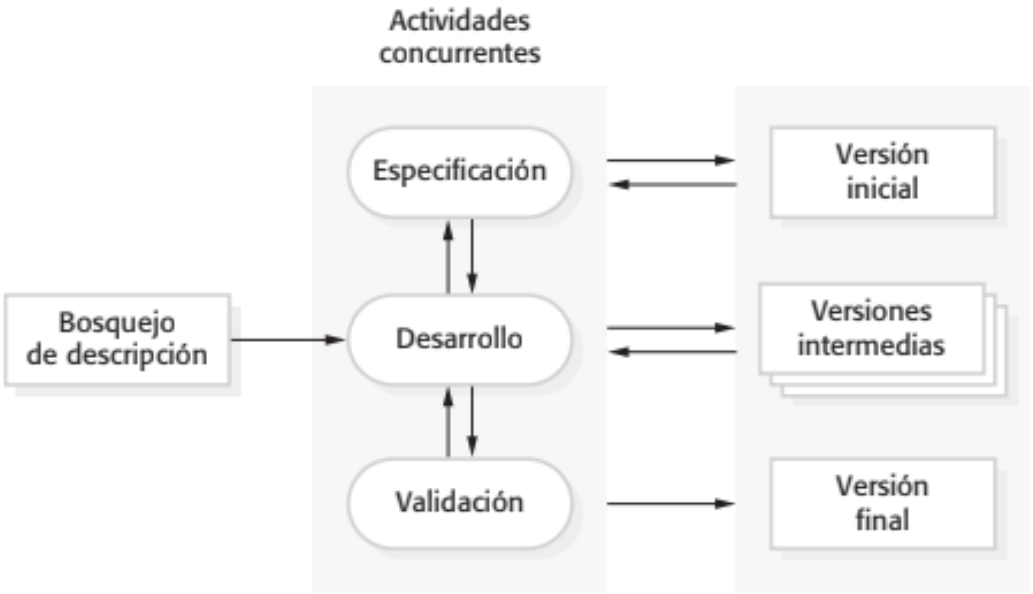


Fig. 2. Modelo de desarrollo incremental

Cada incremento o versión del sistema incorpora algunas de las funciones que necesita el cliente. Por lo general, los primeros incrementos del sistema incluyen la función más importante o la más urgente. Esto significa que el cliente puede evaluar el desarrollo del sistema en una etapa relativamente temprana, para constatar si se entrega lo que se requiere. En caso contrario, sólo el incremento actual debe cambiarse y, posiblemente, definir una nueva función para incrementos posteriores. Comparado con el modelo en cascada, el desarrollo incremental tiene tres beneficios importantes:

1. Reduce el costo de adaptar los requerimientos cambiantes del cliente.

2. Es más sencillo obtener retroalimentación del cliente sobre el trabajo de desarrollo que se realizó.
3. Hace posible que sea más rápida la entrega e implementación de software útil al cliente, aun si no se ha incluido toda la funcionalidad.

4.3.1.3. Espiral

El modelo espiral es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas [16].

El modelo de desarrollo espiral es un generador de modelo de proceso impulsado por el riesgo, que se usa para guiar la ingeniería concurrente con participantes múltiples de sistemas intensivos en software. Tiene dos características distintivas principales. La primera es el enfoque cíclico para el crecimiento incremental del grado de definición de un sistema y su implementación, mientras que disminuye su grado de riesgo. La otra es un conjunto de puntos de referencia de anclaje puntual para asegurar el compromiso del participante con soluciones factibles y mutuamente satisfactorias [17]. La Fig. 3 presenta el ciclo de actividades del modelo de desarrollo en espiral [16].

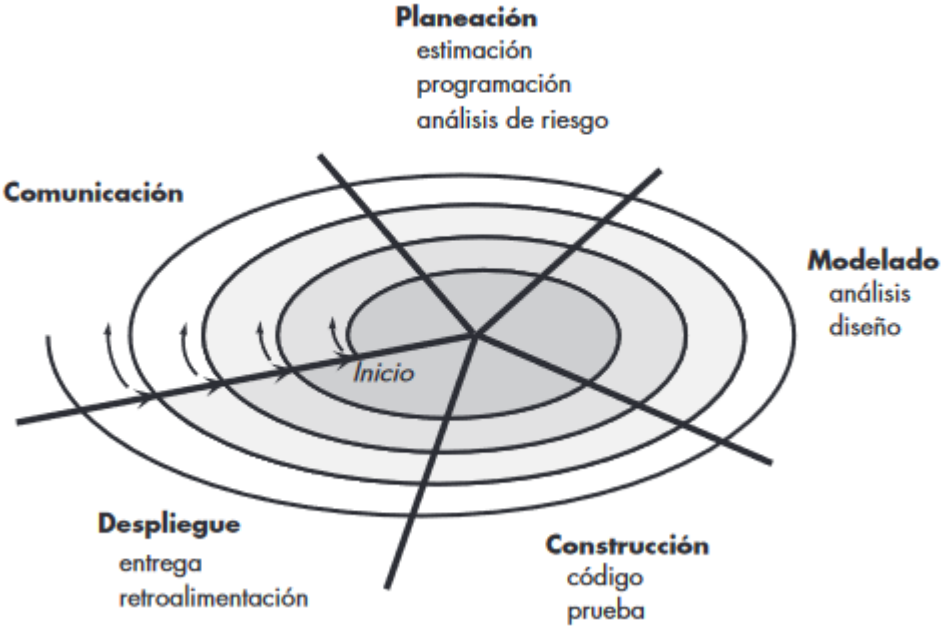


Fig. 3. Modelo en espiral

Con el empleo del modelo espiral, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las

iteraciones posteriores se producen versiones cada vez más completas del sistema cuya ingeniería se está haciendo.

4.3.2. Metodologías ágiles

Hoy en día, el mundo empresarial opera en un entorno global que cambia rápidamente; por ende, se debe responder a las nuevas necesidades y oportunidades del mercado, teniendo en cuenta que el software es partícipe de casi todas las operaciones empresariales, se debe desarrollar soluciones informáticas de manera ágil para poder dar una respuesta de calidad a todo lo necesario [18].

Las metodologías ágiles presentan como principal particularidad la flexibilidad, los proyectos en desarrollo son subdivididos en proyectos más pequeños, incluye una comunicación constante con el usuario, son altamente colaborativos y es mucho más adaptable a los cambios. De hecho, el cambio de requerimientos por parte del cliente es una característica especial, así como también las entregas, revisión y retroalimentación constante [19].

A partir de los años 80, fueron apareciendo modelos de desarrollo basados en iteración y el incremento, algunos de estos modelos son espiral, RAD y RUP. El principio de estos consistió en incrementar sus tareas paso por paso, pero cada tarea tiene un tiempo determinado, por lo tanto, hay alguna interactividad entre ellas.

En febrero de 2001, durante una reunión de expertos de la industria de software, nace el término ágil aplicado al desarrollo de software. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales. A partir de ahí se creó The Agile Alliance, una organización sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y promover dichos conceptos por otras organizaciones. El punto de partida fue el Manifiesto Ágil, un documento dónde se resume toda esta filosofía [20].

4.3.2.1. El manifiesto ágil

Los 12 principios de este manifiesto son [21]:

- **i.** La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
- **ii.** Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- **iii.** Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

iv. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

v. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

vi. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

vii. El software que funciona es la medida principal de progreso.

viii. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

ix. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

x. La simplicidad es esencial.

xi. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

xii. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

4.3.2.2. Scrum

Scrum es un método para trabajar en equipo a partir de iteraciones o Sprints. El objetivo de esta metodología es planificar y controlar proyectos donde hay gran incertidumbre por los cambios que suceden a última hora. La planificación suele hacerse por semanas. Al final de cada sprint o iteración, se va revisando el trabajo validado de la anterior semana. En función de los resultados obtenidos, se priorizan y planifican las actividades en las que invertiremos nuestros recursos en el siguiente Sprint.

La metodología Scrum se centra en ajustar sus resultados y responder a las exigencias reales y exactas de cliente. Se va revisando cada entregable porque los requisitos van cambiando a corto plazo. Por eso, los sprint duran un mínimo de una semana y un máximo de cuatro semanas.

Entre las principales características de la metodología Scrum destaca el desarrollo incremental en lugar de la clásica planificación del desarrollo completo de un producto o servicio. Sus equipos de trabajo se caracterizan por ser auto-organizados. Y se centra en el producto final, en la calidad del mismo. Además, en la metodología Scrum se solapan diferentes fases de desarrollo, en lugar de llevar a cabo una planificación secuencial o de cascada [20].

4.3.2.2.1. Roles de Scrum

La metodología Scrum tiene unos roles y responsabilidades principales, asignados a sus procesos de desarrollo. Estos son:

- Product Owner: Se asegura de que el proyecto se esté desarrollando acorde con la estrategia del negocio. Escribe historias de usuario, las prioriza, y las coloca en el Product Backlog. Debe conocer la velocidad del equipo, para realizar estimaciones de cuando estarán implementadas las necesidades en el producto. Es el responsable de cancelar el sprint si ocurre un imprevisto extremo.
- Scrum Master o Facilitador: Debe participar en las reuniones y asegurarse de que cumplan el tiempo y el objetivo establecido. Elimina los obstáculos que impiden que el equipo cumpla con su objetivo. Se encarga también de que todo el equipo siga la metodología Scrum y que sea entendido por todos.
- Development team: Los encargados de crear el producto para que pueda estar listo con los requerimientos necesarios. Se recomienda que sea un equipo multidisciplinar, de no más de 10 personas. Sin embargo, empresas como Google disponen de unos 15000 desarrolladores trabajando en una rama del código.

4.3.2.2.2. Fases de la metodología Scrum

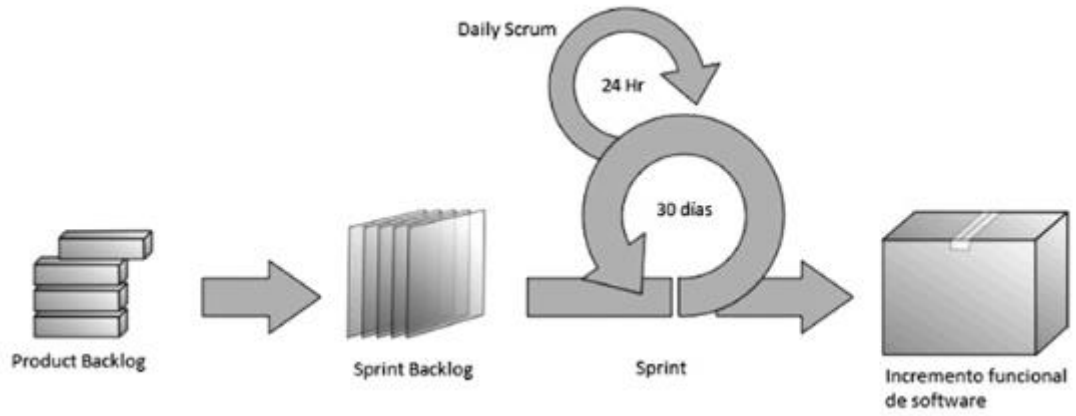


Fig. 4. Actividades y elementos de Scrum

Scrum define un evento principal o Sprint (ver Fig. 4) que corresponde a una ventana de tiempo donde se crea una versión utilizable del producto (incremento). Su duración máxima es de un mes. Un Sprint se compone de: reunión de planeación del Sprint, Daily Scrum, trabajo de desarrollo, revisión del Sprint y retrospectiva del Sprint [19].

En la reunión de Planeación del Sprint se define su plan de trabajo: qué se va a entregar y cómo se logrará. Es decir, el diseño del sistema y la estimación de cantidad de trabajo.

El Daily Scrum es un evento del equipo de desarrollo de quince minutos, que se realiza cada día con el fin de explicar lo que se ha alcanzado desde la última reunión; lo que se hará antes de la siguiente; y los obstáculos que se han presentado.

La Revisión del Sprint ocurre al final del Sprint y su du ración es de cuatro horas para un proyecto de un mes (o una proporción de ese tiempo si la duración es menor). En esta etapa: el dueño del proyecto revisa lo que se hizo, identifica lo que no se hizo y discute acerca del Product Backlog; el equipo de desarrollo muestra el producto y su funcionamiento. Esta reunión es de gran importancia para los siguientes Sprints.

4.3.2.3.Extreme Programming (XP)

Fue desarrollada por Kent Beck buscando guiar equipos de desarrollo de software pequeños o medianos, entre dos y diez desarrolladores, en ambientes de requerimientos imprecisos o cambiantes [22].

XP tiene como base cinco valores: Simplicidad, Comunicación, Retroalimentación, Respeto y Coraje. Estos valores, a su vez, son la base para la definición de sus principios. De ellos, los fundamentales son: la retroalimentación rápida, asumir simplicidad, el cambio incremental, la aceptación del cambio y el trabajo de calidad [22].

Las prácticas de esta metodología se derivan de sus valores y principios y están enfocadas en darle solución a las actividades básicas de un proceso de desarrollo, esto es: escribir código, realizar pruebas, escuchar (planear) y diseñar.

Las prácticas de XP incluyen: planning game, pequeñas entregas, diseño simple, programación en pareja, pruebas, refactoring, integración continua, propiedad común del código, paso sostenible, cliente en sitio, metáfora y están dares de código.

