CONSTRUCCIÓN DE UNA PLATAFORMA, MÓVIL PARA EL APRENDIZAJE DE DEVOPS, COMO ESTRATEGIA DE MEJORES PRÁCTICAS EN EL DESARROLLO DE SOFTWARE, ORIENTADO A ESTUDIANTES DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD CATÓLICA DE ORIENTE.

JANNET VIVIANA BONILLA VALENCIA.

UNIVERSIDAD CATÓLICA DE ORIENTE

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS

RIONEGRO

2019

CONSTRUCCIÓN DE UNA PLATAFORMA, MÓVIL PARA EL APRENDIZAJE DE DEVOPS, COMO ESTRATEGIA DE MEJORES PRÁCTICAS EN EL DESARROLLO DE SOFTWARE, ORIENTADO A ESTUDIANTES DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD CATÓLICA DE ORIENTE.

JANNET VIVIANA BONILLA VALENCIA.

Trabajo para obtener el título de ingeniero de Sistemas

Asesor

MARIA VICTORIA SILVA DOMÍNGUEZ.

Universidad Católica de Oriente

Facultad de Ingenierías

Programa de Ingeniería de Sistemas

Rionegro

2021.

Nota de aceptación:

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Dedicatoria:

A mi familia por su apoyo irrestricto, por sus voces de aliento aún cuando las dificultades parecían hacerme desfallecer; por enseñarme a trabajar por cada sueño, y una vez alcanzado plantearme otro más ambicioso, haciendo de mi lo que soy; también por mostrarme que el amor sincero y desinteresado es realmente la fuerza que transforma el mundo.

Jannet Viviana Bonilla Valencia.

Agradecimientos:

A Dios, por darme la oportunidad de culminar exitosamente mis estudios, lo cual me ha significado éxito en la vida.

A mis padres Orlando y Nubia, por estar ahí en todo momento y ser garantes incondicionales para que este proceso pudiese culminar satisfactoriamente, además de ser guías con carácter, amor y sabiduría.

A la universidad Católica de Oriente, por hacerme parte de esto tan grande que se consolida como familia y permitirme crecimiento intelectual y humano.

A mi asesor y formadores, pues compartir sus conocimientos es lo que hace posible ser lo que hoy soy.

Y finalmente doy gracias a todos los que de una u otra forma intervinieron enriqueciendo mis experiencias con las suyas.

Jannet Viviana Bonilla Valencia.

Contenido

1. **Antecedentes:** 7

2. **Planteamiento del problema:** 9

3. **Justificación:** 11

4. **Objetivos:** 13

   4.1. General: 13
   4.2. Específicos: 13

5. **Marco teórico** 14

6. **Diseño metodológico** 26

7. **Resultados** 27

   7.1. Objetivo 1 28
   7.2. Objetivo 2. 28
   7.3. Historias de usuario. 29
   7.4. Diseño 29
   7.5. Desarrollo 30

8. **Conclusiones** 36

9. **Referencias bibliográficas** 38

10. **Anexos** 40

1. Antecedentes:

Hace apenas unos años llegaron plataformas online asociadas al aprendizaje de múltiples disciplinas, que se han presentado como las salvadoras y refuerzos en la materia, como lo son UDEMY, la cual está dirigida a adultos profesionales conducidos por tradicionales cursos de trabajo creados en el colegio; ésta utiliza contenido de creadores en línea para vender y así conseguir ganancias; además de proporcionar herramientas para los usuarios, creando cursos, que posteriormente se promueven y generan una rentabilidad con los ingresos producto de la matrícula de estudiantes.

De otro lado se encuentra MOOC, la cual es una plataforma que ofrece cursos en línea dirigidos a un número ilimitado de participantes a través de Internet según el principio de educación abierta y masiva; al cual se han unido universidades de todo el mundo ofreciendo miles de cursos en línea gratuitos. Lo que podría considerarse como una evolución de la educación abierta en Internet.

También existe EdX el cual es un proveedor sin fines de lucro de cursos en línea masivos y abiertos de carácter multidisciplinario.

Cómo éstas existen muchas plataformas, dedicadas al aprendizaje multidisciplinar, todo esto en busca de hacerle frente a los diferentes procesos de globalización que se han dado en los últimos años, mismos de los que el mundo y en general la nación no han estado exentos.

De ahí surge el deseo apremiante por aprender y profundizar en el desarrollo de software, todo esto en virtud de que, no estar a la altura del medio, representa no sólo sobrecostos en diferentes procesos, sino más atraso en los referentes de calidad de vida.

Así pues, la región del cercano Oriente Antioqueño no es ajena al cambio y se caracteriza por su resiliencia y flexibilidad, que, combinados con el importante auge tecnológico, han hecho que se convierta también en cuna de aprendizaje del desarrollo de software gracias a los campus de educación superior, que conscientes de las necesidades del entorno han incorporado en sus programas la ingeniería de Sistemas y afines.

Es cierto que se ha profundizado en la materia, y que se ha tratado de hacer frente a la imperante necesidad de programar, pero lo es aún más que con el esfuerzo que hacen las universidades por impartir buenos y sólidos conocimientos se está quedando corto, porque apenas se logra abarcar una pequeña parte de lo que es necesario en el entorno empresarial.

Por eso, se hace menester una herramienta que adicional a concentrarse en el aprendizaje de DevOps, permita tener soportes teóricos para principiantes y que se pueda portar y compartir con facilidad; pues a pesar de ser entornos digitales, no pueden accederse desde dispositivos móviles; y hoy, cuando el teléfono se ha convertido en parte fundamental de la cotidianidad de cada individuo es apremiante que también pueda emplearse como mecanismo de aprendizaje.

2. Planteamiento del problema:

Aprender a programar, se ha convertido casi que, en algo natural en los últimos años, gracias al cada vez más creciente auge tecnológico; pues según los resultados de una reciente encuesta realizada por el ministerio de Ministerio de Ciencia, Tecnología e Innovación, la programación es una herramienta fundamental en el medio, pues se ha convertido en una salida laboral para más del 50% de la población universitaria. Además, su aprendizaje constituye una oportunidad al mejorar el razonamiento lógico formal. (Ministerio de Ciencia, Tecnología e Innovación, 2018).

De otro lado, es aprender a programar estimula la perseverancia, la dedicación, el esfuerzo y la tenacidad; esto construye confianza y persistencia en niños, jóvenes y adultos que les permiten enfrentar nuevos desafíos y problemas en todos los órdenes de la vida; pero lo es más, que aún hoy, con la cantidad de información existente y las múltiples herramientas con las que se cuenta, existen grandes vacíos estructurales en el tipo, forma y fondo de la programación que se aprende, poniendo en desventaja y haciendo a los recién egresados universitarios menos competitivos y apetecibles en el sector (Valdés, 2016). Todo esto en parte también por la cada vez mayor desidia que presentan los jóvenes hacia la lectura y hacia los mecanismos de educación tradicional, lo que hace que su mente esté predispuesta y por consiguiente la atención sea nula; caso contrario cuando se emplean herramientas didácticas y cotidianas, que hacen que sin percibirlo la mente esté relajada y, por tanto, más receptiva, lo que acarrea un aprendizaje más eficaz que luego se refleja en los entornos empresariales.

