DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LA NOTIFICACIÓN Y VISUALIZACIÓN DE NOTAS ACADÉMICAS, DE LOS ESTUDIANTES DE LA CORPORACIÓN UNIVERSITARIA DEL HUILA (CORHUILA) - UH NOTAS

Proyecto de Grado

Andrés Felipe Polo Vanegas

Sebastián Estupiñán Vargas

Corporación Universitaria Del Huila "CORHUILA"

Facultad De Ingeniería

Ingeniería De Sistemas

Neiva - Huila

2018

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LA NOTIFICACIÓN Y VISUALIZACIÓN DE NOTAS ACADÉMICAS, DE LOS ESTUDIANTES DE LA CORPORACIÓN UNIVERSITARIA DEL HUILA (CORHUILA) - UH NOTAS

Proyecto de Grado

Andrés Felipe Polo Vanegas

Sebastián Estupiñán Vargas

Director:

Mg. José Miguel Llanos Mosquera

Corporación Universitaria Del Huila "CORHUILA"

Facultad De Ingeniería

Ingeniería De Sistemas

Neiva - Huila

2018

Tabla de contenido

Lista de tablas

Lista de ilustraciones

1. Introducción

Actualmente en el mundo existen más dispositivos móviles que computadoras, la sociedad comúnmente utiliza estos dispositivos para la comunicación, así como para acceder a servicios comerciales, financieros, de entretenimiento, entre otros. En los últimos años el desarrollo de aplicaciones móviles ha aumentado debido a la masificación de los teléfonos móviles y el acceso a servicios de datos, de igual forma se ha facilitado su desarrollo gracias a la oferta de herramientas de software. Además de tener un amplio campo de implementaciones.(Angélica, Briceño, Consuelo, & Salazar, 2016) Por otra parte en Colombia el panorama de fabricantes y sistemas operativos de Smartphone indica que Android tiene el mayor porcentaje de usuarios con un 86,6% frente a iOS con 7,9%, Windows 3.5% y otros con un 2,0% debido a que la mayoría de marcas manejan el sistema operativo Android respecto a iOS que tiene como único fabricante Apple. (Coscaré, 2015)

La adaptación según el cambio tecnológico es inevitable, los dispositivos móviles ayudan en la mejora de procesos existentes estando a la vanguardia con la tecnología, mejorando la calidad de vida de las personas. Es por este motivo que el proyecto va dirigido a los estudiantes de la Corporación Universitaria del Huila "CORHUILA". Una Institución de Educación Superior de carácter universitario constituida por tres sedes (Prado Alto, Quirinal y Pitalito). Según reportes de la oficina de registro y control actualmente acoge a 4.181 estudiantes para el periodo 2018-A, de los cuales existe la posibilidad que la gran mayoría tengan un celular. Así mismo se puede evidenciar dentro de la institución que los estudiantes hacen uso masivo de teléfonos inteligentes. La exigencia de avanzar por parte de la universidad con la tecnología móvil como medio significativo de mejora en los procesos es motivado por el crecimiento y adopción del Smartphone en el mundo y en Colombia.

Los estudiantes acceden a la información académica a través de una aplicación Web, no obstante, en los dispositivos móviles es difícil la visualización y el acceso rápido a la

información, además no existe un mecanismo de entrega oportuna de las calificaciones académicas para los estudiantes. Por esta razón, surge la necesidad de desarrollar una aplicación móvil como solución tecnológica que se adapte a los Smartphone que implementan el sistema operativo Android, con el fin de mejorar los procesos académicos de consulta recurrente por parte de los estudiantes, solucionando la entrega instantánea de información relacionada a las calificaciones y ayudando al avance tecnológico de la universidad.

2. Planteamiento Del Problema

La Corporación Universitaria del Huila "CORHUILA" cuenta con una aplicación Web que permite a los estudiantes verificar sus calificaciones registradas por el personal docente y consultar información académica como horario de clases, matrícula, promedios, semáforo y perfil estudiantil, entre otros. Con el sistema implementado actualmente la comunidad de estudiantes no puede revisar en tiempo real sus calificaciones debido a que no se tiene un método de notificación inmediato. De igual forma se puede observar que la gran mayoría de los estudiantes de CORHUILA utilizan activamente su teléfono inteligente en sus tareas cotidianas; al momento de acceder a la aplicación Web académica de la universidad con un Smartphone se presenta dificultad para visualizar la información de forma clara debido a que el contenido no se adapta al tamaño del dispositivo, así mismo el acceso sólo es posible mediante un navegador Web ocasionando tiempos de carga mayores de la información y afectando la experiencia de usuario de los estudiantes.

Es necesario el desarrollo de una aplicación móvil para teléfonos inteligentes con sistema operativo Android como solución, para que la mayoría de los estudiantes puedan ver sus calificaciones en tiempo real mediante notificaciones y también consultar información académica relacionada a materias, horarios, pensum, promedios y perfil del estudiante.

2.1. Formulación pregunta de la problemática

El problema descrito anteriormente conlleva a formular la siguiente pregunta de investigación:

¿Qué tipo de herramientas de las tecnologías de la información contribuiría a mejorar la visualización y entrega oportuna de las notas académicas de los estudiantes de la Corporación Universitaria del Huila "CORHUILA"?

3. Justificación

Las instituciones de educación superior poco a poco han implementado aplicaciones móviles que mejoran el servicio a los estudiantes, un ejemplo es la universidad de Harvard que cuentan con una aplicación que mejora la experiencia móvil de estudiantes en el campus (Harvard University, 2018), así mismo universidades en todo el mundo y en Colombia se suman a esta iniciativa, además la utilización de teléfonos inteligentes en las universidades es masiva, un estudio sobre el uso de Smartphone en estudiantes y docentes universitarios afirma que, en la universidad de Baja California, México muestra que los estudiantes universitarios que son usuarios de Smartphone alcanzan un 97%. (Patricio Henríquez Ritchie, Javier Organista Sandoval, 2015). Es así como en la Corporación Universitaria del Huila "CORHUILA", la comunidad de estudiantes utiliza activamente su Smartphone en tareas comunes, donde se evidencia la necesidad de una aplicación móvil capaz de entregar la información de forma rápida y oportuna a sus estudiantes, aportando de igual forma al crecimiento tecnológico de la Universidad.