4.3.2.3.1. Roles De XP

Aunque en otras fuentes de información aparecen algunas variaciones y extensiones de roles XP, en este apartado describiremos los roles de acuerdo con la propuesta original de Beck [13]

- **Programador**: El programador escribe las pruebas unitarias produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.
- **Cliente**: El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

El cliente es sólo uno dentro del proyecto, pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

- Encargado de pruebas (**Tester**): El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- Encargado de seguimiento (**Tracker**): El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.
- Entrenador (**Coach**): Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.
- Gestor (**Big boss**): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

4.3.2.3.2. Proceso de XP

Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos [23]:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad

en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto [22].

La Fig. 5 [24], muestra las actividades de la metodología XP.

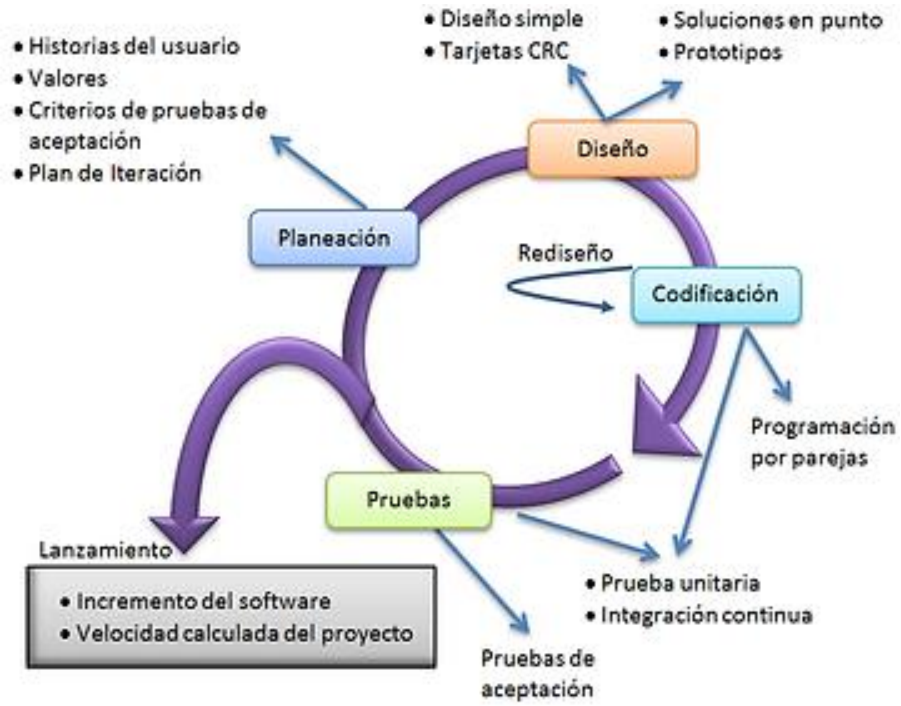


Fig. 5. Marco de trabajo de la metodología XP

4.3.2.4. Rational Unified Processes (RUP)

RUP es proceso de desarrollo de software que, junto con el Lenguaje Unificado de Modelado (UML), constituye una metodología estándar utilizada para el análisis, implementación y documentación de sistemas de software orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización [25].

Se considera que la metodología RUP mejora la productividad del equipo ya que permite que cada miembro del grupo sin importar su responsabilidad específica acceda a la misma base de datos, explícitamente permitiendo realizar pruebas en todos los procesos que se lleva a cabo.

4.3.2.4.1. Fases del RUP

El proceso unificado racional se basa en valores o procesos entre ellos

- Análisis de requerimientos: Los requerimientos son una descripción de los informes de los recursos, la meta principal es identificar y documentar lo que en la realidad se necesita, la forma en que se fácilmente trasmitir, al cliente y al equipo de desarrollo. Se recomienda definir al menos los siguientes puntos: definir los requerimientos, documento de visión y metas.
- Diseño: El diseño es la única manera de materializar los requerimientos del cliente; el diseño debe proporcionar una completa idea de lo que el sistema enfoca, los dominios de datos funcionales y comportamiento desde el punto de vista del desarrollo.

  Las herramientas para el diseño del sistema van de acuerdo al proceso y las características del software para satisfacer los requisitos detectados en la actividad del análisis. En esta fase se define: herramientas de programación, diagramas E-R, software, hardware de base para el desarrollo y operaciones.
- Desarrollo: Con la definición de las herramientas de diseño de software realizado en las fases anteriores se procede a la programación de cada uno de los módulos que compone el software, el código fuente y las aplicaciones a partir de especificaciones funcionales, las pruebas se realizan durante la elaboración del software las mismas pueden ser hechas por las personas que han codificado, la integración de las estrategias garantizan el uso inicial del software que se encuentran libre de los problemas que se descubre durante el proceso que lleva a cabo las correcciones de un buen funcionamiento.
- Implementación y mantenimiento: En este proceso de implementación del software, como resultado de un análisis y diseño previo o mejoramiento de la forma de llevar a cabo un proceso esta debe funcionar de acuerdo a los requerimientos y análisis que los usuarios.

  Se realiza la instalación del producto y se procede al entrenamiento de los usuarios, hasta que el cliente quede satisfecho, por tanto, en esta fase suelen ocurrir cambios. Con estas fases se logra ejecutar un conjunto de mejores prácticas como lo son: Desarrollar software iterativamente, Modelar el software visualmente, Verificar los requisitos y Realizar los cambios correspondientes.

4.3.2.5. RAD

La metodología RAD (Desarrollo rápido de aplicaciones) apareció a finales de los 80 y promueve principalmente la entrega de un sistema de alta calidad en poco tiempo y a un bajo precio. Su creador (James Martin) toma en cuenta cuatro componentes esenciales para el desarrollo RAD, son: Personas, herramientas, metodología y gestión.

La metodología RAD ayuda al desarrollo de aplicaciones de forma rápida y económica para la satisfacción de las empresas con una baja inversión de tiempo y dinero.

4.3.2.5.1. Fases del RAD

Todas las fases de la metodología RAD son cíclicas [26], estas fases se presentan en la Fig. 6 y se exponen a continuación:

Fig. 6. Fases del RAD

- Planificación de necesidades o requerimientos: La primera fase consiste en la definición de las necesidades del proyecto y los requerimientos que solicita la empresa y el alcance del proyecto para tener un punto de partida.
- Diseño y feedback con el usuario: En la fase de diseño se crea un modelo previo del proyecto y se presenta al usuario, recibiendo comentarios que ayudaran a diseñar el modelo definitivo del proyecto, siendo un paso repetitivo las veces que se considere necesarias.
- Construcción: Una vez definido el diseño, se continua con el desarrollo. En esta fase se crea la codificación, las pruebas y la implementación del proyecto. Se pueden realizar las modificaciones necesarias tantas veces como se requiera.
- Transición: La fase final o conocida también como “cutover” consiste en levantar el sistema a un entorno de producción real, donde se realizarán todas las pruebas requeridas.

4.3.2.6. ICONIX

ICONIX, se define como una metodología de desarrollo de software práctico. ICONIX está entre la complejidad del RUP (Rational Unified Processes) y la simplicidad y pragmatismo del XP (Extreme Programming), sin eliminar las tareas de análisis y de diseño que XP no contempla [27].

ICONIX es un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto. Fue elaborado por Doug Rosenberg y Kendall Scott a partir de una síntesis del proceso unificado de los "tres amigos" Booch, Rumbaugh y Jacobson y que ha dado soporte y conocimiento a la metodología ICONIX desde 1993. Presenta claramente las actividades de cada fase y exhibe una secuencia de pasos que deben ser seguidos. Además, ICONIX está adaptado a los patrones y ofrece el soporte de UML, dirigido por casos de uso y es un proceso iterativo e incremental.

Las tres características fundamentales de ICONIX son:

- Iterativo e incremental: varias iteraciones ocurren entre el desarrollo del modelo del dominio y la identificación de los casos de uso. El modelo estático es incrementalmente refinado por los modelos dinámicos.
- Trazabilidad: cada paso está referenciado por algún requisito. Se define trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos producidos.
- Dinámica del UML: La metodología ofrece un uso "dinámico del UML" como los diagramas del caso de uso, diagramas de secuencia y de colaboración.

4.3.2.6.1. Las tareas de ICONIX

Los creadores de esta metodología destacan un análisis de requisitos, un análisis y diseño preliminar, un diseño y una implementación como las principales tareas.

Análisis de Requisitos: El trabajo es iniciado con un relevamiento informal de todos los requisitos que en principio deberían ser parte del sistema. Luego con los requisitos se construye el diagrama de clases, que representa las agrupaciones funcionales con que se estructura el sistema que se desarrolla. En la medida de lo posible se puede presentar una prototipación rápida de las interfaces del sistema, los diagramas de navegación, etc., de forma que los clientes puedan comprender mejor el sistema propuesto.

Análisis y Diseño Preliminar: Durante esta tarea se describen los casos de uso, como un flujo principal de acciones, pudiendo contener los flujos alternativos y los flujos de excepción. La principal sugerencia de ICONIX, en esta actividad, es que no se debe perder mucho tiempo con la descripción textual. Debería usarse un estilo consistente que sea adecuado al contexto del proyecto.

Diseño: Aquí se especifica el comportamiento a través del diagrama de secuencia, por cada caso de uso se debe identificar los mensajes entre los diferentes objetos. El diagrama de secuencia muestra interacciones entre objetos según un punto de vista temporal. El contexto de los objetos no se representa de manera explícita como en los diagramas de colaboración. La representación se concentra sobre la expresión de las interacciones.

El diagrama de secuencia es el núcleo de nuestro modelo dinámico y muestra todos los cursos alternos que pueden tomar todos nuestros casos de uso.

4.3.2.7. Mobile-D

Esta metodología (ver Fig. 7) se concentra especialmente en las pequeñas empresas de desarrollo, debido a los tiempos cortos de desarrollo lo que produce como resultado la minimización de costes de producción, lo cual hace esta metodología se convierta en asequible para pequeñas organizaciones que se limitan a tener poco personal y recursos [28].

4.3.2.7.1. Fases de Mobile-D

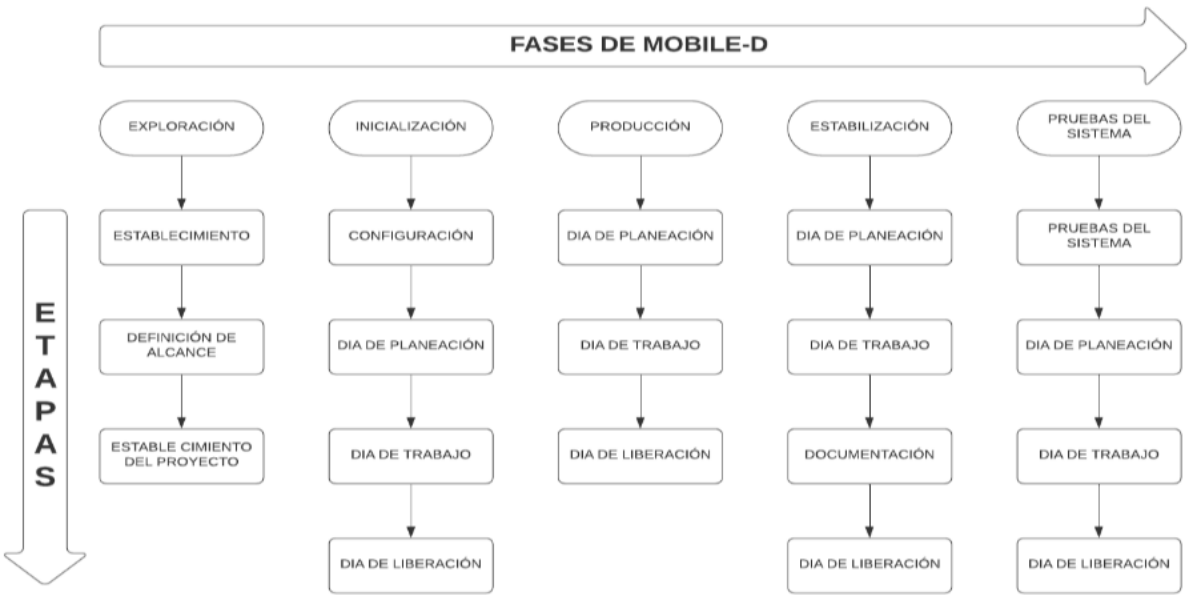


Fig. 7. Fases de Mobile-D

- Exploración: Los objetivos de la primera etapa son establecer los grupos de partes interesadas, que son uno de los principales indicadores de lo que se espera de la aplicación móvil, definir los objetivos de la aplicación móvil y elegir qué miembros van a participar en el desarrollo de la aplicación móvil [29]. Las salidas obtenidas en esta fase son las siguientes:
    - Los requisitos iniciales
    - Plan del proyecto.
    - Descripción de los procesos
    - Plan de medida
    - Plan de capacitación
- Inicialización: En esta se prepara el diseño arquitectónico, diagramas de casos de uso, diseño de interfaz de usuario (UI) y sus diferentes funcionalidades [30]. La documentación conseguida es la siguiente:
    - El plan actual del proyecto.
    - La versión de la arquitectura del software y la descripción del diseño.
    - Requisitos iniciales modificados.
    - Interfaces de Usuario.
    - Diagramas de Casos de Uso.
- Producción: La fase de producción incluye la implementación real. Se divide en Día de planificación, Día de lanzamiento y Días laborables.

  Los días de planificación: tienen como objetivo analizar, mejorar y priorizar los requisitos, planificar los contenidos de la iteración actual y preparar los casos de prueba de aceptación que se usará el día del lanzamiento.

  Los días laborables: implementan funcionalidades en el desarrollo guiado por pruebas de software (TDD).

  Los días de lanzamiento: se lanza una versión funcional para pruebas de aceptación de cliente utilizando los casos de prueba desarrollados durante los días de 11 planificación.