A partir de ahí nace la necesidad de una guía asequible y de sencilla comprensión en cualquier etapa de aprendizaje del desarrollo (principiante, junior, senior), que permita a quienes desean potenciar el arte de la programación, hacerlo de la mejor manera, con buenas prácticas como el desarrollo de la mano de DevOps y enfocado a la obtención de código limpio; tan escaso en las universidades y tan apremiante en el sector productivo. Todo esto de la mano de estrategias didácticas y cotidianas, como lo es el uso de las tecnologías de la información y la comunicación, de tal forma que el aprendizaje sea dinámico; dada la cada vez más creciente aversión por el aprendizaje tradicional.

Pregunta de investigación.

¿Cuál es aporte para los estudiantes de ingeniería de Sistemas de la Universidad Católica de Oriente, disponer de una herramienta móvil que apunte al aprendizaje de DevOps como buena práctica en el desarrollo de software y código limpio, para el entorno empresarial, de una manera concisa y eficaz?

Hipótesis: ¿Cuál es la probabilidad de diseñar un aplicativo móvil que permita a los estudiantes de ingeniería de Sistemas de la Universidad católica de Oriente aprender DevOps de manera sencilla y eficaz?

3. Justificación:

Es innegable el hecho que la tecnología evoluciona a pasos agigantados y los es aún más el que la competitividad y la capacidad para estar constantemente en ese proceso de desaprender para aprender obliga a todos a desarrollar no sólo mayores habilidades, sino que también motiva a la producción de nuevas herramientas que de verdad generen valor como lo es el caso de los aplicativos móviles dedicados a la educación. Ellos requieren un entorno de gestión de contenidos de manera que se permita, crear y visitar contenidos formativos creando verdaderos entornos de enseñanza-aprendizaje.

A partir de esto nacen no sólo cambios drásticos en los estilos de vida, sino que también se producen avances importantes que logran trascender a diversas esferas y sin los cuales sería imposible siquiera pensar en solucionar problemas de la cotidianidad.

Ahora bien, conociendo el crecimiento exponencial que ha tenido el desarrollo de software en los últimos años, la esencialidad de la ingeniería como aportante de innovación y su impacto en la eficiencia y la optimización de diferentes procesos, se hace también menester el fortalecimiento de habilidades teóricas y prácticas que den paso a procesos lógicos un tanto más complejos que no sólo capaciten para el desarrollo, sino que permitan la formación de un pensamiento crítico enfocado en las buenas prácticas, lo que por consiguiente incrementaría su competitividad y requerimiento en el entorno productivo.

Los beneficios de éste están específica y directamente orientados a la población estudiantil, misma que está saliendo al sector productivo con grandes vacíos en su formación, pues llegan a entornos empresariales con más skills de un tecnólogo en sistemas, que el pensamiento propositivo y analítico, propio de un ingeniero, lo que se produce en parte por la ausencia de conocimiento más sólido, en el que deben participar activa y conjuntamente el estudiante, la academia y el sector mismo.

Además, se ve afectado positivamente el entorno empresarial, en la medida en que cada vez crece más la necesidad de personal y al no encontrar lo que requieren en el medio optan por contratar personas sin experiencia y formarlas ellos mismos en las competencias que son

apremiantes, pagando un costo elevado en ese proceso; así las cosas, con herramientas como éstas se permite no sólo egresar personas altamente capacitadas, sino que también es un aporte indirecto al sector económico al permitirles el ahorro de lo que invierten en la formación desde cero, redireccionando esos recursos a la potencialización de otros conocimientos y formación avanzada, concluyendo en personal más y mejor capacitado y un cambio cultural al transformar la forma en que se observa un ingeniero de Sistemas.

4. Objetivos:

4.1. General:

Construir una plataforma, móvil para el aprendizaje de DevOps, como estrategia de mejores prácticas en el desarrollo de software, orientado a estudiantes de ingeniería de Sistemas de la Universidad Católica de Oriente.

4.2. Específicos:

1. Determinar los requisitos funcionales y no funcionales del aplicativo.
2. Diseñar la arquitectura a emplear.
3. Desarrollar la plataforma móvil.

5. Marco teórico.

El desarrollo de aplicaciones ya sean de escritorio o para la web, requieren diferentes herramientas y software para su construcción; a continuación, se abordan las que se emplearán para el desarrollo de la aplicación:

Conceptos técnicos del software.

Cervantes, define la arquitectura del software como la estructura de un sistema, constituida por un conjunto de elementos con ciertas propiedades y relaciones. El Instituto de Ingeniería del Software se refiere a los elementos como a las diferentes entidades relacionadas en el sistema, es decir elementos que pueden ser entidades que existen en tiempo de ejecución como objetos o hilos, o entidades lógicas en tiempo de desarrollo como clases y finalmente a entidades físicas como nodos o directorios; así mismo se habla de un conjunto de relaciones que tienen una funcionalidad específica y que permiten el correcto funcionamiento.

Nace, basado en lo anterior, lo que es patrón de diseño, que se ajusta a un esqueleto de la solución a problemas usuales en el desarrollo de software; es decir, este brinda una solución probada y documentada respecto a constantes problemas identificados en el desarrollo de software y que aplican a ciertos contextos.
Se caracterizan por tener un nombre diferenciador, la descripción del problema, la solución para este y las consecuencias respecto a costo y beneficios al momento de aplicarlos.

Tedeschi (2016) propone la siguiente clasificación para los diferentes patrones de diseño:

- *Creacionales*: Permite inicializar y configurar objetos.
- *Estructurales*: Se encargan de separar interfaces de su implementación, busca que los objetos y las clases se agrupen para formar estructuras.
- *Comportamiento*: Describe la comunicación entre objetos y clases.

Intervienen los artefactos de software entendidos como el resultado parcial o final obtenido y usado en un proyecto, a través de los cuales se captura información.

Un artefacto puede ser un documento, un modelo o un elemento de un modelo que de acuerdo con el conjunto al que pertenezca es decir si hace parte de la fase de análisis, desarrollo, diseño o base de datos presenta información relacionada con la solución del problema identificado en los requerimientos, historias de usuario o la metodología de desarrollo que se esté empleando.

De otro lado, actúan también lo que son los lenguajes de programación definidos como el conjunto de instrucciones ordenadas y sucesivas que permiten ejecutar una tarea específica, estas instrucciones son llamadas "código fuente", el cual es único para cada lenguaje, además de estar diseñado para efectuar una función o propósito determinado.
Los lenguajes de programación manejan diferentes normas para manejar el comportamiento de un dispositivo y permitir el desarrollo de programas informáticos. (Morales, 2013).

Existen entre otros, JAVA, el cual es un lenguaje de programación orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. (Morales, 2013).
Su intención es permitir que los desarrolladores escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente. (Morales, 2013).

Convención y notación.

Sperberg define la convención como el conjunto de reglas que se aplican para lograr un consenso en la programación, de aplicar ciertas reglas que sean claras que permitan acelerar el desarrollo y lograr trabajar mejor en grupo. Así mismo por notación se entiende como el grupo de signos elegidos y empleados para expresar determinados conceptos de una disciplina o área en específico. Tedeschi (2016)

Entre otras, existe la notación Camel Case, la cual consiste en que cada palabra de la notación inicie con mayúscula si es una palabra compuesta, si es una palabra sencilla se da la posibilidad de que inicie con minúscula o mayúscula. Tedeschi (2016)

Conceptos metodológicos.