En la actualidad la CORHUILA implementa una aplicación Web que permite a los estudiantes revisar las notas académicas registradas por el personal docente; esta herramienta ha contribuido a que estudiantes estén enterados de los resultados académicos que se registran en el transcurso del semestre, como también la consulta de información académica como horarios, promedios, matrícula y perfil estudiantil, entre otros. Sin embargo, no existe la forma en la que la información de calificaciones llegue en tiempo real y la información académica sea mucho más fácil y rápida de visualizar para los estudiantes.

Por este motivo es necesario el desarrollo de una aplicación móvil con notificaciones en tiempo real, que además contenga información relacionada al histórico de calificaciones obtenidas en el transcurso de la carrera profesional, horario de clases, pensum de la carrera, promedios por semestre, datos generales del perfil académico y una calculadora de notas que

ayuda a los estudiantes a conocer la calificación que deben obtener en el corte final para aprobar un curso.

4. Objetivos

4.1. General

Desarrollar una aplicación móvil para visualizar y notificar en tiempo real las calificaciones académicas de la comunidad estudiantil de la Corporación Universitaria del Huila "CORHUILA" del municipio de Neiva.

4.2. Específicos

- Recolectar información para la especificación de los requerimientos funcionales y no funcionales de la aplicación móvil.
- Completar el diseño de componentes del software.
- Ejecutar el proceso de desarrollo definido para la construcción de la aplicación.
- Realizar las pruebas de funcionamiento.

5. Marco Teórico

5.1. Antecedentes

5.1.1. Importancia y crecimiento de los dispositivos móviles y Apps

Los dispositivos móviles y sus Apps están presentes en la mayoría de las actividades que realizan las personas en el día a día, así como en las instituciones de educación superior la comunidad generalmente esta comprendía por jóvenes que utiliza de forma masiva tecnologías móviles que permiten tener la información al alcance de su mano. En particular se hace referencia a una sociedad exigente y con la necesidad constantes de acceso a la información.(Cantillo Valero, Roura Redondo, & Sánchez Palacín, 2012)

Con la gran demanda y necesidades tecnológicas de la sociedad todo el terreno conveniente al desarrollo de software está tomando mayor importancia, también gracias a los diferentes cambios producidos en los últimos años en la industria musical, la publicidad y la venta. "Smartphone con los diferentes sistemas operativos existentes hoy en día como el iPhone, Android entre otros, han redefinido lo que se entendía como aplicación móvil".(Durall, Gros, Maina, Johnson, & Adams, 2012)

La evolución de estos dispositivos a través del tiempo cada vez es mayor y no parece detenerse, tan solo "en 2013, las ventas mundiales de móviles inteligentes (Smartphone) alcanzaron 968 millones de unidades, lo que supone un aumento del 42,3% respecto al 2012 y la mayor penetración se da en usuarios de edades comprendidas entre 18 y 25 años", dos o tres años atrás podíamos decir que el uso principal de los móviles eran las llamadas, esto ha cambiado y es usado principalmente para conectarse a Internet donde las "actividades fundamentales que tienen los usuarios a través del móvil son, acceder al correo electrónico (82%), la mensajería instantánea, con un 78%, superando por primera vez a la navegación (74%) y la lectura de noticias de actualidad, con un 60%", todas estas actividades se ejecutan entorno a las Apps que los celulares Smartphone integran. Este mismo año la utilización de

Apps por parte de usuarios en el mundo ha crecido un 115% con respecto al año anterior, siendo este aumento impulsado por el desarrollo de nuevas Apps que ayudan a resolver necesidades puntuales en las vidas de las personas. "Por regiones, la mayor penetración de móviles la encontramos en Europa Central y del Este, con una penetración del 151%, seguido del Oeste de Europa con un 129% y América del Sur con 124%. La más baja se encuentra África, que solo cuenta con una penetración del 67%, seguido del Sur de Asía (72%) y Centroamérica (89%). En cuanto al consumo de datos por regiones (indicación del uso del móvil para conectarse a Internet), Norte América es dónde se consumen más megabytes, seguido del oeste de Europa y Europa Central". Las cifras anteriores evidencian un aumento significativo en el uso de dispositivos móviles a nivel mundial en el año 2013 a diferencia del 2012. (Ditrendia, 2014)

Ya para el año 2014 este crecimiento no se ha estancado y por el contrario encontramos que con "7.300 dispositivos, el número de teléfonos móviles supera por primera vez el número de personas en todo el mundo, se vendieron 1.245 millones de Smartphone en el mundo, lo que supuso un aumento del 28,4% respecto al año anterior" y el dominio del Smartphone en los usuarios respecto a otras tecnologías tradicionales como la televisión en "promedio de tiempo diario que los usuarios pasan en el móvil con 177 minutos superando a la T.V que tiene 168 minutos" del tiempo promedio de las personas. A su vez "los consumidores se muestran más dependientes de sus Smartphone. El 40% de las personas mira la pantalla de su teléfono más de 50 veces al día mientras que el 70% mira su teléfono durante la primera media hora después de haberse despertado. Las aplicaciones son la forma favorita por la cual los usuarios se conectan desde los dispositivos, el 90% del tiempo de conexión proviene de del uso de Apps. Cada mes se lanzan 40.0000 nuevas aplicaciones al mercado móvil" las cuales son distribuidas por las principales tiendas de cada sistema operativo móvil dominante tanto como en Google

Play Store de Android como en la App Store de Apple. "Los usuarios de Smartphone prefieren usar aplicaciones a navegar por Internet". (Ditrendia, 2015)