  Además del modelo informativo y de vista de usuario, durante la producción de la fase, el equipo de desarrollo utiliza el caso de uso, el componente, el diagrama de clase de la actividad y secuencia, así como el patrón MVC para llevar la implementación y alinear la comprensión de los miembros del equipo [31].

  Una vez culminada esta fase se recolecta la siguiente documentación:
    - Funcionalidades puestas en funcionamiento.

- Anotaciones del desarrollo.
- Esquemas de la interfaz de usuario de la aplicación.
- Storycards.
- Requisitos modificados.

- Estabilización: Se llevan a cabo las últimas acciones de integración para asegurar que el sistema completo funciona correctamente. Esta será la fase más importante en los proyectos multi-equipo distinta. En esta fase, los desarrolladores realizaran tareas similares a las que debían desplegar en la fase de "producción", aunque en este caso todo el esfuerzo se dirige a la integración del sistema [32]. Una vez finalizada esta fase se alcanzar los siguientes requisitos:
  - La funcionalidad puesta en funcionamiento en todo software del proyecto.
  - La documentación del producto terminado.
- Pruebas del Sistema: Su propósito es que la aplicación sea estable y funcional para que los clientes la utilicen. La aplicación una vez terminada se la integra y la prueba en base a los requisitos del cliente y suprimen todos los errores descubiertos [33].

4.4. Tipos de aplicaciones móviles

La TABLA II presenta la comparativa entre los tipos de aplicaciones móviles (en función a su diseño) [34].

TABLA II
TIPOS DE APLICACIONES MÓVILES

Aplicaciones web sobre móviles	**Aplicaciones web móviles nativas**	**Aplicaciones nativas**
Las aplicaciones web sobre móviles son aplicaciones que no necesitan ser instaladas en el dispositivo para poder ejecutase. Están basadas en tecnologías HTML, CSS y Javascript, y que se ejecutan en un navegador. A diferencia de las webs móviles, cuyo objetivo básico es mostrar información, estas aplicaciones tienen como objetivo interaccionar con el dispositivo y con el usuario. De esta manera, se le saca un mayor partido a la contextualización.	Existe un tipo de aplicaciones, llamadas aplicaciones web móviles nativas, que no son aplicaciones web propiamente ni tampoco nativas. Se ejecutan con un navegador o, mejor dicho, con un componente nativo que delega en un navegador, y tienen algunas de las ventajas de las aplicaciones nativas.	Las aplicaciones nativas son las aplicaciones propias de cada plataforma. Deben ser desarrolladas pensando en la plataforma concreta. No existe ningún tipo de estandarización, ni en las capacidades ni en los entornos de desarrollo, por lo que los desarrollos que pretenden soportar plataformas diferentes suelen necesitar un esfuerzo extra.

4.5. Frameworks para desarrollo móvil

4.5.1. Flutter

Flutter es un framework de desarrollo de aplicaciones móviles multiplataforma creado por Google. Es de código abierto y permite construir aplicaciones tanto para Android como para iOS. Su versión 1.0 fue lanzada al mundo el 4 de diciembre de 2018, por lo que es una tecnología muy nueva. A pesar de su corta edad, se trata de una tecnología muy madura debido a que es utilizada en Google para crear sus herramientas internas.

Está construido por capas, estando el motor escrito en C/C++ y las librerías en Dart. También usa Skia para el renderizado 2D. El objetivo de esta herramienta es permitir a los desarrolladores construir aplicaciones multiplataforma a partir de una única base de código, la cual es compilada a código nativo para cada una de las plataformas objetivo. Además, se aprovecha de la flexibilidad de Dart en cuanto a su compilación y ejecución para obtener ciclos de desarrollo más rápidos y tiempos de ejecución más bajos.

A diferencia de otras soluciones, en Flutter se construye toda la aplicación usando Dart, incluida la interfaz de usuario. Para ello, se apoya fuertemente en el paradigma de la programación orientada a objetos, más concretamente en la composición y herencia de los llamados widgets, los cuales componen la interfaz. Estos widgets son fundamentales para cualquier aplicación móvil. La peculiaridad de Flutter respecto a otras soluciones se encuentra en que no utiliza los widgets de la plataforma, sino que provee los suyos propios [35].

Flutter construye los widgets a nivel de aplicación, lo que hace que el desarrollador pueda personalizarlos y extenderlos de manera sencilla, consiguiendo una mayor libertad en el diseño de las aplicaciones. Esto se consigue gracias a que Flutter solo requiere de la plataforma un canvas o lienzo donde “pintar” los widgets.

Otro de los puntos fuertes de Flutter es la gran calidad de sus herramientas de desarrollo. Una de estas herramientas, y quizás la más popular, es la recarga rápida. Al usar otras tecnologías, el desarrollador tiene que reiniciar la aplicación cada vez que haga un cambio en el código y quiera ver su efecto. La recarga rápida de Flutter hace uso de la compilación JIT de Dart para reflejar los cambios en el código durante la ejecución de manera casi instantánea manteniendo el estado. Esto contribuye a reducir los tiempos de los ciclos de desarrollo [36].

4.5.2. Ionix

Ionic es un framework de código abierto basado en la tecnología de Apache Cordova que permite crear aplicaciones móviles híbridas. A diferencia de las aplicaciones nativas, las aplicaciones híbridas son implementadas usando tecnologías web (HTML, CSS y JavaScript) y ejecutadas en el dispositivo de destino aprovechando las capacidades de su motor de navegador [37].

El HTML usado para definir la interfaz es renderizado durante la ejecución, al igual que ocurre con el código JavaScript, el cual se interpreta sobre la marcha en el dispositivo. Como ocurría con React Native, el principal inconveniente de este acercamiento es la pérdida de rendimiento. Sin embargo, si comparamos estas dos herramientas, React Native sale favorecida por el hecho de apoyarse en un único lenguaje, JavaScript, en vez del conglomerado de tecnologías que usa Ionic [36].

4.5.3. React

Creado por Facebook, React Native es un framework utilizado principalmente para el desarrollo de aplicaciones móviles, aunque también permite trabajar para la plataforma universal de Windows. La idea principal es permitir el uso de React, una famosa librería para la construcción de interfaces de usuario mantenida por Facebook, en las plataformas nativas [38].

Así, una aplicación construida con esta herramienta usa componentes React para la interfaz y código JavaScript para la lógica. Simplificando, permite utilizar las mismas librerías usadas para el desarrollo de aplicaciones web, pero para plataformas móviles. Esto hace que esta herramienta resulte especialmente atractiva para desarrolladores u organizaciones que ya tengan aplicaciones web implementadas con estas tecnologías. Sin embargo, la librería React tiene una curva de aprendizaje pronunciada, lo que dificulta su adopción por parte de usuarios nuevos. Las aplicaciones desarrolladas con este framework no se compilan a las plataformas de destino, sino que su código es interpretado durante la ejecución en el dispositivo. Esto hace que sea más difícil detectar errores en el código durante el desarrollo, además de suponer un peor rendimiento frente a alternativas que sí que compilan el código [36].

4.5.4. Xamarin

En sus comienzos, Xamarin era una empresa que desarrolló implementaciones de la plataforma .NET de Microsoft para plataformas móviles. Microsoft acabó comprando esta empresa y usando su nombre para identificar al conjunto de herramientas que permiten utilizar una base

de código C# para el desarrollo de aplicaciones en diversas plataformas, permitiendo la compartición de código entre estas [39].

Con Xamarin, se puede usar C# no sólo para la implementación de la lógica de negocio, sino que también provee de controles de interfaz personalizados para cada plataforma. Sin embargo, esto hace que la implementación de estas interfaces con C# no se pueda compartir entre plataformas, haciendo que el porcentaje de código compartido sea mucho menor que en otras soluciones. Por último, otra de las desventajas de Xamarin es el gran tamaño de sus aplicaciones. Las aplicaciones desarrolladas con esta herramienta se compilan para cada plataforma, pero siguen teniendo un tamaño que, en ocasiones, es notablemente mayor que el de aplicaciones nativas [36].

4.6. HSM (Hardware Security Module)

Un módulo de seguridad por hardware es un dispositivo de hardware resistente a la manipulación de externos, que se utiliza principalmente en la industria financiera para proporcionar altos niveles de protección a las claves criptográficas y a la información de las tarjetas de los clientes. Generalmente, estos módulos se emplean para proporcionar llaves para funciones críticas como encriptación, desencriptación y autenticación [40].

Las siguientes son características de los HSM que contribuyen a su seguridad:

- Diseño seguro: utilizan hardware especialmente diseñado que adhiere a los estándares FIPS (Estandarización Federal de Procesamiento de Información, Federal Information Processing Standardization), a los Criterios Comunes, y a los requisitos impuestos por la industria de pagos con tarjeta PCI DSS (Payment Card Industry Data Security Standard).
- Resistencia a la manipulación: se someten a procesos que los vuelven resistentes a accesos malintencionados y manipulación ilícita.
- Sistema operativo seguro: su sistema operativo que fue construido basado en la seguridad.
- Aislamiento: se ubican físicamente en el área segura del centro de datos para evitar el acceso no autorizado.
- Control de acceso: controlan el acceso a sus datos, están diseñados para evidenciar los signos de manipulación y algunos tienen la funcionalidad de volverse inoperables o eliminar sus claves criptográficas si la detectan.

4.7. JWT (Json Web Token)

La autenticación basada en token es una referencia en el desarrollo de aplicaciones web, ya que presenta algunas ventajas respecto a la autenticación más común, en la que se guardan en sesión los datos del usuario [40].

En la autenticación con token, el usuario se identifica bien con un usuario/contraseña o mediante una única clave y la aplicación web le devuelve una especie de firma cifrada que el usuario usará en las cabeceras de cada una de las peticiones HTTP.

Esta información no se tiene que almacenar en la parte del servidor, como en el caso de la autenticación a través de sesión, sino que se guarda en la parte del cliente y es la aplicación la que comprobará si es válida en cada una de las peticiones.

A su vez, con ello se gana en escalabilidad, pudiendo usar cualquier tecnología (Web, Android, iOS...) para hacer uso de la aplicación, sin diferenciar entre ellas en el servidor.

El token que envía como respuesta la aplicación tiene un tiempo de vida el cuál se debe configurar acorde a lo que interese.

El JWT está formado por tres cadenas separadas por punto

- Header: Es la primera parte del token, está formada por el tipo de token y el algoritmo de codificación utilizado
- Payload: Está compuesto por atributos llamados Claims, existen:
    - iss: especifica la tarea para la que se va a usar el token.
    - sub: presenta información del usuario.
    - aud: indica para que se emite el token. Es útil en caso de que la aplicación tenga varios servicios que se quieren distinguir.
    - iat: indica la fecha en la que el token fue creado.
    - exp: indica el tiempo de expiración del token, se calcula a partir del iat.
    - nbf: indica el tiempo en el que el token no será válido hasta que no transcurra.
    - jti: identificador único del token. Es utilizado en aplicaciones con diferentes proveedores.
- Signature: La firma es la última de las tres partes, está formada por la información del Header y el Payload codificada en Base64, más una clave secreta que se configura en la propia aplicación

5. Metodología

En este acápite, se exponen las metodologías abordadas para el cumplimiento del presente TT. Por tal motivo, se expone el área de estudio, el procedimiento realizado para el cumplimiento de cada objetivo, y los recursos empleados.

5.1. Área de estudio

El presente TT se realizó en la Cooperativa de Ahorro y Crédito 15 de abril LTDA. Entidad que ofrece servicios financieros y es regulada por la Superintendencia de economía popular y solidaria. La matriz de la Cooperativa, se encuentra domiciliada en la ciudad de Portoviejo, calle 18 de octubre entre Córdova y 10 de agosto, siendo su representante legal la Ing. María Verónica Mendoza Cevallos. El trabajo fue realizado entre los meses de diciembre/2022 y marzo/2023.

5.2. Procedimiento

Para la obtención de resultados sobre cada objetivo, se establecieron los siguientes procedimientos:

5.2.1. Definir la metodología necesaria para el desarrollo de aplicaciones móviles orientadas a sistemas operativos Android e iOS, mediante el análisis y comparación de metodologías agiles y tradicionales.

- Se partió de la revisión de obras grises, utilizando un estudio elaborado con propósito similar, en donde se escogió una metodología usando los criterios de nivel de presencia, documentación y nivel de conocimiento.
- Para la selección de la metodología final, se usó la encuesta, con el fin de determinar el grado de conocimiento que tiene el equipo de desarrollo sobre las metodologías que se postularon como alternativas (véase Anexo 1).

5.2.2. Establecer una arquitectura para exponer y desplegar un api Gateway utilizable en aplicaciones móviles, analizando plataformas de despliegue y comparando frameworks orientados a estos desarrollos.

- Se realizó entrevista con el administrador de infraestructura de la Cooperativa, para determinar la situación actual y documentar el modelo de despliegue usado en sus aplicaciones. Adicionalmente, la entrevista ayudó a evidenciar necesidades que fueron plasmadas posteriormente como requerimientos (véase Anexo 4).

- La selección del framework de desarrollo se determinó utilizando el mismo criterio empleado para dar cumplimiento al objetivo 1, aplicando una variante en la ponderación de los criterios de selección (véase Anexo 2).

5.2.3. Establecer un mecanismo de seguridad para asegurar la comunicación los dispositivos clientes y las apis expuestas, evaluando mecanismos de encriptación y de seguridad realizados por hardware y software.