Las metodologías ágiles surgieron a partir del manifiesto agile lanzado en el 2001 por 17 expertos en gestión de proyectos y desarrollo de software, definiendo con él los 12 principios de esta metodología:

- *Prioridad*: satisfacer al cliente.
- *Los cambios son bienvenidos*: cualquier sugerencia o cambio se puede adaptar, siempre hay vuelta atrás.
- *Entregas periódicas:* el trabajo en fases define esta metodología, especialmente el trabajo por semanas.
- *Trabajo en equipo:* todos los equipos implicados tienen que trabajar conjuntamente.
- La comunicación mejor cara a cara.
- Para motivar a la gente lo mejor es crear un buen ambiente de trabajo.
- Medir el progreso es importante.
- Perfeccionismo y excelencia técnica juntando un buen diseño y una buena calidad del trabajo.
- Simplicidad
- Autogestión de cada equipo.
- Adaptación de los equipos a los posibles cambios de circunstancias.

Bajo este paraguas de principios se diseñaron unas metodologías que darían como resultado procesos más ágiles y eficaces como lo son Scrum, que es una de las metodologías ágiles que ayuda a gestionar proyectos de manera eficaz y se divide en fases:

"*Product Backlog*" en la cual se recoge en un documento lo que requiere el proyecto, las tareas y funciones necesarias para llevarlo a cabo. Este documento lo controlará el Product Owner (el encargado de que el equipo trabaje bien para conseguir los objetivos).

El, *Sprint Backlog*, que es donde se visualizan las tareas que finalmente se tienen que realizar y quién tiene que hacer cada una, se pueden asignar horas de trabajo o metas para que estas sean hechas de manera eficiente. Los llamados sprints suponen entregas parciales para ir viendo el producto y decidir cambios.

Todas las tareas y su evolución serán controladas en el momento del Burn Down.

Y la segunda, Kanban, la cual significa "tarjetas visuales" en japonés y dio nombre a un método inventado por Toyota para controlar sus métodos de producción.

Igual que la anterior técnica, con el Kanban también se dividen tareas, pero se ayuda de tarjetas visuales, como post it en una pizarra, para representar qué tiene que hacer cada uno y qué prioridad tiene cada tarea. Estas prioridades pueden cambiar, de ahí la facilidad de la movilidad de las tarjetas.

Otro punto fundamental es que por ciclo de trabajo hay delimitadas ciertas tareas, consiguiendo así que el equipo se centre en terminarlas y no en dejar varias abiertas.

Es un método compatible con el anterior, por eso es normal que se empleen juntos.

Así pues, es preciso hablar de lo que son las historias de usuario, las cuales Villamizar Suaza define como la descripción de una funcionalidad para un sistema de software que aporta un gran valor al usuario, son tarjetas en donde la persona interesada describe las características que el sistema debe poseer, ya sean requerimientos funcionales o no funcionales, las cuales deben ser comprensibles y delimitadas para que puedan construirse en un periodo de tres semanas.

Éstas se caracterizan por tener un nombre descriptivo y sencillo, poseer una breve descripción a manera de monólogo en el que el usuario determina la funcionalidad y unos criterios de aceptación. (Villamizar, 2017).

Conceptos pedagógicos.

Aprendizaje.

Según Gagné, el aprendizaje es un cambio en la disposición o capacidad de las personas que puede retenerse y que no es atribuible simplemente al proceso de crecimiento. (García, 2011).

Robbins de otro lado, señala que es cualquier cambio a la conducta que es relativamente permanente y que se da como consecuencia de una experiencia. (García, 2011).

Basado en eso, nacen lo que son las teorías de aprendizaje las cuales son construcciones teóricas por medio de las cuales se proponen las formas de aprendizaje del ser humano, integrando elementos biológicos, sociales, culturales y emocionales. (García, 2011).

Se encuentran algunas como el conductismo, el cual según su fundador John Watson, es una ciencia natural que se arroga todo el campo de las adaptaciones humanas. Para Skinner en cambio se trata de una filosofía de la ciencia de la conducta, que define varios aspectos esenciales de su objeto de estudio. Sin embargo, este objeto es entendido de diversos modos, según el enfoque conductista del cual sea parte. (García, 2011).

B.F. Skinner, el propulsor de la teoría, afirma que el lenguaje aprendido viene condicionado por la adaptación del exterior de las correcciones. producto de esta repetición, se van aprendiendo palabras asociadas a momentos y objetos determinados. lo aprendido es utilizado para satisfacer sus propias necesidades. (Carrillo, 2015).

Se encuentra también la teoría constructivista la cual tiene un el enfoque pedagógico; se sostiene que el conocimiento no se descubre, se construye: el alumno construye su conocimiento a partir de su propia forma de ser, pensar e interpretar la información. Desde esta perspectiva, el alumno es un ser responsable que participa activamente en su proceso de aprendizaje. (Carrillo, 2015).

El Constructivismo ha recibido aportes de importantes autores, entre los cuales se encuentran Jean Piaget, Vygotsky, Ausubel y Bruner. (Carrillo, 2015).

Piaget aporta a la teoría Constructivista el concebir el aprendizaje como un proceso interno de construcción, en donde el individuo participa activamente adquiriendo estructuras cada vez más complejas, a los que este autor denomina estadios. (García, 2011).

Un tema importante en la estructura teórica de Bruner es que el aprendizaje es un proceso activo en el cual los alumnos construyen nuevas ideas o conceptos basándose en su conocimiento corriente o pasado. El alumno selecciona y transforma información, construye hipótesis, y toma decisiones, confiando en una estructura cognitiva para hacerlo. La estructura cognitiva (es decir, esquemas, modelos mentales) provee significado y organización a las experiencias y permite al individuo ir más allá de la información dada. (Carrillo, 2015).

De otro lado, la teoría de aprendizaje por descubrimiento fue concebida por Jerome S. Bruner, y el espíritu de ella es la de propiciar la participación del alumno durante el proceso de enseñanza-aprendizaje, a partir de la consideración de que un aprendizaje efectivo depende, básicamente, de que un problema real se presente como un reto para la inteligencia del alumno, motivándolo a enfrentar su solución, y aún a ir más allá, hasta el fin primordial del aprendizaje que consiste en su transferencia. (Carrillo, 2015).

Resulta importante destacar el hecho de que, en la mayoría de los aspectos a tratar, Bruner coincida con las ideas expuestas por Jean Piaget y su colaboradora Barbel Inhelder. (Carrillo, 2015).

Para Bruner, el desarrollo intelectual depende directamente de que se dominen ciertas técnicas. En este dominio deben considerarse como determinantes dos factores: la maduración y la integración. (García, 2011).

La maduración le permite al alumno representarse al mundo de estímulos desde tres dimensiones, que se van perfeccionando de manera progresiva:

- La acción.
- La imagen.
- El lenguaje simbólico.

La integración consiste en el empleo de grandes unidades de información para la resolución de problemas.

Según García, 2011, Bruner también menciona la existencia de cuatro grandes ventajas en la manera heurística e hipotética de presentar el material de enseñanza:

- La potencia intelectual. El descubrir y resolver problemas por parte del alumno, habilita su capacidad de construcción y organización racional de los elementos de un problema.
- Las motivaciones intrínseca y extrínseca. El alumno se recompensa con los efectos de sus propios descubrimientos.
- El aprendizaje y la heurística del descubrimiento. Sólo se aprende realmente a través de la solución de problemas y el interés-esfuerzo por descubrir.
- La memoria. El alumno retiene con mayor facilidad lo aprendido si él mismo organiza sus materiales y procesos respectivos.

Conceptos objeto de aprendizaje.