El uso de los dispositivos móviles a nivel mundial entre los años 2015 y 2020 crecerá cerca de 8 veces y a finales de 2015 "la penetración de teléfonos móviles en el mundo ascendió al 97%" sumando a esto que más de la mitad de las visitas que reciben los grandes buscadores proceden ya del móvil y un 62% del tiempo total pasado por los usuarios en el mundo online ya se realiza desde Smartphones y Tablets", esto supone una alza cada vez mayor a años anteriores lo que deja en evidencia el emergente crecimiento y consolidación de esta tecnología en la sociedad, podemos preguntarnos cuando se detendrá. "Un crecimiento que se debe principalmente al auge experimentado en los mercados emergentes de Asia, Pacífico, Japón, Oriente próximo y África. Pero este crecimiento continuado de las ventas de Smartphones tenderá a desacelerarse en los próximos años, dada la madurez de los mercados en Norteamérica, Europa y China". (Ditrendia, 2016)

Con la integración de Chatbots, Smartwatchs, banca móvil y el Internet de las cosas la tendencia de uso de estos dispositivos no decrece y los hábitos de consumo varían tras el pasar del tiempo. En el año 2017, el número de usuarios móviles en el mundo ascendió a 4,9 mil millones que representa el 66% de la población mundial. Además, las ventas de Smartphone siguen creciendo: en total en 2016 se vendieron 1,5 mil millones de unidades en todo el mundo, un 5% más que en 2015. Hoy en día podemos decir que el móvil es parte de nuestras vidas, en un día normal, los usuarios utilizan más el Smartphone que el ordenador tanto así que ha cambiado la forma en que las personas realizan sus actividades. Por primera vez en todos los grupos de edad, el mayor porcentaje de consumidores dice que la primera cosa que hacen en sus teléfonos cada día es comprobar los mensajes instantáneos. La segunda actividad es revisar el correo electrónico. En los últimos 3 años, el tiempo total dedicado al mundo digital ha crecido un 53%, un crecimiento principalmente motivado por el uso de las Apps móviles y un

poco menos por el uso de la Web móvil. En concreto, el uso de las Apps ha aumentado un 111% en los últimos 3 años, siendo un 11% el crecimiento experimentado entre 2015 y 2016. De hecho, el uso de las Apps representa el 60% del tiempo total pasado por los usuarios en el mundo digital. De media, cada usuario en el mundo dedicó 73,8 horas al mes a las aplicaciones móviles. Por edades, son los Millennials los que dedican un tiempo desorbitado a las Apps móviles, (93,5 horas al mes en 2016, un 3% más que en 2015) aunque todos los segmentos de edad dedican cada vez más tiempo a las aplicaciones, siendo los usuarios de entre 55 y 64 años los que más han aumentado su tiempo de dedicación a estas plataformas. (Ditrendia, 2017)

En Colombia un 42% de la población, interfiere con la información consultada desde uno de los diversos dispositivos (Desktop, Smartphone, Tablet). 16.4 millones de personas vinculan sus consultas desde una Desktop, manipulado en su mayoría por personas entre los 6 años de edad en adelante. 13.3 millones de personas desde un Smartphone y 3 millones realizan sus consultas desde una Tablet, donde el manejo de dispositivos móviles es manipulado en gran parte por personas desde los 17 años y más. El anterior análisis parte de datos individuales por uso de dispositivos, pero si consideramos que una persona puede tener uno o más de estos dispositivos, resaltaría una nueva estadística. Es así como la audiencia multiplataforma no puede ser considerada de forma aislada, donde el total de población digital combinada es de 21,6 millones. Se estiman que 9 millones de personas utilizan 2 o más dispositivos, dando origen a que exista 14.2 millones de personas en el manejo de dispositivos móviles. Estas personas están conformadas en su mayor parte por la generación del milenio, de los cuales el 85 por ciento corresponden en el manejo de dispositivos móviles y multiplataforma. (ComScore, Marchant, & Valencia, 2015)

En las instituciones de educación superior el uso masivo de dispositivos móviles y sus Apps no es ajeno y está particularmente relacionada con una población de jóvenes, proyecciones de la universidad Javeriana indican que en general, puede considerarse que la

población de 15 a 29 años de edad es la principal demandante de servicios educativos postsecundarios, por esta razón su comportamiento hasta el año 2025 es un factor importante en la estimación del número de estudiantes en entidades de educación postsecundaria. (Tenjo Galarza, 2012) Así mismo el DANE utiliza rangos de edades similares para realizar sus estudios, mencionado lo siguiente: al analizar las estimaciones y proyecciones de población para el grupo de 17 a 21 años de edad “edades teóricas para el cálculo de las tasas de cobertura en educación superior” (Ministerio de Educación, 2016), donde la población de jóvenes mencionada permite determinar una comunidad estudiantil con hábitos y características especiales conocidos como Millennials. Un artículo define a esta población como "los nacidos entre 1981 y 1995, jóvenes entre 20 y 35 años que se hicieron adultos con el cambio de milenio. Según el reporte de Tendencias Digitales Conecta, actualmente en Latinoamérica un 30 % de la población es Millennials. La demanda de los Millennials está impulsando el extraordinario crecimiento de las aplicaciones móviles en las tiendas de aplicaciones se registran en el mundo 5 millones de descargas de aplicaciones diarias". (Gutiérrez-Rubí, 2014)

A nivel mundial, nacional y regional se encuentra la implementación de aplicaciones móviles para la mejora de los procesos educativos en las universidades.