- Se usó la entrevista, con el administrador de infraestructura de la Cooperativa donde se evaluaron las alternativas existentes que pudieran brindar un mecanismo de seguridad al canal de comunicación api-cliente.
- Usando inducción, y analizando los requerimientos levantados en la fase anterior, se determinó la mejor alternativa que brindará una solución al problema de seguridad, considerando los factores funcionales y económicos.

5.3. Recursos

Para dar cumplimiento a los objetivos propuestos, se utilizaron los siguientes recursos:

5.3.1. Recursos científicos

5.3.1.1.Investigación bibliográfica

Mediante el uso de esta técnica se logró recabar información fundamentada científicamente que aportó a la estructuración del presente trabajo de titulación. Se tomaron como fuentes de información: artículos científicos, revistas científicas, tesis y libros.

5.3.2. Recursos técnicos

5.3.2.1. Encuesta

Este recurso fue dirigido al equipo de desarrollo de la Cooperativa, con la finalidad de determinar el grado de conocimiento en metodologías y frameworks de desarrollo. Los resultados tabulados de estas encuestas fueron determinantes en la toma de decisiones que llevaron al cumplimiento de los objetivos planteados.

5.3.2.2.Entrevista

Este instrumento investigativo ayudó a evidenciar e identificar las necesidades y requerimientos que debieron de satisfacerse para el cumplimiento de los objetivos propuestos. Las entrevistas realizadas se presentan en el apartado de Anexos.

5.3.3. Recursos de hardware y software

5.3.3.1. Hardware

- Laptop Dell Vostro 3400: Utilizado para la realización del presente TT.

5.3.3.2. Software

- Microsft Office: Utilizado para documentar el TT.
- Diagrams.net: Se usó para el desarrollo de los diagramas incluidos en el presente trabajo.
- Google Forms: Usado para la elaboración y tabulación de encuestas.
- Mendeley Reference Manager: Utilizado para la gestión de bibliografía.

5.3.4. Recursos humanos

Para el desarrollo del presente TT, se contó con los siguientes participantes:

- Ing. Emilio Vera Meza, autor del trabajo de investigación
- Ing. Wilman Patricio Chamba Zaragocín, director y revisor del proyecto.
- Ing. Carlos Moreira Ubillus, analista de infraestructura de la Cooperativa de Ahorro y Crédito 15 de abril.

6. Resultados

Con el afán de exponer la evidencia de los resultados obtenidos para el cumplimiento de los objetivos específicos establecidos para el presente TT, se pudo obtener los siguientes resultados:

6.1. Definir la metodología necesaria para el desarrollo de aplicaciones móviles orientadas a sistemas operativos Android e iOS, mediante el análisis y comparación de metodologías agiles y tradicionales.

Para el desarrollo de este objetivo, se tomó como base un estudio realizado en la Universidad Nacional Mayor de San Marcos (Perú) [11], en donde se usaron criterios basados en el nivel de presencia en el medio y nivel de conocimiento del equipo de trabajo, para la selección de una metodología de desarrollo.

Criterio de acuerdo al nivel de presencia (popularidad)

Los diseñadores de software se decantan por trabajar con metodologías que se encuentren suficientemente documentadas, y que faciliten el desarrollo de actividades y planificación. Sin embargo, es importante poner a consideración las metodologías que ofrecen training y certificaciones.

Según estas condiciones se determinaron 6 (seis) clasificaciones para la selección de la metodología:

- La metodología más popular en Internet.
- La metodología mejor documentada.
- Metodologías que ofertan training y certificaciones.
- Metodologías con respaldo de una comunidad.
- Metodología con mayor presencia empresarial.
- Metodología más utilizada en proyectos de software.

Se considera como metodologías certificadas, aquellas que emiten un certificado que aseguran el cumplimiento y seguimiento de la metodología. Una metodología dispone de training, si se encuentra alguna institución, organización o compañía que ofrezca formación de las metodologías. Se considera que una metodología tiene comunidad, contemplando si se ha formado una comunidad relevante o si está asociada a la Agile Alliance, soportándola y cumpliendo sus principios.

Se consideran los proyectos realizados, en su mayoría por metodologías que se han aplicado en empresas privadas y por lo tanto no existe mucha documentación pública al respecto. Por lo tanto, determinar esta clasificación, requiere de una búsqueda exhaustiva.

Para puntuar esta clasificación los autores del estudio consideraron 1 (uno) como el menor puntaje y 10 (diez) el mayor puntaje, y para aplicar la calificación a cada clasificación se tuvo la siguiente consideración:

Presencia en internet**:** Se realizaron búsquedas en Google, Google Academic, Yahoo y Live.

Nivel de documentación: Se consideraron libros y papers en español e inglés que hablen sobre la aplicación de la metodología.

Certificación y training: Se buscaron empresas que certifiquen la implementación de la metodología, así como también se oferte capacitación y entrenamiento en la misma.

Respaldo de una comunidad: La mayoría de metodologías revisadas, pertenecen a la Agile Alliance, pero hay algunas que tienen sus propias comunidades, alianzas e intensa actividad a su alrededor.

El resultado del estudio [11], bajo este criterio, se muestra en la TABLA III.

TABLA III
RESULTADO DE SELECCIÓN DE METODOLOGÍA BAJO CRITERIO DE POPULARIDAD

Metodología	Mayor presencia en Internet	Mejor documentación	Certificadas y con training	Comunidades	Presencia empresarial	Proyectos de software	Total
Agile Project Management (APM)	2	1	3	5	1	1	11
Dynamic Systems development methods (DSDM)	1	3	5	5	4	4	22
Scrum	5	2	5	5	5	5	27
Test Driven Development	3	4	3	2	2	2	16
Extreme Programming (XP)	4	5	3	2	3	3	19
Total	15	15	19	19	15	15	95

Criterio por nivel de conocimiento

Para la selección bajo esta clasificación, se partió con las metodologías halladas, y se elaboró una encuesta aplicada a su equipo de trabajo, donde se tuvieron las siguientes consideraciones:

- Grado de conocimiento
- Soporte orientado a objetos
- Adaptabilidad

- Basado en casos de uso
- Posee documentación adecuada
- Facilita la integración entre las etapas de desarrollo
- Relación con UML
- Permite desarrollo software sobre cualquier tecnología

Para puntuar esta clasificación, según Rios y Suntaxi [41], y como se indica en la TABLA IV, se establecieron los pesos para cada criterio. Del 100% de la calificación, se asignó el 20% para el Grado de conocimiento, 15% para la adaptabilidad y la documentación adecuada, 10% para el resto de criterios.

TABLA IV
PONDERACIÓN DE CRITERIOS PARA SELECCIÓN DE METODOLOGÍA

Criterios	Regular	Normal	Bueno	Alto
Grado de Conocimiento	5	10	15	20
Soporte Orientada a Objetos	2	5	8	10
Adaptable a Cambios	2	5	10	15
Basado en Casos de uso	2	5	8	10
Posee Documentación Adecuada	2	5	10	15
Facilita la Integración entre las Etapas de Desarrollo	2	5	8	10
Relación con UML	2	5	8	10
Permite Desarrollar Software Sobre Cualquier Tecnología	2	5	8	10

En nuestro caso, para la definición y selección de la metodología, se usó el resultado de selección por criterio de popularidad, y se aplicó el criterio de conocimiento a nuestro equipo de trabajando, utilizando encuestas con formularios de Google. Aplicando el criterio de pesos a la tabulación de la encuesta, se obtuvo el siguiente resultado (ver TABLA V), el cual se puede ver en detalle en el Anexo 1.

TABLA V
RESULTADO DE LA SELECCIÓN DE METODOLOGÍA

Criterio	**%**	**RUP**	**RAD**	**XP**	**Iconix**	**Scrum**	**Mobile-D**
Grado de conocimiento	20			15		15	10
Soporte Orientado a objetos	10			8		5	
Adaptable a cambios	15				10	10	5
Basado en casos de uso	10	5				10	5
Posee documentación adecuada	15			10		10	5
Facilita la integración entre las etapas de desarrollo	10	8	5		5	5	
Relación con UML	10	8			5	5	5
Permite desarrollo sobre cualquier tecnología	10				5	5	5
Total	100	21	5	33	25	65	35

Como se observa de acuerdo a la TABLA V, los encuestados respondieron que, Scrum es la metodología que prefieren definir en sus proyectos según los criterios de: conocimiento y popularidad; y por lo tanto de acuerdo a esta consideración, se establece que el marco de trabajo más adecuado para el desarrollo y seguimiento de proyectos es Scrum.

6.2. Establecer una arquitectura para exponer y desplegar un api Gateway utilizable en aplicaciones móviles, analizando plataformas de despliegue y comparando frameworks orientados a estos desarrollos.

Para la resolución del objetivo planteado, se partió de la observación directa y la entrevista. En el Anexo 3, se evidencia la entrevista realizada al administrador de infraestructura de la Cooperativa, el mismo que colaboró en el levantamiento de la documentación para el diagrama de despliegue que utilizan actualmente en una de sus aplicaciones web. Como resultado, se pudo determinar que ellos utilizan los siguientes estilos arquitectónicos:

- Cliente/Servidor
- FrontEnd y BackEnd, para separar las vistas y las reglas de negocio

La Fig. 8 presenta el diagrama de despliegue utilizado actualmente para la construcción y despliegue de aplicaciones desarrolladas por la Cooperativa.

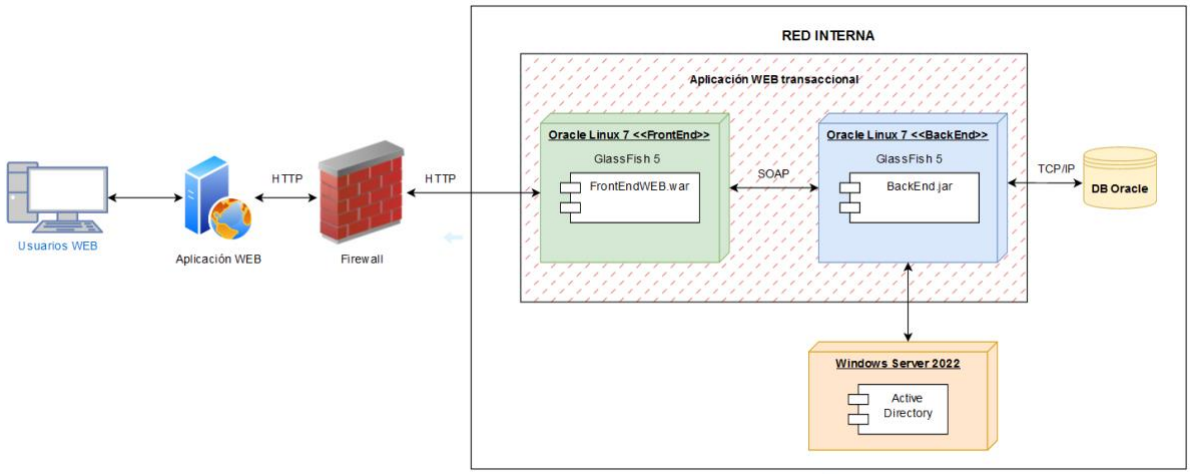


Fig. 8. Despliegue actual de aplicación WEB

El modelo de despliegue de esta aplicación es básico, se expone la aplicación en una red interna, y se le otorga la salida al internet asegurado por un firewall.

Posterior al levantamiento del diagrama de despliegue inicial, se determinaron requerimientos funcionales y no funcionales con el administrador de infraestructura, en la medida de las necesidades de la Cooperativa, enfocados en llevar la aplicación evaluada a un ambiente móvil. La evidencia de esta actividad se presenta en el Anexo 3.

Como resultado del análisis de requisitos, se determinaron los siguientes requerimientos no funcionales (ver TABLA VI):

TABLA VI
REQUISITOS NO FUNCIONALES DEFINIDOS

Código	**Descripción**
RNF01	Implementar un componente que exponga servicios REST, este componente debe desplegar en el servidor FrontEnd, tendrá como finalidad exponer la lógica de negocios contenida en el BackEnd para integrarla con los clientes móviles.
RNF02	Implementar el patrón Api Gateway, para centralizar todas las peticiones recibidas desde el exterior a cada servicio local.
RNF03	Proponer el patrón Service Discovery, que se encargará de recuperar todas las instancias de los servicios disponibles y realizar el balanceo de cargas.
RNF04	Escalar la autenticación a un SSO, y delegar las tareas de verificación de identidad a un solo componente centralizado, esto permitirá implementar modelos de autenticación y seguridad para aplicaciones futuras.
RNF05	Implementar un mecanismo de seguridad práctico y económico para establecer la comunicación entre las apis y las aplicaciones clientes.

La solución a las necesidades encontradas, se presentan en la Fig. 9.