DevOps.

Es uno de los términos más mencionados en el actual entorno de IT. Normalmente se asocia a estrategias de transformación digital, y a metodologías como Continuous Delivery o desarrollo ágil. (Ruiz, 2015)

“Es un acrónimo inglés de development (desarrollo) y operations (operaciones), que se refiere a una metodología de desarrollo de software que se centra en la comunicación, colaboración e integración entre desarrolladores de software y los profesionales de sistemas en las tecnologías de la información (IT)”. (Ruiz, 2015).

También es una respuesta a la interdependencia del desarrollo de software y las operaciones IT. Su objetivo es ayudar a una organización a producir productos y servicios software más rápidamente, de mejor calidad y a un coste menor. (Valdés, 2016).

Así pues, se puede decir que DevOps es una metodología para creación de software, se basa en la integración entre desarrolladores software y administradores de sistemas y permite fabricar software más rápidamente, con mayor calidad, menor coste y una altísima frecuencia de releases. (Ruiz, 2015).
Con esto en mente, se pueden repasar corrientes de opinión en torno a DevOps, pues hay quienes lo consideran una cultura y quienes lo observan como una profesión.

Mucho se ha hablado de DevOps como cultura y aunque no es en sí una, sí requiere de un fuerte cambio cultural y organizativo para su implementación, orientado hacia la colaboración, la comunicación, y en último término la completa integración entre las antiguas áreas (en lo habitual rabiosamente estancas) de desarrollo y sistemas. (Valdés, 2016).

Este cambio cultural es tan complicado de conseguir en algunas organizaciones, que son muchos los que lo identifican directamente con DevOps, pero es preciso recordar que DevOps es una metodología de desarrollo software, y un cambio de cultura no es en sí mismo una forma de desarrollar software. (Valdés, 2016).

Otro error común es confundir DevOps con modelos que algunos startups se ven abocados a adoptar en sus inicios, en los que todos los miembros del equipo técnico saben de desarrollo, de sistemas, de tuning de rendimiento, de bases de datos y hasta de cablear la oficina. (Valdés, 2016).

Según Rob Steward, vicepresidente de desarrollo de producto de Progress Software, “una buena práctica de DevOps liberará a los desarrolladores para que se centren en hacer lo que mejor saben hacer: escribir software. DevOps elimina el trabajo y las preocupaciones de la puesta en producción del software una vez que está escrito”. (Steward, 2018).

Para un desarrollador pasar a un modelo DevOps resulta inmediato, mientras que un ingeniero de sistemas necesita nuevas habilidades. Estas habilidades, según una investigación de Puppet Labs, son, por este orden: scripting, don de gentes, reingeniería de procesos, y en último lugar experiencia con herramientas específicas. Un perfil que no es fácil de encontrar. (Valdés, 2016).

En ese orden de ideas DevOps no es una profesión, y estrictamente no existen ni perfiles DevOps ni ingenieros DevOps, sino “ingenieros de sistemas con capacidades específicas para integrarse en equipos DevOps”. (Valdés, 2016).

De otro lado, como DevOps pretende ser un modo de trabajo interfuncional, en lugar de una sola herramienta, por eso existen conjuntos (o "toolchains") de múltiples herramientas con las cuales es posible llevar a cabo dichos procesos, como lo es Jenkins, el cual es un servidor de automatización open source escrito en Java, que está basado en el proyecto Hudson y es, dependiendo de la visión, un fork del proyecto o simplemente un cambio de nombre.

Éste ayuda en la automatización de parte del proceso de desarrollo de software mediante integración continua y facilita ciertos aspectos de la entrega continua. A demás, admite herramientas de control de versiones como CVS, Subversion, Git, Mercurial, Perforce y Clearcase y puede ejecutar proyectos basados en Apache Ant y Apache Maven, así como secuencias de comandos de consola y programas por lotes de Windows. El desarrollador principal es Kohsuke Kawaguchi. Publicado bajo licencia MIT, Jenkins es software libre.1

En el proceso de DevOps intervienen diversos conceptos como son:

El código, con el desarrollo y revisión de código, herramientas de administración de código fuente y la fusión de este. La construcción, para la cual se emplean herramientas de integración continua y el estado de la compilación; la prueba, a través de herramientas de prueba continuas que brindan retroalimentación sobre los riesgos comerciales; el paquete, el cual es un repositorio de artefactos, distribución previa a la implementación de la aplicación, lanzamiento - gestión de cambios, aprobaciones de versiones y automatización de versiones; la configuración y gestión de la infraestructura, donde interviene la infraestructura como código; El monitoreo del rendimiento de las aplicaciones, experiencia del usuario final a través de monitor.

Algunas categorías son más esenciales en una cadena de herramientas DevOps que otras; especialmente la integración continua para lo que se emplea Jenkins y la infraestructura como código en lo que se usa Puppet.

DevOps tiene además otros enfoques como lo es el concepto de agilismo, debido a que la necesidad de DevOps surgió del creciente éxito del desarrollo de software ágil, ya que eso llevó a que las organizaciones pudieran lanzar su software más rápido y con una frecuencia significativamente alta. A medida que trataban de superar la tensión que esto suponía para sus procesos de gestión de versiones, debían adoptar patrones como la automatización del lanzamiento de aplicaciones, las herramientas de integración y la entrega continuas.

Así pues, la entrega continua y DevOps tienen objetivos comunes y a menudo se usan en conjunto, pero hay diferencias sutiles, ya que, si la entrega continua se centra en la automatización de los procesos de entrega de software, DevOps también se centra en el cambio de la organización para admitir una gran colaboración entre las muchas funciones involucradas.

A demás, comparten una base común en métodos ágiles y pensamiento delgado: cambios pequeños y frecuentes con valor focalizado para el cliente final.

Existen además otras corrientes derivadas y por ende estrechamente relacionadas con el área de DevOps como lo es ArchOps, el cual es una extensión de DevOps que incrementa el

nivel de abstracción al priorizar los artefactos de arquitectura de software por encima del código fuente para el despliegue y operación de soluciones de software. Aquí se establece que los modelos de arquitectura son entidades de primera clase dentro del desarrollo, despliegue y operación de soluciones de software.

También la aplicación de entrega continua y DevOps para el análisis de datos se ha denominado DataOps, la cual busca integrar ingeniería de datos, integración de datos, calidad de datos, seguridad de datos y privacidad de datos con operaciones.

Ésta se aplica a principios de DevOps, desarrollo ágil y el control estadístico del proceso, utilizado en la fabricación ajustada, para mejorar el tiempo de ciclo de extracción de valor del análisis de datos.

Así pues, DevOps tiene unos objetivos que abarcan todo el proceso de entrega los cuales incluyen:

Frecuencia de despliegue mejorada.

Llegada al mercado más rápida.

Baja tasa de errores en nuevas versiones.

Tiempo de entrega más corto entre parches.

Tiempo de recuperación más rápido (en caso de que una nueva versión falle).

De ésta forma los procesos simples se vuelven cada vez más programables y dinámicos, utilizando un enfoque DevOps, los cuales tienen como objetivo maximizar la previsibilidad, eficiencia, seguridad y mantenimiento de los procesos operativos. Muy a menudo, la automatización apoya este objetivo.

La integración de DevOps se enfoca en la entrega de productos, pruebas continuas, pruebas de calidad, desarrollo de características y versiones de mantenimiento para mejorar la confiabilidad y la seguridad y proporcionar ciclos de desarrollo e implementación más rápidos. Muchas de las ideas y personas involucradas en DevOps provienen de la administración de sistemas empresariales y los movimientos ágiles de desarrollo de software.