5.1.2. Apps en instituciones de educación superior

5.1.2.1. Harvard Mobile 2.0

La aplicación Harvard Mobile es una iniciativa de toda la universidad Harvard para mejorar la experiencia móvil de estudiantes, docentes, personal, visitantes y vecinos que interactúan con el campus y la comunidad de Harvard. Lanzado en enero de 2013. Harvard Mobile 2.0 tiene aplicaciones nativas para los sistemas operativos Android e iOS. (Harvard University, 2018)

5.1.2.2. Stanford Mobile

Stanford Mobile es la aplicación móvil oficial de la universidad de Stanford. ofrece acceso a información y servicios esenciales de Stanford en cualquier momento y en cualquier lugar en su dispositivo móvil. Stanford Mobile permite: Buscar en el mapa del campus, Buscar en el directorio del campus, Encontrar restaurantes y horarios, Verificar las rutas y horarios del servicio de autobús, Leer noticias de la universidad, Navegar nuestro calendario de eventos, Explorar cursos, Buscar colecciones de la biblioteca. (Stanford University, 2018)

5.1.2.3. Mobile Oxford

Es la aplicación móvil oficial de la universidad Oxford para los residentes y miembros de la Universidad. Mobile Oxford es una guía fundamental para ayudarle a hacer sus tareas del día a día. Si se trata de encontrar un libro de la biblioteca, revisar el próximo autobús o incluso encontrar a qué hora el buzón más cercano se recoge. (University of Oxford IT Services, 2017)

Al igual que en las universidades a nivel mundial en Colombia existen desarrollos e implementaciones de Apps en las instituciones. Ricardo Sotaquirá, director de la Facultad de Ingeniería Informática de la Universidad de La Sabana y experto en desarrollo de aplicaciones educativas, explica que "la actividad académica tiene muchas tareas de control y de ordenamiento que son claves para el éxito del estudiante, y en ese propósito las Apps pueden ser una mejor vía para llegar a él". Incluso, más que los computadores de escritorio o portátiles, herramientas que muchos aún creen las más adecuadas para consumir contenidos. (Betancur, 2014)

A continuación, algunas Apps de las universidades colombianas que están disponibles en las tiendas de aplicaciones.

5.1.2.4. UPB Colombia

App de la Universidad Pontificia Bolivariana permite acceder a información de la institución y datos del estudiante. Con la posibilidad de acceder al portal de noticias y al

calendario de eventos, ver la ubicación de los edificios de cada sede de la universidad en las distintas ciudades, teléfonos de contactos y acceso a redes sociales. (Universidad Pontificia Bolivariana, 2017)

5.1.2.5. UniSabana APP

La aplicación de la universidad de La Sabana permite consultar noticias y eventos, además de consultar el menú del día, las diferentes opciones de los restaurantes, escuchar Unisabanaradio, los estudiantes conocer el desempeño en las asignaturas que está cursando, ver horario, consultar material bibliográfico y estar atento a las alertas o emergencias que afectan a la comunidad dentro y fuera del campus. (Universidad de La Sabana, 2017)

En cuanto a nivel regional en el departamento del Huila existe la aplicación móvil de la Universidad Surcolombiana de la ciudad de Neiva llamada USCO APP que ofrece noticias, información institucional, agenda y eventos entre otros servicios.

En definitiva, la evidencia de la masificación de los Smartphone a nivel mundial y el crecimiento en el uso de Apps, la implementación de aplicaciones móviles en las instituciones educativas de nivel superior tanto en el mundo como en Colombia y en del departamento del Huila, además de conocer que la gran mayoría de la población universitaria está compuesta por jóvenes catalogados como Millennials, se entiende la necesidad tecnológica que tienen los estudiantes de la Corporación Universitaria del Huila "CORHUILA".

5.2. Referentes conceptuales

5.2.1. Desarrollo de aplicaciones

En el mundo actualmente existen tres plataformas móviles líderes iOS, Android OS, Windows 10 Mobile. (Richter, 2017) Para el año 2016 a nivel mundial Android representaba la cuota más alta del mercado con un 84.8% frente a iOS con 14.4% y Windows menos del 2%. (Gartner, 2017)

Actualmente en el desarrollo de aplicaciones móviles cada plataforma ofrece su propio SDK, lenguajes de programación y herramientas para el desarrollo de Apps, para poder cubrir el total de la cuota del mercado de sistema operativo móviles, los desarrolladores se ven en la difícil tarea de pasar por la curva de aprendizaje en cada plataforma móvil es decir aprender un lenguaje y herramientas distintas para el desarrollo de Apps tanto para iOS como para Android. Es complejo y costoso para una persona pasar por ese aprendizaje, así como para una empresa de mediano tamaño tener dos equipos de desarrollo por cada plataforma específica. Entre las dificultades más significativas al momento de desarrollar una aplicación móvil por cada plataforma encontramos el tiempo de aprendizaje, versiones distintas de la App y recurso humano especialista que influyen directamente en los costos de desarrollo en cualquier aplicación.

Las tecnologías de desarrollo Cross-Platform o Multi-Plataforma nativa e híbrida facilitan la construcción de aplicaciones móviles que se ejecuten en distintos sistemas operativos principalmente porque se desarrolla una sola aplicación de distribución para las diferentes tiendas de Apps (App Store, Play Store y Windows Marketplace). Así mismo cada uno de las tecnologías de desarrollo móvil multiplataforma ofrecen lenguajes y herramientas diferentes, es responsabilidad del desarrollador o equipo de desarrollo escoger la tecnología que mejor se adapte a sus conocimientos y solución móvil. Además, se deben tener en cuenta pros y contras que permitan hacer más segura la elección.

5.2.2. Plataformas móviles

5.2.2.1. iOS - Apple

En 2007 Apple lanzó el iPhone con iOS como su sistema operativo generando un importante cambio en la manera cómo los usuarios interactúan con las aplicaciones este nuevo dispositivo integró la pantalla táctil, agregó la conexión a Internet en cualquier lugar por medio de redes celulares y más importante una tienda de aplicaciones para la distribución de Apps a

usuarios finales que posteriormente empresas de software adoptaron como modelo de mercadeo de aplicaciones. (Mobile Marketing Association, 2011)

Es muy importante para un desarrollador tener en cuenta las nuevas versiones que con el transcurso del tiempo se han lanzado porque dentro de esta plataforma existe una segmentación de usuarios de acuerdo a las versiones que Apple presenta de su dispositivo móvil y su sistema operativo. Hoy día el iPhone está en su versión 7 y iOS en la 10.(Apple, 2017b)

Para la creación de Apps iOS, Apple proporciona un entorno de desarrollo basado principalmente en:

- XCode, Entorno de desarrollo integrado por sus siglas en inglés IDE.
- Un simulador de dispositivos móviles.
- Objective-C o Swift, lenguajes de programación.
- Un Kit de desarrollo de software SDK para iOS.