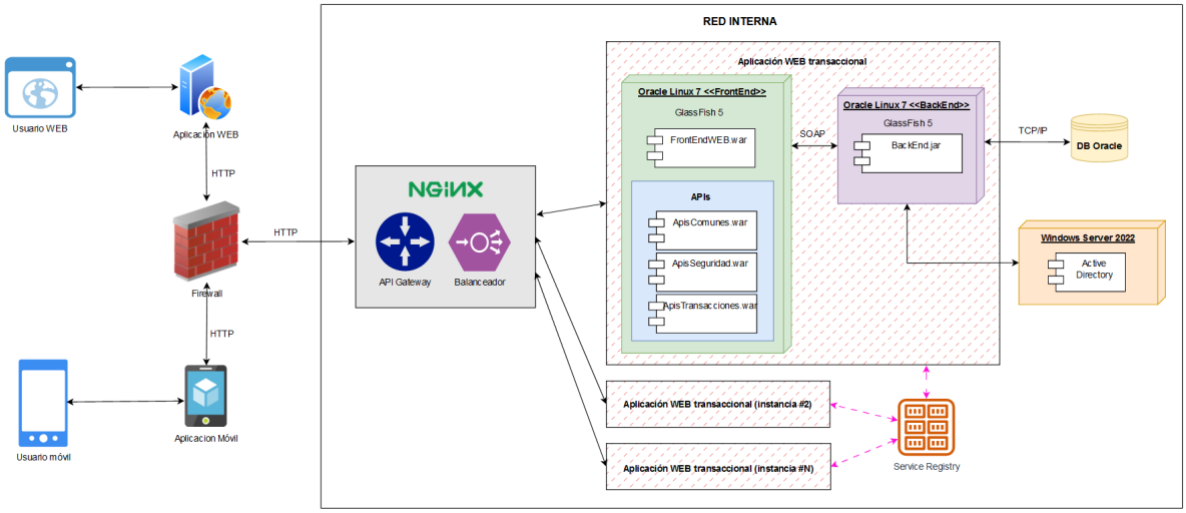


Fig. 9. Propuesta para el despliegue de aplicaciones

En la Fig. 9, se expone el modelo de despliegue propuesto, que asegura alta disponibilidad y escalabilidad, facilitando la introducción e integración de nuevas aplicaciones a futuro.

En lo que respecta a la selección de un framework orientado al desarrollo móvil, se usó el criterio empleado en la metodología según Rios y Suntaxi [41]. Para este efecto, se proponen 4 frameworks populares en el entorno empresarial de desarrollo a la fecha con son: Flutter, Ionic, React y Xamarin; con una variante en los criterios usados (ver TABLA VII).

TABLA VII
PONDERACIÓN DE CRITERIOS PARA SELECCIÓN DE FRAMEWORK

Criterio	**Regular**	**Normal**	**Bueno**	**Alto**
Grado de conocimiento	5	10	15	20
Posee documentación	2	5	10	15
Desarrollo multiplataforma	2	5	10	15
Estandarización de datos y estructuras	2	5	8	10
Alto rendimiento y desarrollo rápido	2	5	8	10
Permite desarrollar apps hibridas	2	5	8	10
Contiene elementos personalizables	2	5	8	10
Facilita pruebas y depuración	2	5	8	10

La evaluación de los criterios, fue socializada con el equipo de trabajo, utilizando encuestas con formularios de Google. Aplicando el criterio de pesos a la tabulación de la encuesta, se obtuvo el siguiente resultado (ver TABLA VIII), el cual se puede ver en detalle en el Anexo 2.

TABLA VIII
RESULTADO DE LA SELECCIÓN DE FRAMEWORK

Criterio	**%**	**Flutter**	**Ionic**	**React**	**Xamarin**
Grado de conocimiento	20	10		20	10
Posee documentación	15	10		15	10
Desarrollo multiplataforma	15			5	15
Estandarización de datos y estructuras	10	15	5		
Alto rendimiento y desarrollo rápido	10			10	
Permite desarrollar apps hibridas	10		8	8	10
Contiene elementos personalizables	10	10	8		
Facilita pruebas y depuración	10		8		10
Total	100	45	29	58	55

Tal como se presenta en la TABLA VIII, el equipo de trabajo se orienta por el uso de React como framework para el desarrollo de apps, y bajo esta consideración se determina que el framework recomendado para el desarrollo de aplicaciones móviles debe ser React debido al grado de conocimiento que tiene el equipo de trabajo sobre este framework, sin embargo, en base a los trabajos relacionados revisados [36], podría considerarse Flutter como una segunda alternativa, considerando que es un framework popular con una corta curva de aprendizaje, además de tener una gran ventaja sobre los demás frameworks, permite la creación de vistas reactivas sin necesidad de usar un puente JavaScript, y lo más importante, que permite crear aplicaciones multiplataformas tanto para web, móvil y escritorio, sin necesidad de hacer ajustes en el código original.

6.3. Establecer un mecanismo de seguridad para asegurar la comunicación los dispositivos clientes y las apis expuestas, evaluando mecanismos de encriptación y de seguridad realizados por hardware y software.

Para dar cumplimiento a este objetivo, se recurrió nuevamente a la entrevista con el administrador de infraestructura de la Cooperativa, y se expusieron 2 alternativas para establecer la seguridad en el canal de comunicación (1 solución basada en hardware y 1 solución basada en software). Las mismas que se presentan a manera de comparativa en la TABLA IX.

TABLA IX
COMPARATIVA ENTRE SOLUCIÓN HSM Y JWT

	HSM (Hardware Security Module)	JWT (JSON Web Token)
Tipo de solución	Hardware	Software
Tipo de estándar	Propietario	Abierto (RFC 7519)
Costo/Licenciamiento	Costo del equipo $16000 aprox. adicional el tiempo y capacitación para los operarios. (ver Anexo 4)	Solo se necesita invertir tiempo para su implementación / autoeducación.
Tipo de autenticación	Al menos un par de llaves entre usuario y encriptador (cada llave requiere de al menos 2 componentes)	Usuario/contraseña
Análisis de alta disponibilidad	Se requiere invertir en equipos de iguales características	Se puede implementar nuevas instancias de JWT en servidores ya existentes.

Bajo criterio inductivo, se recomienda la selección de la solución JWT basada en software, partiendo del factor económico. Tanto su implementación inicial, como las implementaciones para asegurar alta disponibilidad requieren de la creación de nuevas instancias de la aplicación, mientras que, para la solución HSM es necesario realizar una inversión económica elevada para la Cooperativa. Adicionalmente, si se considera el volumen de concurrencia de las aplicaciones cliente, se tiene que puntualizar que, para los HSM al tratarse de hardware de uso específico para encriptación, es necesario que cada aplicación cliente conozca la llave que cifra el canal, mientras que, para los JWT cada aplicación autentica de manera separada en base a un usuario y clave. Resultando más práctica y escalable la dicha solución, en comparación con la solución de HSM que implica realizar una inversión económica, cumpliendo con uno de los RNF antes indicados (RNF05).

Con el objetivo de formular una capa adicional de seguridad, se propone también:

- Implementar el uso de certificados SSL para las aplicaciones expuestas al internet.
- Implementar una solución de firewall para aplicaciones (WAF).

Como resultado, se propone la arquitectura presentada en la Fig. 10.

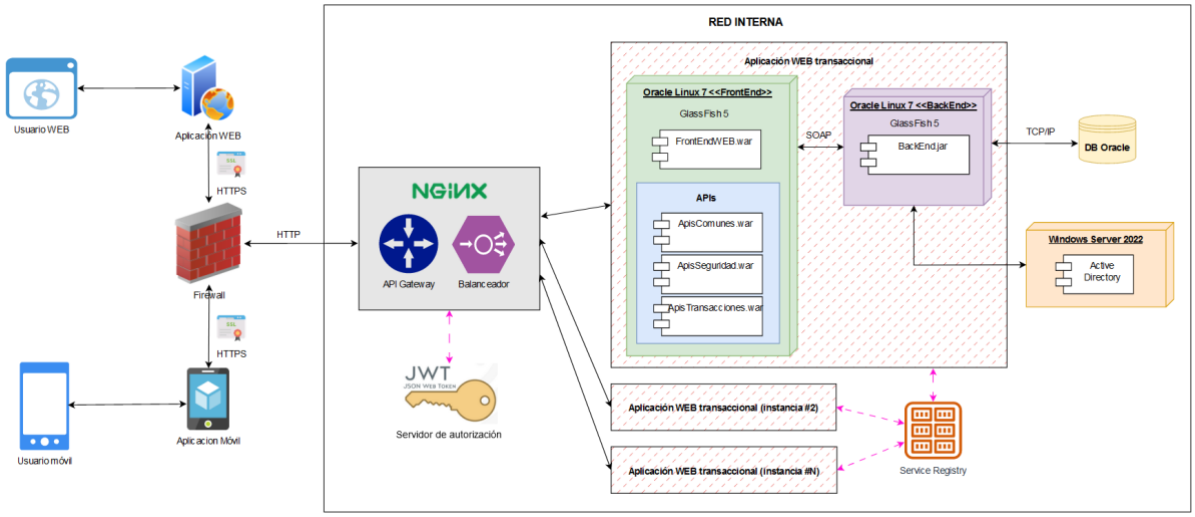


Fig. 10. Propuesta final para el despliegue de aplicaciones

7. Discusión

La propuesta arquitectónica para el despliegue de aplicaciones móviles transacciones de la Cooperativa de ahorro y crédito 15 de abril LTDA. es el resultado del trabajo en conjunto con el personal de T.I. de dicha entidad, quienes colaboraron con el levantamiento de diagramas de especificación de requerimientos, que fueron fundamentales para el desarrollo del presente trabajo de titulación.

La arquitectura resultante de este proyecto pudo determinarse en la medida del cumplimiento de los objetivos propuestos previamente. Dichos objetivos se discuten a continuación:

7.1. Objetivo 1: Definir la metodología necesaria para el desarrollo de aplicaciones móviles orientadas a sistemas operativos Android e iOS, mediante el análisis y comparación de metodologías agiles y tradicionales.

A pesar de que hoy por hoy existan muchas metodologías de desarrollo de software y gestión de proyectos orientadas a entregas incrementales y parciales, según los trabajos referenciados se estableció en primera instancia que Scrum es la metodología elegida por preferencia en proyectos de este tipo.

Scrum es un marco de trabajo ágil, que se puede enfocar en el desarrollo de software, aumentando la eficiencia y la calidad del producto final. En el caso del desarrollo de aplicaciones móviles, Scrum puede ser especialmente beneficioso por varias razones:

- Flexibilidad: Scrum facilita la adaptación a los cambios y a las necesidades del cliente de forma ágil, lo que es especialmente importante en el desarrollo de aplicaciones móviles, donde la competencia y la tecnología cambian presurosamente.
- Entrega continua: Scrum está enfocado en la entrega de incrementos de software funcionales de manera frecuente y regular, lo que permite a los clientes y usuarios finales ver el progreso y hacer comentarios en tiempo real.
- Colaboración: Scrum promueve la colaboración y la comunicación constante entre los miembros del equipo, lo que es fundamental para el éxito del proyecto de desarrollo de aplicaciones móviles.
- Mejora continua: Scrum enfatiza la revisión y la mejora continua del proceso de desarrollo, lo que ayuda a los equipos a identificar y solucionar problemas de manera proactiva y a mejorar la calidad del producto final.

- Enfoque en el usuario: Scrum se enfoca en el usuario final y en sus necesidades, lo que es garantiza que la aplicación sea útil, fácil de usar y satisfaga las necesidades planteadas.

En nuestro caso, haciendo uso de la inducción y las encuestas dirigidas al equipo de desarrollo de software pudimos determinar que nuestro trabajo concuerda con los estudios analizados, por tanto, la metodología como marco de trabajo elegida fue Scrum debido a los antecedentes ya expuestos, y sobre todo es una metodología que aplica la filosofía ágil y entregas continuas.

7.2. Objetivo 2: Establecer una arquitectura para exponer y desplegar un api Gateway utilizable en aplicaciones móviles, analizando plataformas de despliegue y comparando frameworks orientados a estos desarrollos.

Gracias a las entrevistas, y apoyados con el personal de T.I. de la Cooperativa se pudo identificar algunos inconvenientes en la arquitectura aplicada para el despliegue de sus aplicaciones, como se indica el diagrama de arquitectura inicial (ver Fig. 8). A través del levantamiento de requisitos se determinó que se pueden aplicar los patrones Api Gateway y Service Discovery para solucionar los inconvenientes encontrados en la arquitectura inicial, y sobre todo para cubrir los requerimientos solicitados; además, estos patrones ofrecen mejoras consideradas dentro de los requerimientos no funcionales (ver TABLA VI). Entre los que destacan:

- Seguridad: API Gateway actúa como una puerta de enlace para todas las solicitudes entrantes y salientes, lo que le permite aplicar políticas de seguridad sobre todas las APIs.
- Escalabilidad: Api Gateway puede manejar la carga y el tráfico entrante a través de una variedad de técnicas de balanceo de carga y escalado automático, lo que permite a los servicios y las APIs detrás de ella escalar dinámicamente en función de la demanda, mientras que el Service Registry facilita la escalabilidad de los servicios, ya que los clientes pueden acceder a varios servicios idénticos y replicados.
- Flexibilidad: Se permite a los equipos de desarrollo agregar, modificar y quitar servicios y funcionalidades sin afectar directamente a los clientes o usuarios finales. También permite la implementación de múltiples versiones de una API, lo que permite la evolución gradual de la API sin interrupciones para los usuarios.
- Monitoreo y análisis: Es posible disponer de información detallada sobre el tráfico entrante y saliente, lo que permite a los equipos de desarrollo supervisar el rendimiento,

la latencia y la tasa de errores de las APIs. También puede ayudar a identificar cuellos de botella y problemas de rendimiento en los servicios y APIs detrás de ella.

- Mejora de la experiencia del usuario: API Gateway puede ayudar a mejorar la experiencia del usuario final mediante la agregación de servicios y funcionalidades de diferentes fuentes en una sola API coherente y fácil de usar.
- Tolerancia a fallos: El Service Registry puede detectar automáticamente los servicios que fallan y redirigir a los clientes a servicios alternativos que estén disponibles.

La propuesta arquitectónica de un Api Gateway + Service Registry permitirá a la Cooperativa proporcionar servicios y funcionalidades de manera más efectiva y eficiente a través de una API centralizada y bien gestionada, mejorando significativamente la gestión y el rendimiento de una infraestructura orientada a microservicios.