Intervienen también patrones de arquitectura, como lo es el caso de los microservicios, ya que este tipo de enfoque permite a las empresas digitales brindar alta disponibilidad y estabilidad a sus aplicaciones; esto se debe a que todas las partes de las aplicaciones (base de datos, back-end, frontend, etc.) son independientes y, si una de ellas falla, no implica que toda la aplicación se caiga, en lugar de eso, las otras partes continuarán trabajando mientras se restaura el componente afectado. En DevOps se requiere para optimizar sus desarrollos, y dejar atrás arquitecturas monolíticas.

6. Diseño metodológico.

Inicialmente se recolecta información sobre DevOps, sus ventajas y formas de ejecución. Así como mecanismos de pedagogía mediante los cuales se garanticen formas adecuadas de enseñanza, aprendizaje y / o refuerzo de este, a través de la búsqueda en libros, internet y la asesoría de personas expertas en el tema para realizar una caracterización y diagnóstico en cuanto al tema.

Posterior, se indagan no sólo sobre patrones de diseño usados en aplicaciones móviles, sino que se fortalecen los principios SOLID, introducidos por Robert C. Martin en el año 2000 y descritos por él mismo como guías para crear código limpio garantizando así que el diseño sea lo más mantenible posible y que se puedan agregar diferentes funcionalidades cuando se precisen.

Se emplea adicional la metodología ágil SCRUM, la cual es definida por Schwaber y Sutherland (2013) como un marco de trabajo para el desarrollo y el mantenimiento de productos complejos, en donde las personas pueden abordarlos y entregar productos con un valor máximo para su negocio.

Su ejecución fue:

Sprint 1.
Definición de los requisitos funcionales y no funcionales del sistema.
Definición de la arquitectura.
Definición de frameworks y bases de datos.

Sprint 2.
Configuración de los entornos de desarrollo Android Studio (Mobile).
Creación del proyecto en el entorno.

Sprint 3.
Priorización de requisitos funcionales y no funcionales.
Definición de plugin a emplear.
Instalación de plugin en los ambientes de desarrollo.

Sprint 4.
Diseño de actividades que apunten al aprendizaje.

Definición de recursos emplear.

Sprint 5.

Configuración de conexión a la base de datos.

Implementación de registro y login de la aplicación.

Implementación de logOut.

Sprint 6.

Implementación funcionalidades generales.

Sprint 7.

Generación jars.

Documentación e informe final.

En la construcción del cronograma de actividades se emplea un Kanban manual, que contiene los diferentes estados en los que se podía encontrar una tarea. [Ver anexo 4].

Por último, se seleccionan 10 estudiantes de ingeniería de Sistemas, y se les permite una interacción con el aplicativo y posterior feedback a través de una encuesta calificada de 1 a 5, siendo 1 muy malo o bajo y 5 muy alto o bueno, todas con el mismo peso, para entender el impacto de tener herramientas de este tipo en su proceso formativo. [Ver anexo 5].

[Figura. Fuente: Autor].

7. Resultados

A continuación, se evocan los resultados por objetivo y un promedio de las respuestas entregadas por los usuarios de prueba:

[Figura. Fuente: Autor].

7.1. Objetivo 1

Requisitos funcionales y no funcionales del aplicativo.

RF1. Se permitirá el acceso directo a la aplicación.

RF2. Se podrán consultar instrucciones generales.

RF3. Se registrarán jugadores en la base de datos local.

RF4. Se podrá consultar todo lo relacionado a las instrucciones de la aplicación.

RF5. Se podrá jugar a través de preguntas y respuestas.

RF7. Será inicialmente para dispositivos Android.

RF8. Debe poder ser mantenible.

7.2. Objetivo 2.

Arquitectura.

Modelo vista controlador.

La finalidad principal de este tipo de arquitectura es la separación de los datos y la lógica de negocio de una aplicación de su presentación y el módulo encargado de gestionar los

eventos. MVC propone la construcción de tres componentes desacoplados: modelo, vista, controlador. [Ver anexo 1].

Modelo: Por medio de este componente se administra la información que precisa el sistema para operar, se hacen las gestiones de base de datos y se envía a la vista la información que es requerida en determinados momentos.

Controlador: En la aplicación responde a los eventos producidos por el usuario, transformándolos en peticiones al modelo.

Vista: Enseña la información y lógica de negocio en un formato adecuado para la interacción con la interfaz de usuario.

7.3. Historias de usuario.

- **HU 1.** Yo como usuario deseo ingresar a la aplicación para hacer uso de las funcionalidades.
- **HU 2.** Yo como usuario deseo consultar las instrucciones de la APP para entender su funcionamiento.
- **HU 3.** Yo como usuario deseo poder gestionar los usuarios dentro de la aplicación, al registrarlos, consultarlos y actualizar datos para manejo y mantenibilidad.
- **HU 4**. Yo como usuario deseo realizar ajustes en la vista para tener control y personalización.
- **HU 5**. Yo como usuario quiero poder personalizar el juego para accesibilidad en cuanto a colores.
- **HU 6**. Yo como usuario quiero poder jugar a través de preguntas y respuestas.

7.4. Diseño

[Ver apartado de resultados]

Patrones de diseño:

Abstract Factory: A través de una interfaz capaz de delegar la creación de un conjunto de objetos relacionados. [Ver anexo 2].

Adapter: Para permitir la conversión de un tipo de objetos a otro. [Ver anexo 3].

7.5. Desarrollo

Se empleó Android Studio para el desarrollo de la aplicación, así como el uso de fragments y entidades variadas para el entorno gráfico, provisionadas por el IDE en sí.

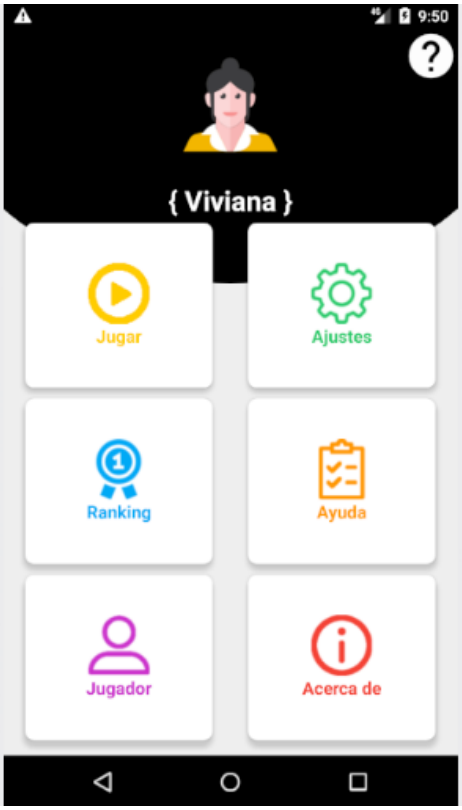
Se opta por acotar el alcance de la App a ser netamente offline y para un jugador; sin embargo, queda abierta la posibilidad de ser online y multijugador sólo cambiando la base de datos que se usa, pues actualmente se desarrolla en sqLite, pero se puede transformar el gestor de base de datos a online.