El SDK contiene herramientas necesarias para el desarrollo, instalación, ejecución y prueba de aplicaciones nativas. (Developer, 2017) Con estos componentes se puede crear Apps para iPhone, Mac, e iPad.(Inc, 2017)

Hay que aclarar que, para desarrollar Apps para esta plataforma, es necesaria una computadora Mac con su respectivo sistema operativo Mac OS.

5.2.2.2. Android OS

Google dio origen al sistema operativo Android OS que logró posicionarse como principal competidor de Apple con la ventaja de ser un sistema libre y de código abierto dando la posibilidad a cualquier fabricante de dispositivos inteligentes hacer uso del mismo. (Mobile Marketing Association, 2011)

Google nos ofrece un entorno de desarrollo para la creación de Apps en Android OS, compuesto por:

- Android Studio como IDE.

- Un emulador de dispositivos móviles Android Virtual Devices - AVD.
- Java, Kotlin o C++ como lenguajes de programación.
- Un sdk-tools (proporcional al lenguaje seleccionado).

A diferencia de Apple, Google permite tener el entorno de desarrollo tanto en PC Windows como en Mac e incluso en computadoras con sistemas basados en Linux.(Android, 2017)

5.2.2.3. Windows 10 Mobile

Microsoft intentó en muchas ocasiones llegar a rivalizar el dominio en el mundo de los dispositivos móviles, inicialmente su enfoque empresarial dio origen al sistema operativo Windows Mobile que integraba aplicaciones del ya conocido Windows para Desktop, para el año 2008 presentaron Windows Phone un sistema pensado para el consumo personal pero por problemas de fragmentación en 2015 anunció que terminaría con el proyecto, dando cabida a Windows 10 Mobile un sistema operativo capaz de ejecutarse en diferentes dispositivos (Smartphone, Tablets, Computadoras) así mismo proporcionó a los desarrolladores un SDK para construir aplicaciones y una plataforma de desarrollo flexible llamada Windows Universal Platform (UWP) con soporte en Visual Studio 2017 además UWP permite programar Apps utilizando: C# Y XAML (Microsoft, 2017)

5.2.3. Tecnología Multiplataforma Xamarin

En Xamarin con un único lenguaje de programación (C#) y una Biblioteca de Clases Portable "PCL" se puede construir una aplicación móvil que se ejecute de forma nativa en distintas plataformas como iOS, Android o Windows Phone. (Microsoft Xamarin Docs, 2017) Sabemos que cada plataforma móvil tiene características diferentes para desarrollar aplicaciones nativas. Así mismo otras plataformas permiten crear aplicaciones hibridas con lenguajes como HTML y JavaScript que pueden no utilizar el Kit de Herramientas Nativo. (Microsoft Xamarin Docs, 2017)

Xamarin provee acceso completo a las APIs de las plataformas nativas, además de tener interoperabilidad con bibliotecas escritas en el lenguaje nativo de cada sistema operativo dando la posibilidad de utilizar código de terceros. Así como un lenguaje moderno C Sharp que es utilizado para la codificación por completo de toda la aplicación, en conjunto con la Bibliotecas de Clases Portables son pilares fundamentales en Xamarin. Las BCL son una colección de clases que integran código asociado a la lógica de negocio, acceso a datos, servicios y más. Que pueden ser compartidos entre proyectos. Todo esto en el marco de un entorno de desarrollo integrado como Visual Studio. En consecuencia, se puede reducir significativamente los costos de desarrollo y el tiempo de comercialización para los desarrolladores móviles que se dirigen a las tres plataformas móviles más populares. (Microsoft Xamarin Docs, 2017)

Xamarin divide el desarrollo orientado a plataformas especificas en Xamarin.iOS y Xamarin.Android. Ambos se basan en el proyecto Mono que existe desde los tiempos del .NET Framework y puede ejecutarse en prácticamente todas las plataformas, incluido Linux, Unix, FreeBSD y Mac OS X. En iOS, el compilador Ahead Of Time (AOT) de Xamarin compila aplicaciones de Xamarin.iOS directamente en código de ensamblado nativo de ARM. En Android, el compilador de Xamarin compila en lenguaje intermedio (IL), que es Just-In-Time (JIT) compilado en ensamblado nativo cuando se inicia la aplicación. En ambos casos, las aplicaciones de Xamarin usan un tiempo de ejecución propio de cada plataforma. (Microsoft Xamarin Docs, 2017)

Cuando se compilan aplicaciones de Xamarin, el resultado es un paquete de aplicación, ya sea un archivo .app en iOS o un archivo .apk en Android. Estos archivos no se distinguen de los paquetes de aplicación compilados con los IDE predeterminados de la plataforma y se implementan de la misma manera. (Microsoft Xamarin Docs, 2017)

5.2.4. Arquitectura de una aplicación móvil Xamarin

El desarrollo de aplicaciones móviles multiplataforma con Xamarin tiene como estrategia principal compartir código entre plataformas usando C# como único lenguaje de programación, con el objetivo de aumentar la productividad y disminuir el esfuerzo para la creación de soluciones móviles que se ejecute en múltiples sistemas operativos (iOS, Android o Windows). La importancia de definir una arquitectura y patrones de diseño que sigan los principios de encapsulamiento, separación de responsabilidades y polimorfismo de la Programación Orientada a Objetos es clave para hacer posible la creación de aplicaciones multiplataforma. (Microsoft, 2018)