Por otra parte, durante la ejecución de este objetivo se evaluó también el grado de conocimiento del equipo de desarrollo en función a frameworks de trabajo para desarrollo de aplicaciones móviles, utilizando el mismo criterio de selección que el empleado en la selección de la metodología. El resultado permitió determinar que React sería la herramienta apropiada para el desarrollo de este tipo de aplicaciones debido al nivel de conocimiento del equipo.

Sin embargo, según la investigación realizada y la documentación revisada como referencia, se destacó Flutter como un framework que se encuentra en auge y goza de popularidad debido a sus numerosos beneficios y ventajas dentro de las aplicaciones empresariales como comerciales. Es así como Flutter se volvió una excelente opción para el desarrollo de aplicaciones móviles debido a su capacidad de crear aplicaciones multiplataforma de alto rendimiento, su amplia gama de widgets personalizados, su corta curva de aprendizaje y su gran comunidad de desarrolladores. Por tales motivos, se sugirió evaluar este framework como una segunda alternativa para el desarrollo de las siguientes aplicaciones.

7.3. Objetivo 3: Establecer un mecanismo de seguridad para asegurar la comunicación los dispositivos clientes y las apis expuestas, evaluando mecanismos de encriptación y de seguridad realizados por hardware y software.

Basados en la inducción y las necesidades del usuario, para cumplir con este objetivo se partió de 2 posibles opciones (JWT y HSM).

A pesar de que JWT y HSM son dos conceptos diferentes que no son necesariamente comparables entre sí, debido a que abordan diferentes aspectos dentro de la seguridad de la

información, fueron considerados y evaluados para el desarrollo de este objetivo ya que ambas soluciones se pueden orientar hacia la autenticación y autorización de transacciones.

Durante la evaluación, fue evidente que la implementación de la solución basada en hardware resultó desde un inicio poco viable debido al costo de adquisición e implementación que la Cooperativa debería de asumir en caso de escoger esta alternativa, además de encontrar las siguientes limitantes:

- Al ser un hardware de uso específico para encriptación, es necesario que cada aplicación cliente conozca la llave que cifra el canal, derivando en 2 posibles problemas:
    - Si se llegara a compartir una misma llave para todos los clientes existe un riesgo algo para vulnerar el canal, en caso de que la llave sea extraída.
    - Generar llaves personalizadas para cada cliente implicaría crear aplicaciones personalizadas, lo cual no sería viable en aplicaciones de distribución masiva.
- El procesamiento de transacciones por segundo también es un recurso limitado, lo cual implicaría escalar en número de equipos en la medida que crezca la demanda sobre las peticiones. Al tratarse de un componente de hardware implica que, ante el crecimiento de la demanda de peticiones se deba realizar nuevas inversiones en compra de equipos, lo que podría comprometer la ejecución de futuros desarrollos afectando al presupuesto.

Esto, en comparación con la alternativa de software que por tratarse de un estándar abierto no requiere costo de inversión en su etapa de implementación, además de presentar varias ventajas adicionales como:

- Seguridad en la autenticación y autorización de usuario a nivel web y móvil, utilizando algoritmos de criptografía al firmar y verificar los tokens.
- Portabilidad, al poder integrarse en aplicaciones web, móviles y de escritorio.
- Permite a las aplicaciones escalar fácilmente al manejar la autenticación y autorización de los usuarios sin la necesidad de mantener sesiones de usuario en el servidor, lo que puede ser costoso y limitante en términos de escalabilidad.
- Al tratarse de un formato de tokens ligero, no requiere almacenamiento adicional en el servidor. Lo que permite una mayor eficiencia y rendimiento en la comunicación entre la aplicación cliente y el equipo de autenticación.

- Proporciona flexibilidad para personalizar los campos de los tokens y las políticas de expiración, lo que permite a los desarrolladores adaptarse a diferentes requisitos de negocio y necesidades específicas de la aplicación.

Por los motivos expuestos se decidió incluir la solución de JWT dentro de la propuesta arquitectónica, con la finalidad de tener un componente encargado de autenticar y autorizar las peticiones generadas por las aplicaciones clientes y recibidas en el api Gateway.

Adicional a este análisis, y con el afán de ofrecer una capa adicional de seguridad a las aplicaciones, se propuso también la implementación de certificados SSL para la publicación de aplicaciones, y el uso de firewall WAF para mitigar vulnerabilidades de tipo Cross-Ssite scripting, Sql injection, DDoS, entre otros, además del filtro de contenido y generación de informes y alertas.

Debemos recalcar que, la arquitectura basada en microservicios conjuntamente con el marco de trabajo Scrum y el desarrollo basado en el framework React / Flutter, efectivamente ayudan con el desarrollo y despliegue de aplicaciones moviles dentro de la Cooperativa de ahorro y Crédito 15 de abril Ltda., dando respuesta a la pregunta de investigación.

8. Conclusiones

Posterior a la culminación del presente TT, se puede concluir lo siguiente:

- La selección del marco de trabajo SCRUM como metodología de desarrollo permite administrar los proyectos de software, debido a que se enfoca en la colaboración y la entrega continua de producto, logrando así aumentar la eficiencia y el compromiso de los equipos de trabajo, ayudando a gestionar el tiempo y los recursos en un ambiente dirigido y colaborativo.
- La propuesta arquitectónica entregada a la Cooperativa de Ahorro y Crédito 15 de abril Ltda, es una propuesta que se elabora a la medida de sus necesidades, priorizando la optimización de los recursos existentes, y que se proyecta con la finalidad de facilitar el desarrollo, mantenimiento y despliegue de aplicaciones orientadas a entornos móviles, asegurando funcionalidad, alta disponibilidad de los servicios, escalabilidad y seguridad.
- Según las encuestas realizadas para medir el nivel de conocimiento relacionado con frameworks orientados a desarrollo móvil, se determinó que el framework React es la opción deseada por el equipo de desarrollo para la construcción de las aplicaciones móviles; pero, también en base a los trabajos relacionados se determina que Flutter es la opción factible para el desarrollo de aplicaciones relacionas con el BackEnd (móviles, web y escritorio). Ambos frameworks se adaptan perfectamente con la metodología Scrum para fortalecer el desarrollo en entornos ágiles.
- JWT (Json Web Token) es la mejor opción para asegurar las aplicaciones que requieren autenticación y autorización de sus recursos, y perfectamente se adapta a la mayoría de patrones de arquitectura como el Api Gateway; ya que, al generar tokens ligeros, permite que las aplicaciones no necesiten almacenamiento ni procesamiento adicional, otorgando un plus de eficiencia con respecto a soluciones de autenticación y autorización basadas en hardware.
- Formular arquitecturas de software que incluyan un marco metodológico de referencia para el desarrollo de aplicaciones es una tarea que requiere de experiencia y responsabilidad, pues, el éxito de los proyectos elaborados bajo esas referencias depende de correcta selección de tecnologías y toma de decisiones.

9. Recomendaciones

Una vez concluido el TT, se pudo llegar a las siguientes recomendaciones:

- Para la correcta aplicación del marco de trabajo Scrum, es necesario que todo el equipo de trabajo entienda los principios fundamentales de la metodología. Comprendiendo el papel de cada uno de los miembros, el proceso de trabajo y los artefactos que deben de crearse durante cada actividad y entrega.
- Afianzar conocimientos relacionados con el marco de trabajo SCRUM, enfocado principalmente al equipo de desarrollo. Estas actividades pueden incluir herramientas para la automatización de pruebas unitarias.
- Para la etapa de desarrollo y pruebas en los futuros desarrollos, se recomienda implementar un ambiente de pruebas y certificación similar al ambiente productivo.
- Tener conocimiento relacionado con el desarrollo y las tecnologías presentadas en la propuesta arquitectónica final, estoy incluye servidores de aplicaciones, y frameworks de desarrollo.
- Si bien, se sugiere que el framework de desarrollo sea React, se recomienda considerar Flutter como la opción válida para las aplicaciones que se proyecten a entornos multiplataformas.

Trabajos futuros

- Implementar servicios de tracking en el Api Gateway, con la finalidad de facilitar el seguimiento a las peticiones y respuestas generadas por cada servicio, evaluando también sus tiempos de respuesta.
- Implementar herramientas para la automatización de tareas (compilación, pruebas, despliegue y monitorización) escalando la arquitectura propuesta hacia DevOps. Esto permitirá desarrollos en ciclos más cortos y de alta calidad.

10. Bibliografía

[1] Cayetano Medina Molina, Manuel Rey Moreno, Victor Cazurro Barahona, and Sergio Parrondo, "The adoption of mobile banking applications from a dual perspective," *Revista Universidad de Valladolid*, 2019, [Online]. Available: https://revistas.uva.es/index.php/sociotecno/article/view/3639

[2] Dpto Económico, "El avance de la banca digital en Ecuador," Ecuador, 2022. [Online]. Available: https://asobanca.org.ec/estudios-especiales/

[3] Ignacio Leiva Mundaca and Marco Villalobos Abarca, "Método ágil híbrido para desarrollar software en dispositivos móviles," *Ingeniare. Revista chilena de ingeniería*, 2021, [Online]. Available: https://www.scielo.cl/scielo.php?pid=S0718-33052015000300016&script=sci_arttext&tlng=pt

[4] R. Arregui, Arregui Guerrero, and K. Ponce, *INCLUSIÓN FINANCIERA Y DESARROLLO Situación actual, Retos y desafíos de la banca*, 1era edición. Ecuador: Universidad Espíritu Santo, 2022. [Online]. Available: https://www.superbancos.gob.ec/bancos/wp-content/uploads/downloads/2020/09/LIBRO-INCLUSION-FINANCIERA-Y-DESARROLLO-3.pdf

[5] Maximiliano Cristiá, "Introducción a la Arquitectura de Software," *Universidad Nacional de Rosario*, 2007, [Online]. Available: https://www.fceia.unr.edu.ar/~mcristia/Introduccion_a_la_Arquitectura_de_Software.pdf

[6] Patricio Letelier and Mª Carmen Penadés, "Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)," *Universitat Politècnica de València*, 2021, [Online]. Available: https://www.researchgate.net/publication/26428496_Metodologias_agiles_para_el_desarrollo_de_software_eXtreme_Programming_XP

[7] María Julia Blas, "Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube," *Santa fé - Argentina*, 2019, [Online]. Available: https://ri.conicet.gov.ar/handle/11336/125130

[8] Resm Cibh, *Introducción a la Arquitectura de Sofware*. Buenos Aires, 2004. [Online]. Available: "https://www.academia.edu/download/64699769/Arquitectura_software.pdf https://es.scribd.com/document/560954714/Arquitectura-Software-With-Cover-Page-v2"

[9] Oscar Blancarte Iturralde, *Introduccion a la arquitectura de software, un enfoque práctico*, 1era ed. Mexico, 2020.

[10] Luis Felipe Fernandez, "Arquitectura de Software," *Software Guru*, 2006, [Online]. Available: https://ozarate.net/articulos/arquitectura_sw_sg_2006.pdf

[11] Oscar Tinoco Gómez, Pedro Pablo Rosales López, and Julio Salas Bacalla, "Criterios de selección de metodologías de desarrollo de software," *Revista de la Facultad de Ingeniería Industrial*, 2010, [Online]. Available: https://www.redalyc.org/pdf/816/81619984009.pdf

[12] Navarro Mirta Elizabeth, M. P. Moreno, J. Aranda, L. Parra, and J. R. Rueda, "Arquitectura de software en el proceso de desarrollo ágil: una perspectiva basada en requisitos significantes para la arquitectura," Departamento de Informática - F.C.E.F. y N. - U.N.S.J, 2018. [Online]. Available: http://sedici.unlp.edu.ar/handle/10915/67795

[13] José H. Canós, Patricio Letelier, and Mª Carmen Penadés, "Metodologías Ágiles en el Desarrollo de Software," *DSIC -Universidad Politécnica de Valencia*, 2005, [Online]. Available: http://aleteya.cs.buap.mx/~jlavalle/papers/agileMethodology/TodoAgil.pdf