Inicialmente se pretende el aprendizaje de conceptos implicados en el proceso de DevOps, para esto se emplea la asociación de palabras con su significado, de tal forma que se conozca su existencia, y esto se convierta en una invitación a consultar acerca de los mismos; sin embargo se puede extender, para que en el tiempo ésta pueda buscar enlaces de estudio y sugerirlos a quien está jugando, así como la facilidad para el streaming con otros jugadores en lo que se permita estudiar en conjunto.

Posterior, se somete a calificación la aplicación, recibiendo un 75% de favorabilidad en la facilitación del aprendizaje, pues regala una visión más amplia de los procesos que intervienen en el desarrollo de software. También es preciso mencionar que entre los feedBacks recibidos se encuentran los identificados en la etapa inicial del proceso de desarrollo, los cuales fueron los enlaces de estudio y conexión multijugador, evitando así salir del aplicativo.

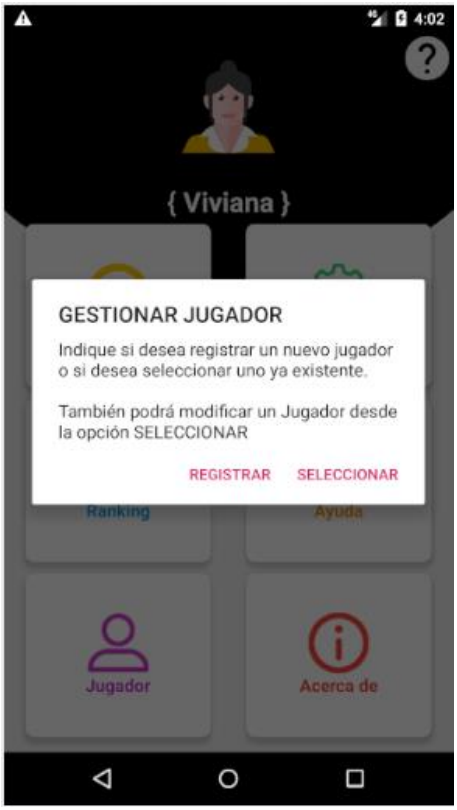
Así pues, se asevera que el aporte fue realmente importante, porque a través de la implementación se transforma la forma en que se estudia, pues si bien es cierto que existen herramientas orientadas al aprendizaje, muy pocas buscan que se haga a través del juego y mucho menos hay las que busquen dar nociones de lo que es DevOps.

Vistas de funcionalidades por historia de usuario.

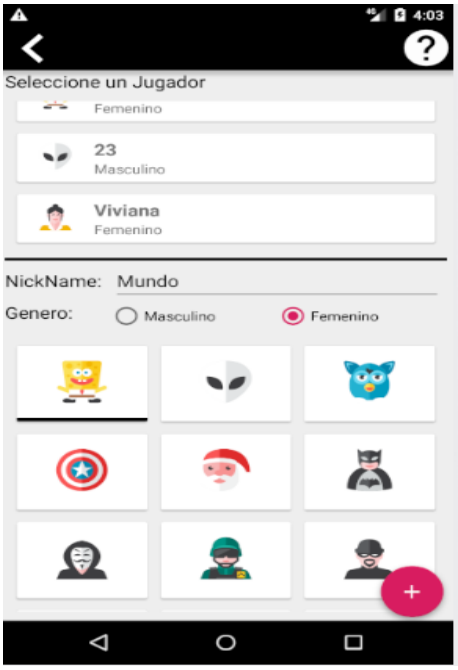


HU 1. *Ingreso Aplicación*. [Fuente: Autor].

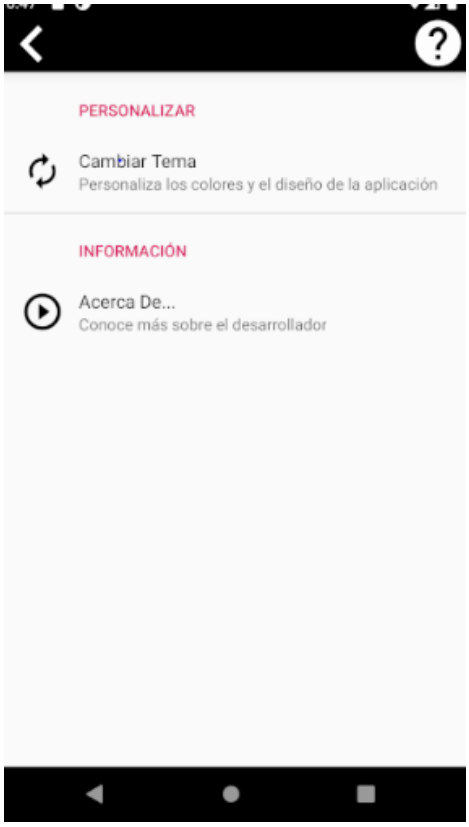
HU 2. *Instrucciones.* [Fuente: Autor].



HU 3. *Gestión jugadores*. [Fuente: Autor].



Intercambio jugadores. [Fuente: Autor].

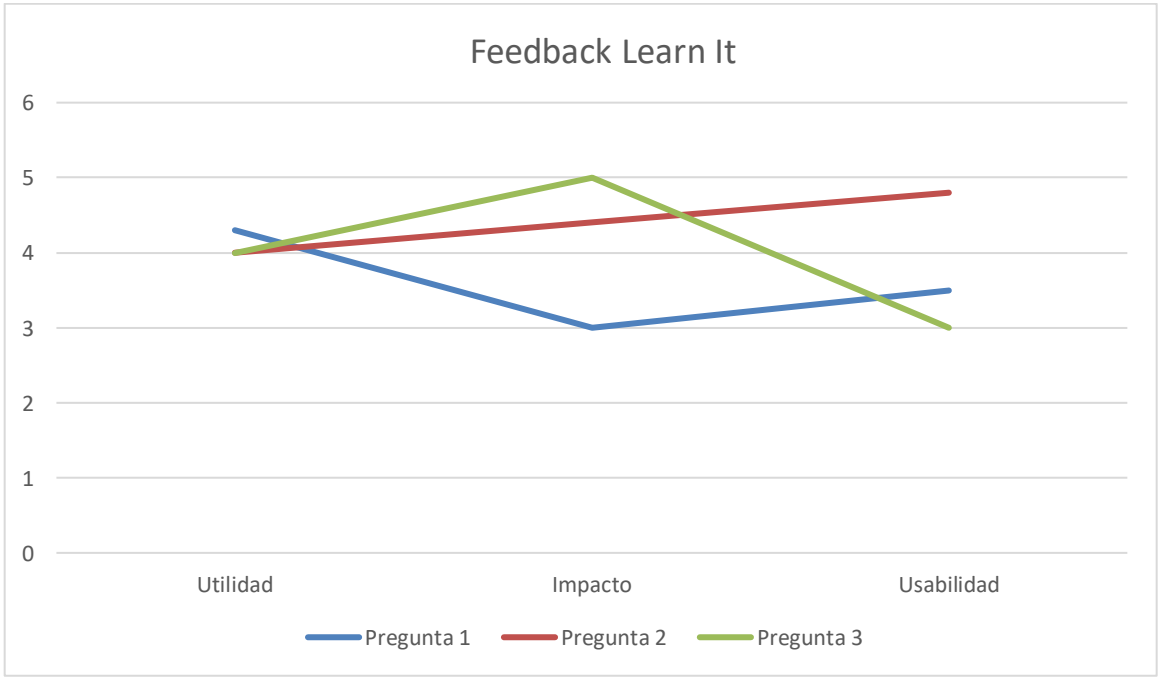


HU 4. *Ajustes*. [Fuente: Autor].