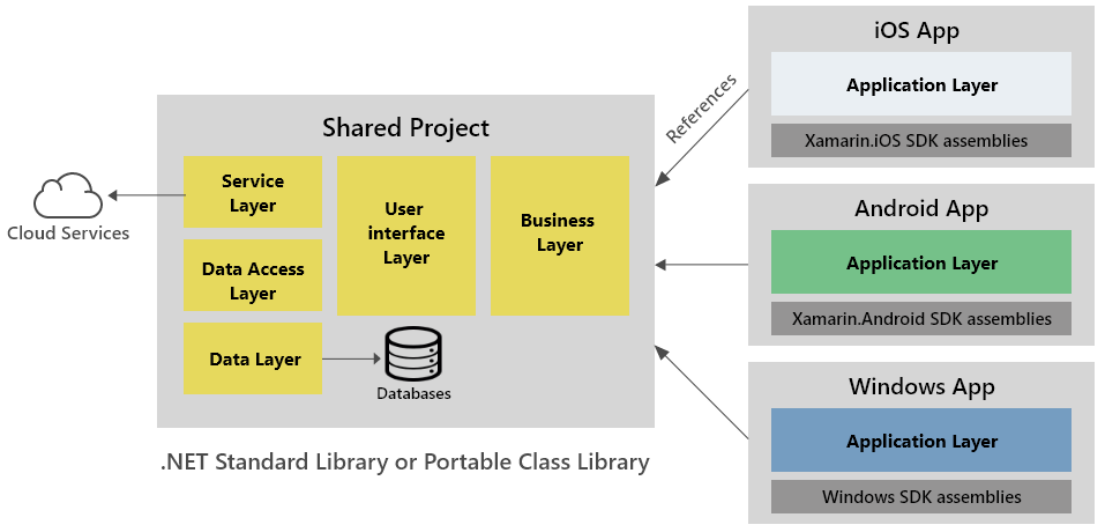
Una solución móvil implementada según el planteamiento anterior se compone de capas lógicas separadas que tengan un propósito claro y bien definido, obteniendo aplicaciones fáciles de comprender, probar y mantener.

En una aplicación estándar podemos encontrar las siguientes capas:

- Capa Interfaz gráfica: Contiene pantallas y controles que se muestran al usuario.
- Capa Datos: Compone la persistencia de los datos que puede mantenerse en una base de datos SQLite.
- Capa Acceso a Datos: Proporciona acceso de creación, lectura, actualización y eliminación (CRUD) a los datos sin exponer detalles de implementación.
- Capa Negocio: Contiene la lógica de negocio.
- Capa Servicios: Proporciona acceso a servicios en la nube, servicios Web complejos como REST.(Microsoft, 2018)

Además de separar las responsabilidades por capas en una aplicación móvil los patrones de diseño ayudan en la creación de aplicaciones móviles fáciles de entender y mantener.

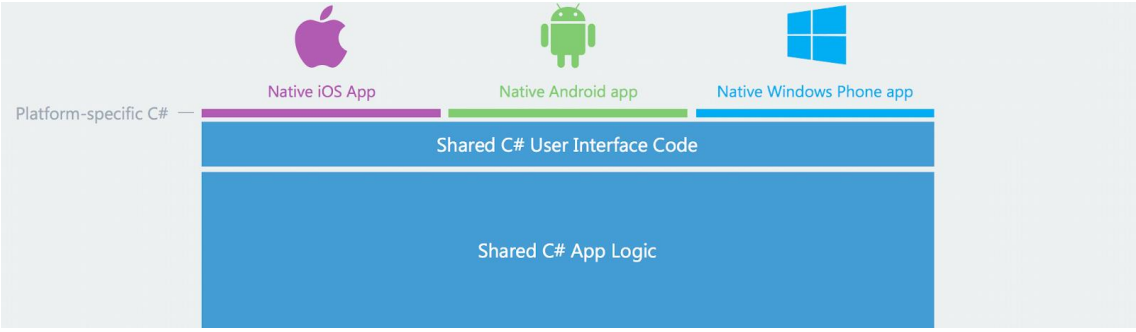
El diseño de una aplicación con responsabilidades bien definidas por capas y la implementación de patrones acordes a problemas recurrentes en soluciones móviles permite a los desarrolladores trabajar simultáneamente en diferentes partes de la aplicación sin presentar problemas al unificar el trabajo realizado. Como ejemplo podemos tomar dos roles en un equipo de desarrollo, diseñador de interfaces y programador cada uno puede enfocarse en trabajar en una funcionalidad en particular sin estropear el trabajo del otro, siendo esto posible gracias a la separación clara entre interfaz de usuario y la lógica de negocios de la aplicación.



Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 1. Arquitectura de una aplicación Xamarin Forms

Xamarin ofrece dos enfoques para la construcción de aplicaciones móviles, primeramente, el enfoque tradicional con Xamarin.android y Xamarin.iOS tienen acceso a todas las APIs de la plataforma en específico y se comparte hasta el 75% del código de negocio, el código de interfaz de usuario (UI) es diferente por cada plataforma. El segundo enfoque, Xamarin.Forms permite compartir hasta el 100% del código de interfaz de usuario y lógica de negocio entre plataformas. (Xamarin, 2018)



Recuperado de: *https://www.xamarin.com/platform*

Ilustración 2. Plataforma de desarrollo Xamarin

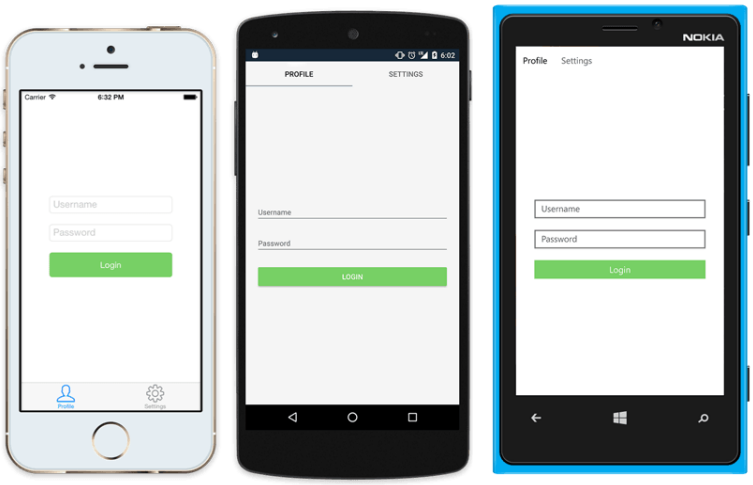
En Xamarin.forms es posible escribir código asociado a la interfaz de usuario UI utilizando XAML o CSharp. En el proceso de ejecución cada objeto de la UI se muestra como un control nativo de la plataforma específica. Las siguientes imágenes muestra un ejemplo.