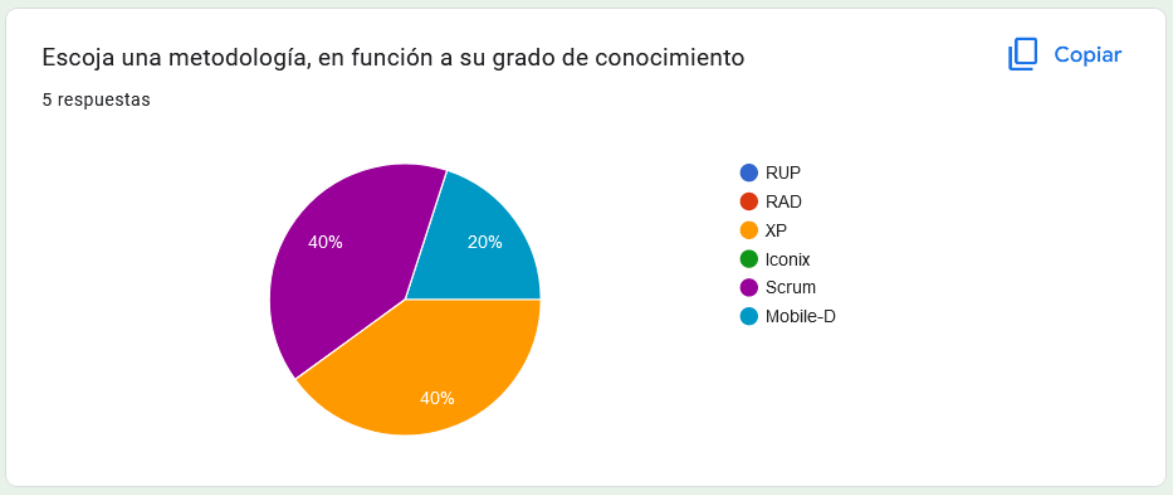
[14] Bryan Molina M, Harry Vite C, and Jefferson Dávila C, "Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software," *ResearchGate*, 2018, [Online]. Available: https://gc.scalahed.com/recursos/files/r161r/w25597w/438760423-269-823-1-PB-pdf.pdf
[15] Ian Sommerville, *Ingeniera de Software*, 9na ed. 2011.
[16] Roger S. Pressman, *Ingeniera de Software. Un enfoque práctico*, 7ma ed. 2010.
[17] Barry Boehm, "The Spiral Model as a Tool for Evolutionary Acquisition," *University of Southern California*, 2014, [Online]. Available: https://www.researchgate.net/publication/228805054_The_Spiral_Model_as_a_Tool_for_Evolutionary_Acquisition
[18] Rivas Carlos I., Corona Verónica P., Gutierrez José F., and Hernández Lizeth, "Metodologías actuales de desarrollo de software," *Revista Tecnología e Innovación*, México, 2015. [Online]. Available: https://www.ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num5/Tecnologia_e_Innovacion_Vol2_Num5_6.pdf
[19] Andrés Navarro Cadavid, Juan Daniel Fernández Martínez, and Jonathan Morales Vélez, "Revisión de metodologías ágiles para el desarrollo de software," *Red de Revistas Científicas de América Latina, el Caribe, España y Portugal*, Colombia, 2013. [Online]. Available: https://www.redalyc.org/pdf/4962/496250736004.pdf
[20] A. López Gil, "Estudio comparativo de metodologías tradicionales y ágiles para proyectos de Desarrollo de Software," Valladolid, 2018. [Online]. Available: https://uvadoc.uva.es/handle/10324/32875
[21] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, and M. Fowler, *Manifesto for agile software development*. 2001. [Online]. Available: https://zbook.org/savetopdf/MTAyMw==
[22] K. Beck, *Extreme Programming Explained: Embrace Change* , 1ª ed. Stoughton, 1999.
[23] R. Jeffries, A. Anderson, and C. Hendrickson, "Extreme Programming Installed," *Addison-Wesley*, 2001.
[24] Rojas, "Programación Extrema," *CodeJobs*. http://www.codejobs.biz/es/blog/2013/06/05/ programacion-extrema-xp#sthash.7z7S2a2S.dpbs
[25] Heredia Taipe, Ana Gabriela, Chiliquinga Yugcha, and Betty Leonor, "Desarrollo de un sistema de información utilizando herramientas open source y la metodología rup para el control y administración de los recursos del centro de desarrollo infantil rayitos de luz del barrio Laigua de Maldonado de la parroquia Aláquez del cantón Latacunga provincia de Cotopaxi.," UTC, LATACUNGA . [Online]. Available: http://repositorio.utc.edu.ec/handle/27000/1406
[26] R. Campaña, "El proceso de desarrollo rápido de aplicaciones de software," *ResearchGate*, 2010, [Online]. Available: http://dspace.unach.edu.ec/bitstream/51000/10023/1/Desarrollo%20de%20una%20plataforma%20web%20para%20recorridos%20virtuales%20360%c2%b0%20mediante%20la%20metodolog%c3%ada%20RAD.pdf
[27] L. Debrauwer and F. Heyde, *UML 2 Iniciación, ejemplos y ejercicios corregidos*, Segunda edición. 2014. [Online]. Available: https://www.laccei.org/LACCEI2014-Guayaquil/RefereedPapers/RP246.pdf
[28] O. Rodriguez and R. Socorro, "Seguridad y usabilidad de los esquemas y técnicas de autenticación gráfica," *Revista Cubana de Ciencias Informáticas*, 2018.
[29] D. Supan, K. Tekovic, J. Skalec, and Z. Stapic, "Using Mobile-D Methodology in Development of Mobile Applications: Challenges and Issues," *CROSBI*, 2013.

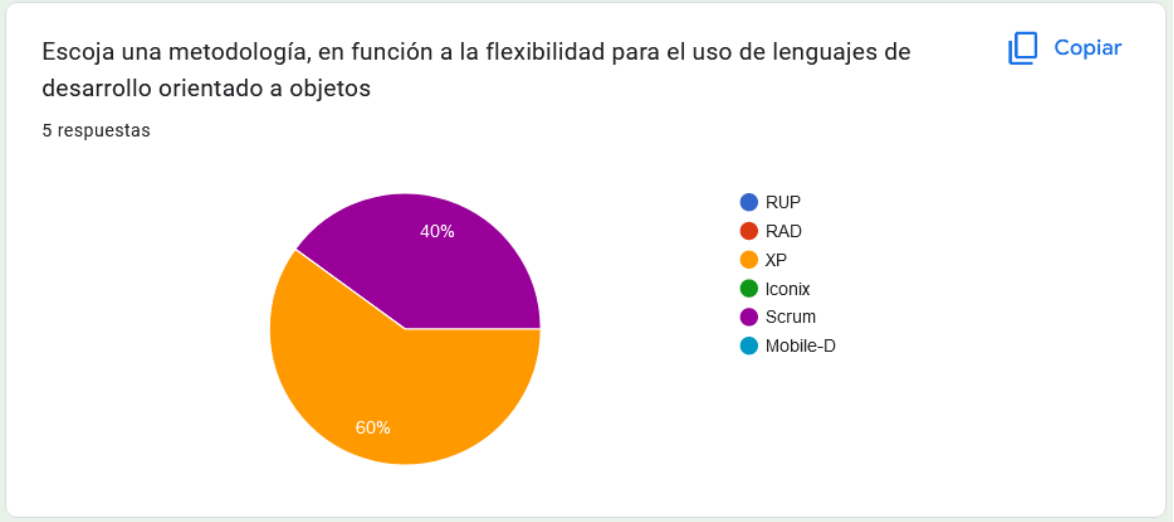
[30] J. L. Sardasht Mahmood, “An Investigation into Mobile Based Approach for Healthcare Activites,” *Proceedings of The International Conference on Software Engineering Research and Practice*, 2013.
[31] E. Alsabi and A. Dahanayake, “Smart Modeling for Lightweight Mobile Application Development Methods,” *New Trends in Databases and Information Systems. Communications in Computer and Information Science*, 2016.
[32] A. Asfour, S. Zain, N. Salleh, and J. Grundy, “ Exploring Agile Mobile App.,” *International Journal of Technology in Education and Science (IJTES)*, 2019.
[33] Amaya B, “Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles,” *Journal Technology*, 2017.
[34] Robert Ramírez Vique, “Métodos para el desarrollo de aplicaciones móviles,” *Universitat Oberta de Catalunya*, 2019, [Online]. Available: http://190.57.147.202:90/jspui/bitstream/123456789/464/1/Tecnologia_y_desarrollo_en_dispositivos_moviles.pdf
[35] “Flutter documentation,” *Introduction to widgets[Online]*. https://flutter.dev/docs/development/ui/widgets-intro
[36] Rodriguez Victor, “Desarrollo de aplicaciones móviles multiplataforma con Flutter,” UNIVERSIDAD DE ALMERIA, 2019. [Online]. Available: http://repositorio.ual.es/bitstream/handle/10835/8010/TFG_VAZQUEZ%20RODRIGUEZ,%20VICTOR.pdf?sequence=1
[37] Arvind Ravulavaru, *Learning Ionic 2*, 2nd ed. Packt Publishing, 2017.
[38] Bonnie Eisenman, *Learning React Native: Building Native Mobile Apps with JavaScript*, 2nd ed. O’Reilly, 2017.
[39] Jim Bennett, *Xamarin in Action*. Manning Publications, 2018.
[40] Ramiro Higonet Lang, “Hardware Security API,” Universidad Nacional de La Pampa, 2022. [Online]. Available: https://repo.unlpam.edu.ar/handle/unlpam/8288
[41] Edgar Rios P. and Wilson Suntaxi L., “Desarrollo de un sistema informático para los procesos de cosecha y post cosecha de la camaronera Pampas de Cayanca,” Escuela Politecnica Nacional, Quito, 2008. [Online]. Available: https://bibdigital.epn.edu.ec/handle/15000/1072#:~:text=El%20Desarrollo%20de%20un%20Sistema,de%20los%20despachos%20y%20liquidaciones

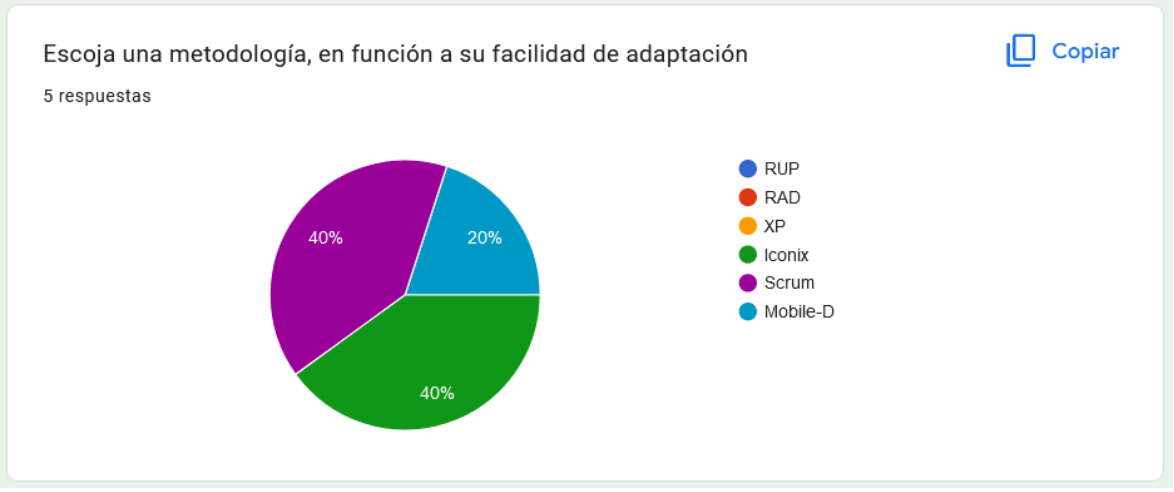
11. Anexos

Anexo 1. Resultado de la encuesta realizada para evaluar el nivel de conocimiento relacionado con las metodologías de desarrollo

Enlace: https://forms.gle/VfYreqALYSq4phLf7



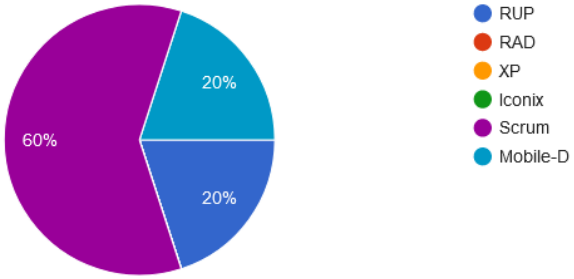




Escoja una metodología, en función a la flexibilidad para la aplicación de casos de uso

Copiar

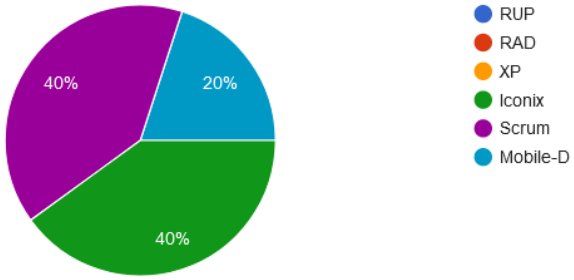
5 respuestas



Escoja una metodología, en función a la valoración de la documentación conocida

Copiar

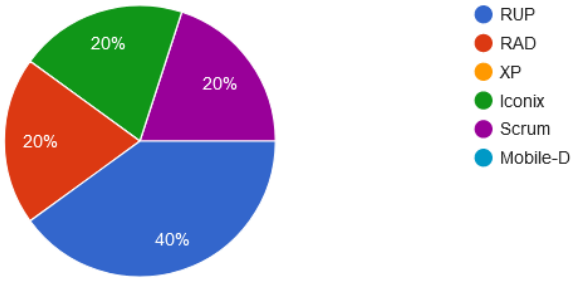
5 respuestas



Escoja una metodología, en función a la facilidad para la integración entre etapas de desarrollo

Copiar

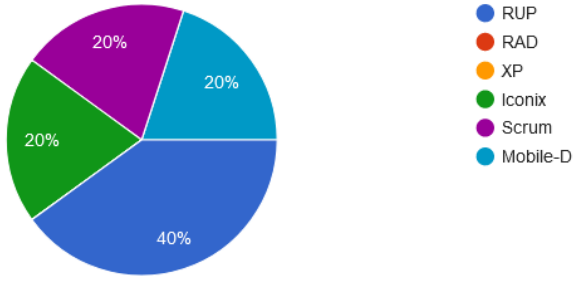
5 respuestas



Escoja una metodología, en función a la facilidad a la relación de la metodología con UML

Copiar

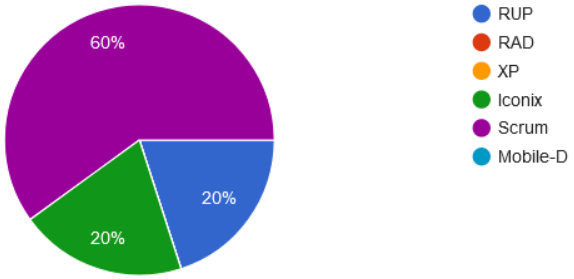
5 respuestas



Escoja una metodología, en función a su integración con cualquier tecnología de desarrollo de software