HU 5. *Tema*. [Fuente: Autor].

HU 6. *Preguntas.* [Fuente: Autor].

Feedback para la aplicación learn it.



Cuestionamientos llevados a cabo en la encuesta.

Pregunta 1. ¿Considera necesaria esta aplicación en su formación como ingeniero de sistemas?
Pregunta 2. ¿Es fácil entender el funcionamiento y objetivo de esta aplicación?
Pregunta 3. ¿Aprendió de DevOps al hacer uso de ésta app?

Cuadro promedio preguntas.

	Pregunta 1	**Pregunta 2**	**Pregunta 3**
Utilidad	4,3	4	4
Impacto	3	4,4	5
Usabilidad	3,5	4,1	3

En el cuadro anterior se evidencia el promedio de las respuestas a las preguntas elaboradas a 10 estudiantes de ingeniería de sistemas las cuales también dan como resultado final un 75% de aceptación a la aplicación, basada en tres pilares que son: utilidad, impacto y usabilidad, ya que esto es lo que permite saber si iniciativas como éstas aportan verdaderamente al desarrollo de profesionales más competitivos.

El 86% de los encuestados consideran necesarias iniciativas de este tipo como garantes y aportantes en la formación como ingenieros de sistemas, para el otro 14% es irrelevante si ocurren o no, pues aseveran que el medio no es directamente proporcional a aprendizaje; pero coinciden con los anteriores en que pueden ser útiles a largo y mediano plazo.

Con respecto al impacto, el 70% de las personas que respondieron la encuesta consideraron que ésta herramienta en específico impacta directa o indirectamente en los diferentes procesos de formación que se dan al interior de la Universidad Católica de Oriente, pero consideran que aun puede enriquecerse de otras funcionalidades que amplíen el campo de acción, tanto a las facilidades de conexión como dejar la posibilidad para que se hable en ella no sólo de DevOps sino que se den otros procesos que intervienen en el desarrollo de software.

Como último ítem se encuentra la usabilidad, lo cual arroja un 70% de calificación favorable, ya que a pesar de tener instructivo y usar variedad de colores, sería útil un manual de usuario que pudiesen descargar con el ánimo de no pasar por las instrucciones muchas veces.

Así pues, es evidente que este tipo de iniciativas sí van en dirección y están en sintonía con los procesos de aprendizaje y el cada vez más creciente auge tecnológico, porque a través de la implementación se transforma la forma en que se estudia y se integran actores que hacen parte de la cotidianidad, lo que hace que sea realmente sencillo de usar.

8. Conclusiones.

En el proceso de desarrollo de la aplicación móvil Learn It, se encuentran subprocesos que aun siendo secundarios por el momento en que se ejecutan, no son menos importantes, como lo es el hecho de la documentación vista como el desarrollo mismo, pues para quien programa es intuitivo porque conoce a ciencia cierta qué hace y las razones por las que lo hace, pero quien se ve afectado realmente es el usuario final, que al no conocer cómo está elaborado o la manera en que debería emplearse, no sólo pierde interés en la herramienta sino que el objetivo de la misma se ve opacado y en el peor de los casos no se cumple.

Con ésta aplicación se logra sentar un precedente, en cuanto a la forma en que se estudia, pues si bien es cierto que existen muchas herramientas que facilitan el aprendizaje, ninguna está orientada a aprender jugando, metodologías tan vastas como lo es DevOps; además, por la

forma en que está construida será sencillo enriquecerla en cualquier momento, no sólo con nuevas funcionalidades como la conexión multijugador, la sugerencia de enlaces para el estudio de los conceptos en sí o con otras más ambiciosas como el streaming sino también con diferentes conceptos que interfieran en el desarrollo de software.

Lo anterior no es sólo alentado por los resultados de quienes tuvieron la oportunidad de probar y posteriormente responder la encuesta, con una calificación positiva del 75% a partir de tres pilares los cuales fueron utilidad, impacto y usabilidad, sino que se hizo un ejercicio juicioso de pensar en un diseño que incluyera patrones a nivel arquitectónico y a nivel de la escritura de código.

De otro lado, es preciso mencionar el aporte que la elaboración de actividades académicas de este tipo tiene, pues no sólo es la puesta en práctica de conceptos adquiridos en el campus, sino que facilita el conocimiento de otros tantos que sólo se harán evidentes en el proceso de desarrollo; permitiendo visibilizar áreas en las que es menester un fortalecimiento.

Además, son estas actividades las que permiten que la formación sea integral, pues no es sólo sentarse a escribir código, sino que requiere de la formación de habilidades analíticas e investigativas que permitan aportar desde cada enfoque a la solución de problemáticas propias de cada contexto, desarrollando como consecuencia de ello, la capacidad para identificar problemas, generar preguntas cada vez más afinadas, interpretar, argumentar, analizar y hacer síntesis de la información proveniente de fuentes primarias o secundarias, potenciando el pensamiento crítico y otras como creatividad, observación, descripción y comparación, lo que facilita no sólo el tránsito por la universidad, sino por el entorno empresarial.

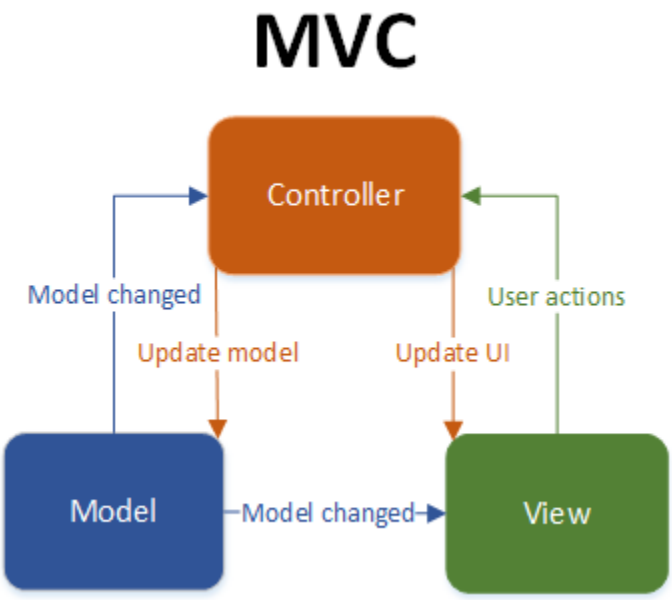
También se puede aseverar que permite una especie de extensión en la que nace la oportunidad de conectar a la universidad con el entorno inmediato; conexión que puede traducirse en alianzas estratégicas con el medio productivo, empresarial, y el gobierno.