C # XAML

```
<?xml version="1.0" encoding="UTF-8"?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="MyApp.MainPage">
    <TabbedPage.Children>
        <ContentPage Title="Profile" Icon="Profile.png">
            <StackLayout Spacing="20" Padding="20"
                         VerticalOptions="Center">
                <Entry Placeholder="Username"
                       Text="{Binding Username}"/>
                <Entry Placeholder="Password"
                       Text="{Binding Password}"
                       IsPassword="true"/>
                <Button Text="Login" TextColor="White"
                        BackgroundColor="#77D065"
                        Command="{Binding LoginCommand}"/>
            </StackLayout>
        </ContentPage>
        <ContentPage Title="Settings" Icon="Settings.png">
            <!-- Settings -->
        </ContentPage>
    </TabbedPage.Children>
</TabbedPage>
```

Recuperado de: *https://www.xamarin.com/forms*

Ilustración 3. Código XAML que define la interfaz de usuario para todas las plataformas



Recuperado de: *https://www.xamarin.com/forms*

Ilustración 4. Aplicación móvil en diferentes plataformas iOS, Android y Windows

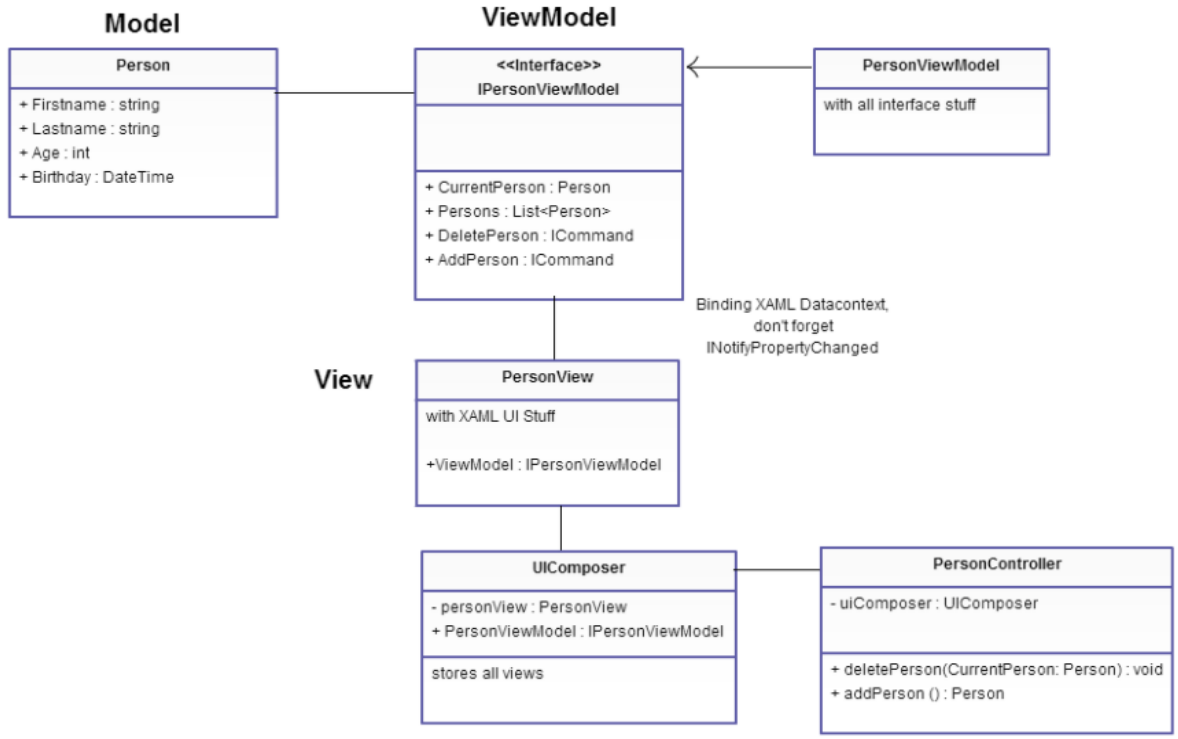
En definitiva, Xamarin como plataforma de desarrollo provee enfoques con una arquitectura bien definida para compartir código entre plataformas (iOS, Android, Windows, etc.) con la implementación de los principios de Programación Orientada a Objetos y patrones de diseño que permite a los desarrolladores crear aplicaciones fáciles de entender, mantener y probar, ayudando a aumentar la productividad y reducir los costos de producción.

5.2.5. Model Vista VistaModelo (MVVM)

Es un patrón arquitectónico que ayuda a separar la lógica de negocios y la capa presentación o interfaz de usuario "UI" de una aplicación permitiendo hacer más fácil de probar, mantener y desarrollar. Una gran ventaja de implementar el patrón MVVM es reutilización de código y permitir a los desarrolladores y diseñadores de interfaz de usuario trabajar en sus respectivas partes de una aplicación al mismo tiempo.

Así mismo es importante comprender las responsabilidades de cada componente y cómo interactúan entre sí. Brevemente podemos decir que la Vista conoce el Modelo de Vista y el Modelo de Vista conoce el Modelo, pero el Modelo no es consciente del Modelo de vista y el Modelo de Vista no es consciente de la Vista. Por lo tanto, el Modelo de Vista aísla la Vista

del Modelo y permite que al Modelo evolucionar independientemente de la Vista. (Xamarin MVVM, 2017)



Recuperado de: *https://creately.com/diagram/example/hepl76ynl/MVVM*

Ilustración 5. Modelo – Vista – Vista – Modelo (MVVM)

Es importante comprender la separación de responsabilidades e interacción entre las Clases que representan cada componente para usar de forma correcta MVVM (Xamarin MVVM, 2017)

5.2.5.1. View

La Vista es responsable del diseño y la apariencia de la aplicación en la pantalla del dispositivo. Es recomendable definir la interfaz del usuario "UI" en XAML y no en C#. En una aplicación de Xamarin.Forms, una vista es normalmente un Page que contiene todos los elementos de UI .(Xamarin MVVM, 2017)