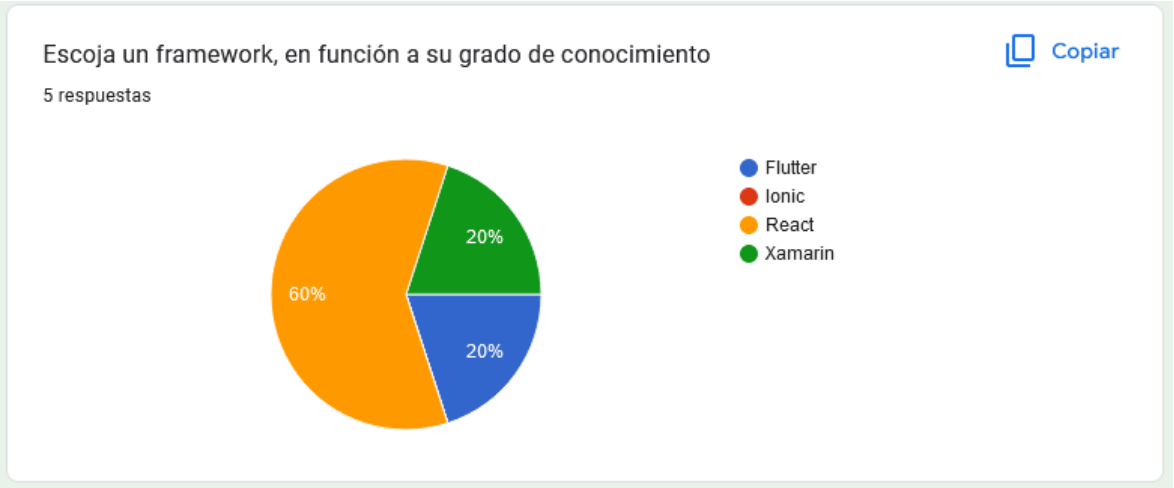
Copiar

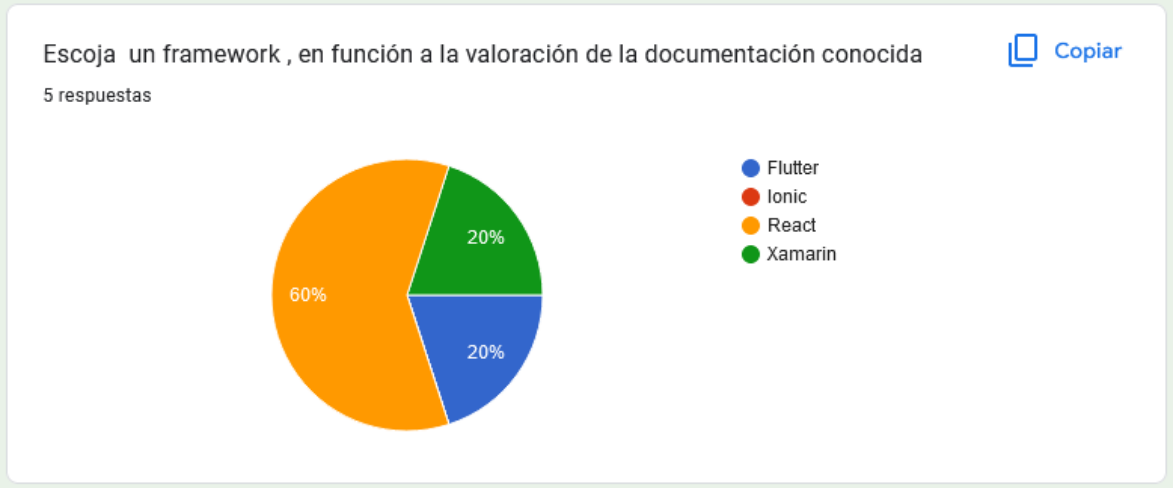
5 respuestas

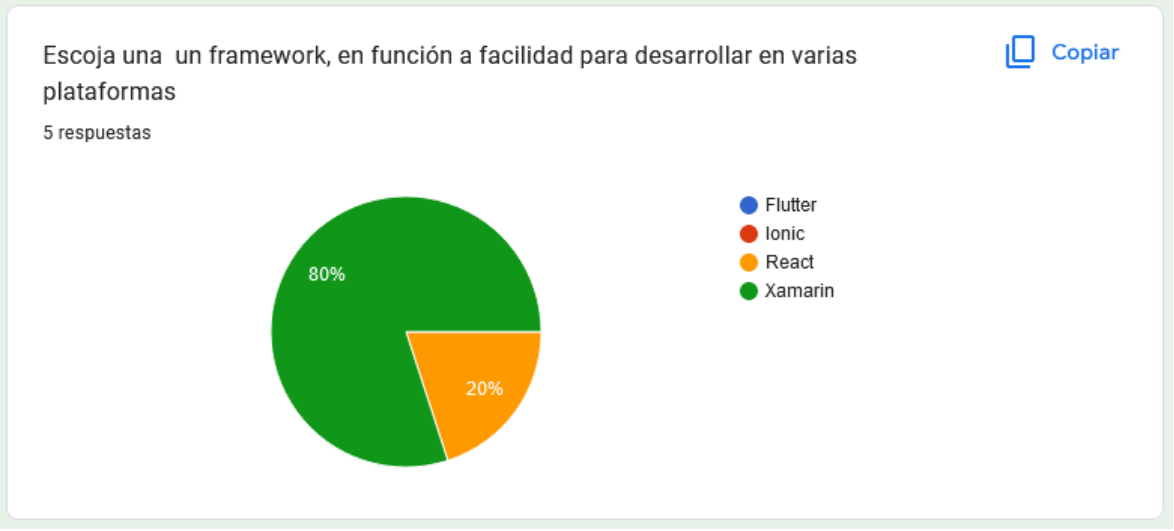


Anexo 2. Resultado de la encuesta realizada para evaluar el nivel de conocimiento relacionado con los frameworks de desarrollo

Enlace: https://forms.gle/yPQ5c5YED3WDf9Q68



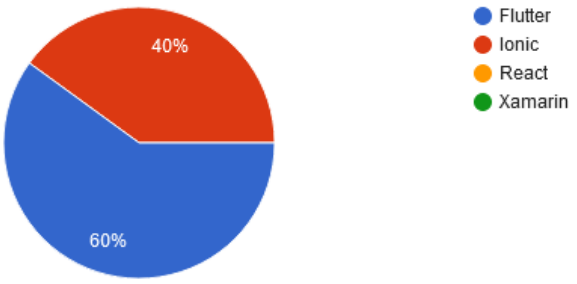




Escoja una un framework, en función a la estandarización de datos y estructuras

Copiar

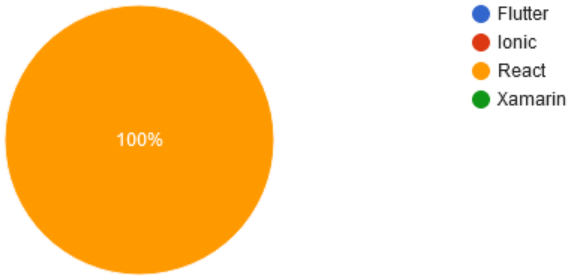
5 respuestas



Escoja una un framework, en función al rendimiento y la facilidad para desarrollos rápidos

Copiar

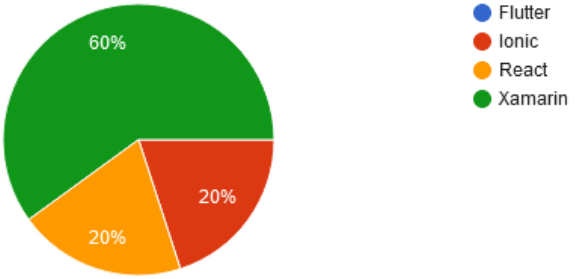
5 respuestas



Escoja una un framework, en función a la facilidad para el desarrollo de aplicaciones híbridas

Copiar

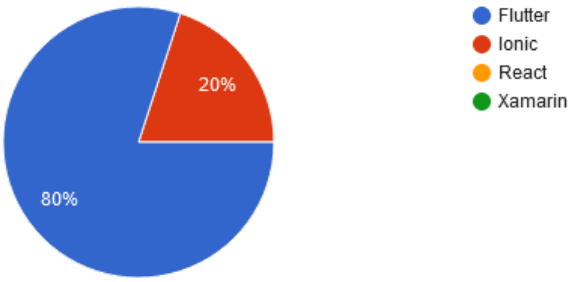
5 respuestas



Escoja una un framework, en función a la cantidad de elementos personalizables

Copiar

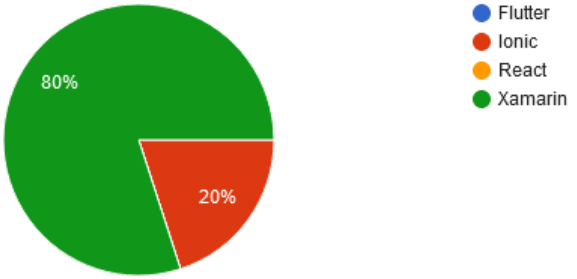
5 respuestas



Escoja una un framework, en función a la facilidad para la ejecución de pruebas y depuración

Copiar

5 respuestas



Anexo 3. Acta de reunión para levantamiento de situación actual y alcance de requisitos

Cooperativa de Ahorro y Crédito
15 de Abril
Seguridad y Confianza

ACTA DE REUNION

Fecha de reunión	01 de febrero de 2023
Hora de inicio	10:00 a.m.
Hora de finalización	11:30 a.m.
Convocada por	Emilio Vera Meza
Tipo de reunión	Revisión de infraestructura y requerimientos
Asistentes	Emilio Vera Meza (Jefe de desarrollo) Carlos Moreira Ubillus (Administrador de infraestructura)

Antecedente:

Como parte del proyecto de titulación para la Maestría en Ingeniería en software impartida por la Universidad Nacional De Loja, se solicita una reunión para analizar y documentar el modelo de desarrollo y despliegue que se utiliza actualmente en las aplicaciones web de la Cooperativa. Además de dimensionar los requerimientos deseados para desarrollos posteriores.

Análisis:

Se revisó la aplicación de onboarding que se encuentra publicada en internet, y se determinó lo siguiente:

- Se tiene un modelo de autenticación basado en AD.
- El despliegue de esta aplicación esta dividido en BackEnd y FrontEnd.
- El intercambio de información entre BackEnd y FrontEnd se realiza mediante SOAP.
- Únicamente el FrontEnd tiene salida directa al internet por medio de NAT a nivel de firewall.
- Toda la infraestructura se encuentra virtualizada.
- La tecnología utilizada para el desarrollo es JEE.

Además, se pudo identificar requerimientos deseables en implementaciones posteriores:

- Centralizara los requerimientos en un solo equipo, que tenga la capacidad de atender peticiones REST/SOAP.

Pág. 1 | 2

- Establecer un sistema de autenticación centralizada para todas las aplicaciones.
- Permitir que las aplicaciones se publiquen en múltiples instancias con la finalidad de que trabajen a manera de clúster y se garantice alta disponibilidad.
- Establecer un mecanismo de balanceo de carga, que permita parametrizar niveles de prioridad a los requerimientos, y que estos sean procesados tanto de manera síncrona como asíncrona.

Emilio Vera Meza
Jefe de desarrollo

Carlos Moreira Ubillus
Administrador de infraestructura

Anexo 4. Cotización de equipo HSM

prosupply

PROPUESTA DE SERVICIOS

OFERTA ECONÓMICA

Prosupply proveerá de los equipos Luna EFT de acuerdo a las siguientes alternativas:

Por la función que realiza el HSM, es recomendable la compra de dos equipos, que estaría igual en producción y a la vez actuaría como back-up.

HSM LUNA EFT PL60			
Cantidad	HSM	Precio Unitario	Total USD
1	LUNA-EFT (PH EFT). TCPIP, PL60: MARKII or AMB and power cord. Versión 2.3.0 Para trabajar en: Balanceo de carga y test	$ 13,583.00	$13,583.00
1	Soporte - Plan de Soporte estándar tres años	$ 5,788.00	$5,788.00
1	Soporte - Plan de Soporte estándar un año	$ 2,539.00	$2,539.00
1	Instalación y Capacitación	$ 399.00	$399.00
		INVERSION 1 EQUIPO + SOPORTE 3 AÑOS + INST.	$19.770,00
		INVERSION 1 EQUIPO + SOPORTE 1 AÑO + INST.	$ 16.521,00

*Valores No incluyen IVA

Anexo 5. Certificación que avala la traducción del resumen del TT.

Francisco Ricardo Vera Vélez

Licenciado en Ciencias de la Educación: Mención Ingles

Registro Profesional SENESCYT N° 1009-11-1064517

CERTIFICA

Que, en la ciudad de Portoviejo, a los 25 días del mes de abril de dos mil veintitrés, se ha procedido a realizar una revisión del documento RESUMEN del Trabajo de Titulación denominado "Diseño de arquitectura para el desarrollo y despliegue de aplicaciones móviles transaccionales, dirigido a la Cooperativa de Ahorro y Crédito 15 de abril Ltda." Perteneciente al señor Emilio José Vera Meza, con número de cédula: 1312598269. Documento que consta de 317 palabras en español y 294 palabras en idioma inglés, el cual está traducido en su integridad, manteniendo el mismo mensaje de su originalidad en español.

Digitally signed by FRANCISCO RICARDO VERA VELEZ
Date: 2023.04.25 21:15:30 COT

Lic. Francisco Vera Vélez Mg Ge

ESL Teacher

131174690-1