9. Referencias bibliográficas

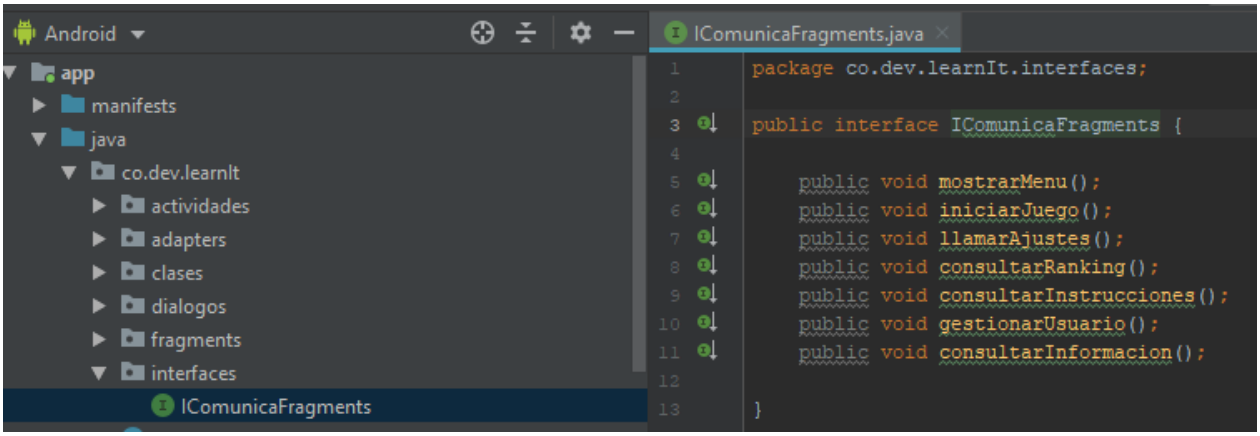
- Carrillo L. (2015). El arte de enseñar. Volumen 1°. Isbn13 9781506503424. 75-105 p
- García B. (2011). Aprendiendo de aprender. {online}.
- Martin R. (2008). Clean Code. Vol 1. {onLine}
- Ministerio ciencia tecnología e información. (2018). Impacto de los sistemas en el medio. Volumen 1°. 12-14 p. {onLine}.
- Morales, J, (2013). Metodología SCRUM. {onLine}.
- Ruiz J. (2015). Buenas prácticas en el desarrollo de software. Volumen 1°. 55-60 p. {onLine}.
- Schwaber y Sutherland (2013), Metodologías ágiles. {onLine}.
- Steward R, (2018). Programación orientada a objetos. Volumen 1°. 56-70 p

- Udemy. (2013). Plataforma Interactiva.. Recuperado de https://www.udemy.com/
- Valdés J. (2016). Desarrollo de software a la medida. Edición 1. 34-44 p.

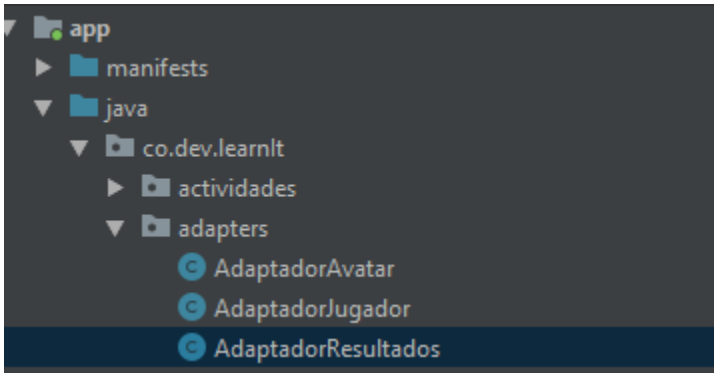
10. Anexos.



Anexo 1. [Figura. {onLine}].

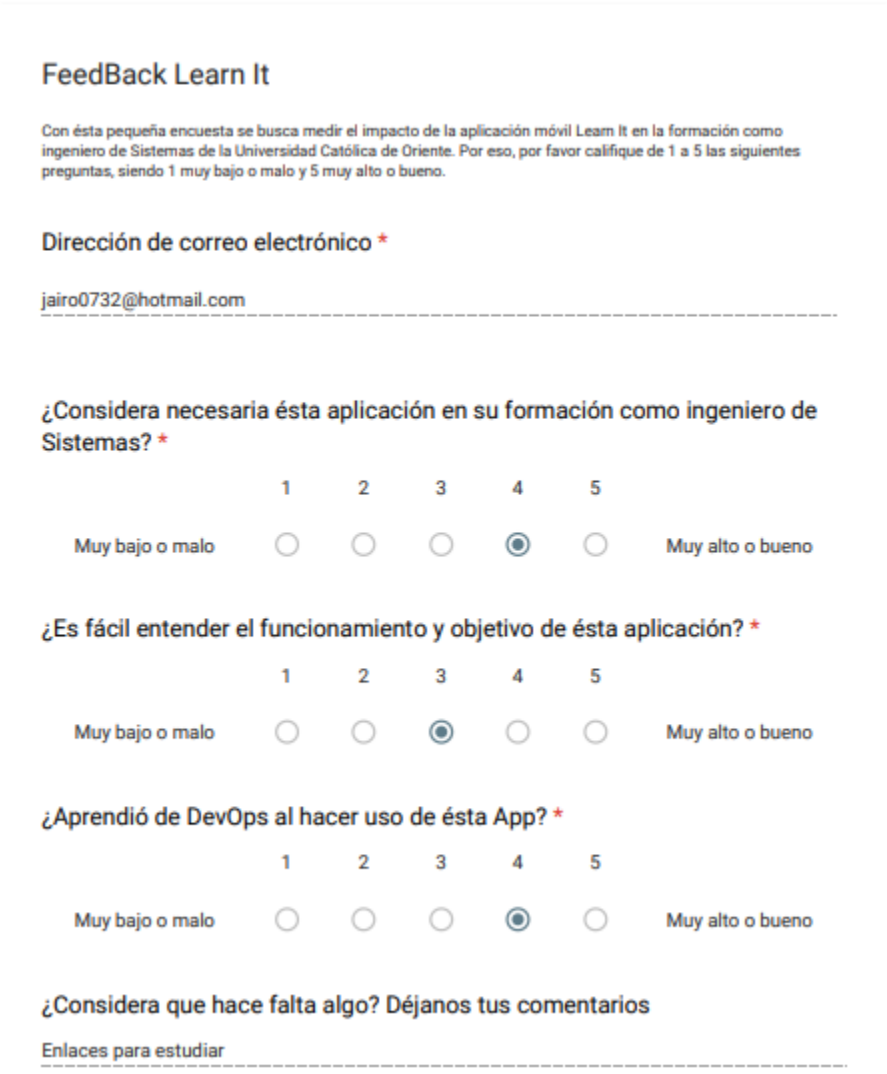


Anexo 2. [Figura. Fuente: Autor].



Anexo 3. [Figura. Fuente: Autor].

Anexo 4. [Figura. Fuente: Autor].



FeedBack Learn It

Con ésta pequeña encuesta se busca medir el impacto de la aplicación móvil Learn It en la formación como ingeniero de Sistemas de la Universidad Católica de Oriente. Por eso, por favor califique de 1 a 5 las siguientes preguntas, siendo 1 muy bajo o malo y 5 muy alto o bueno.

Dirección de correo electrónico *

jairo0732@hotmail.com

¿Considera necesaria ésta aplicación en su formación como ingeniero de Sistemas? *

	1	2	3	4	5	
Muy bajo o malo	○	○	○	◉	○	Muy alto o bueno

¿Es fácil entender el funcionamiento y objetivo de ésta aplicación? *

	1	2	3	4	5	
Muy bajo o malo	○	○	◉	○	○	Muy alto o bueno

¿Aprendió de DevOps al hacer uso de ésta App? *

	1	2	3	4	5	
Muy bajo o malo	○	○	○	◉	○	Muy alto o bueno

¿Considera que hace falta algo? Déjanos tus comentarios

Enlaces para estudiar

https://forms.gle/d6pEsZzpevu1bReXA

Anexo 5. Figura respuestas de una encuesta y enlace a encuesta de impacto [Fuente: Autor].