5.2.5.2. ViewModel

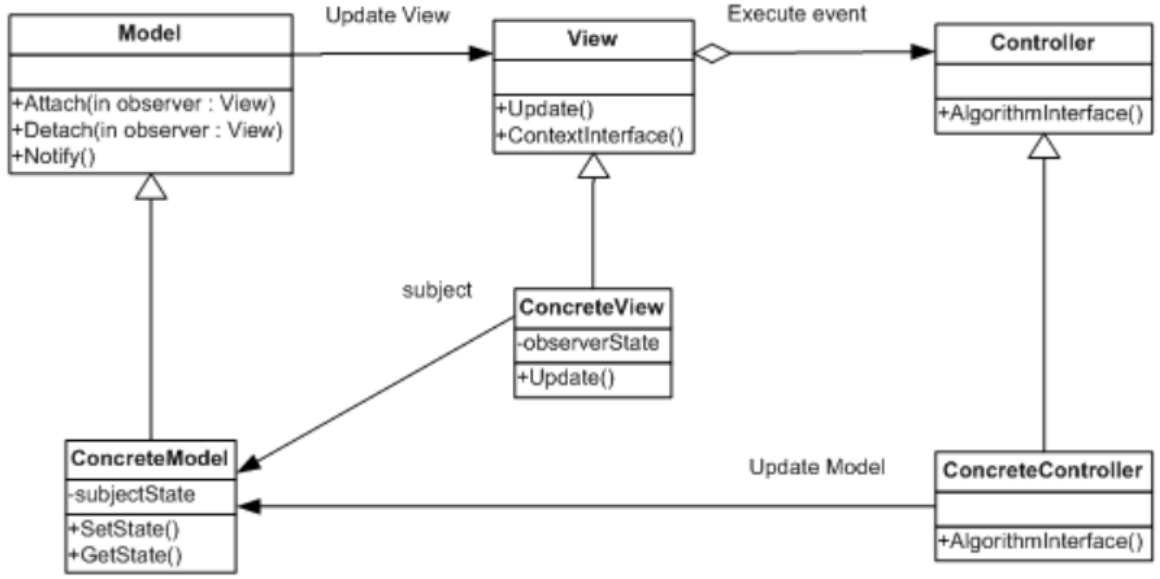
El Modelo de Vista implementa propiedades y comandos que definen la funcionalidad de la interfaz de usuario en la Vista, por medio de Data Bindings se enlazan estas dos clases Modelo de Vista y Vista donde el Modelo de Vista notifica a la Vista sobre nuevos cambios en los datos, Así mismo la Vista determina cómo se mostraran los datos y que acciones puede ejecutar. El Modelo de Vista también es responsable de coordinar las interacciones de la Vista con las clases del Modelo. Normalmente hay una relación de uno a varios entre el Modelo de Vista y las clases del modelo. El Modelo de Vista puede optar por exponer solo algunos atributos del Modelo.(Xamarin MVVM, 2017)

5.2.5.3. Modelo

Las Clases de modelo son representaciones de los datos de la aplicación. Los cuales se utilizan generalmente en las capas de servicios o repositorios que encapsulan el acceso a datos y almacenamiento en caché. De igual forma el Modelo se enlaza con el Modelo de Vista por medio de propiedades (Xamarin MVVM, 2017)

5.2.6. Modelo Vista Controlador (MVC)

MVC es un patrón de arquitectura que es utilizado generalmente es aplicaciones Web, Siguiendo los principios de diseño de software el objetivo del patrón es la separación de lógica de negocio de la aplicación y su presentación. Donde una aplicación se divide en tres partes interconectadas(Kılıçdağı & YILMAZ, 2014)



Recuperado de:
https://www.codeproject.com/KB/architecture/MVC_MVP_MVVM_design/MVC_MVP_MVVM_figure3.gif

Ilustración 6. Modelo Vista Controlador (MVC)

Uno de las grandes ventas de implementar el patrón es que permite reciclar la lógica de la aplicación para diferentes presentaciones. De esta forma la aplicación puede comportase como una API y al mismo tiempo como un sitio disponible en la Web.(Kılıçdağı & YILMAZ, 2014)

5.2.6.1. Modelo

El Modelo es una parte del patrón de diseño Modelo-Vista-Controlador que simplemente maneja la gestión de los datos. El Modelo es angostico en sí mismo a la fuente que provee de datos de la aplicación. (Kılıçdağı & YILMAZ, 2014)

5.2.6.2. Vista

La Vista contiene características de lógica de presentación y plantillas de interfaz de usuario, en si la vista formatea los datos para mostrar al usuario una interfaz que sea fácil de entender y manejas. También hay que mencionar que una Vista puede mostrar los datos en diferentes formatos como JSON o HTML dependiendo del tipo de respuesta que el usuario solicite a la aplicación .(Kılıçdağı & YILMAZ, 2014)

5.2.6.3. Controlador

En el controlador reside gran parte de la lógica de nuestra aplicación. Entiende las solicitudes y luego devuelve los datos a la Vista correspondiente. Además, Actúa como intermediario entre el Modelo y la Vista y otros componentes de la aplicación. (Kılıçdağı & YILMAZ, 2014)

5.2.7. Ciclo de vida de desarrollo de software móvil

La creación de aplicaciones móviles puede ser fácil o complicado dependiendo de la eficiencia en la ejecución de los procesos asociados a cada fase del ciclo de vida desde el diseño hasta las pruebas de facilidad de uso y control de calidad en miles de dispositivos, un ciclo de vida completo debe llegar hasta versiones beta he implementación del producto. (Microsoft Xamarin, 2016)

En especial para el desarrollo de aplicaciones móviles hay una serie de consideraciones que se deben tener en cuenta particularmente si se comparan con las aplicaciones de escritorio o Web tradicionales donde normalmente hay 5 fases importantes del proceso.(Microsoft Xamarin, 2016)

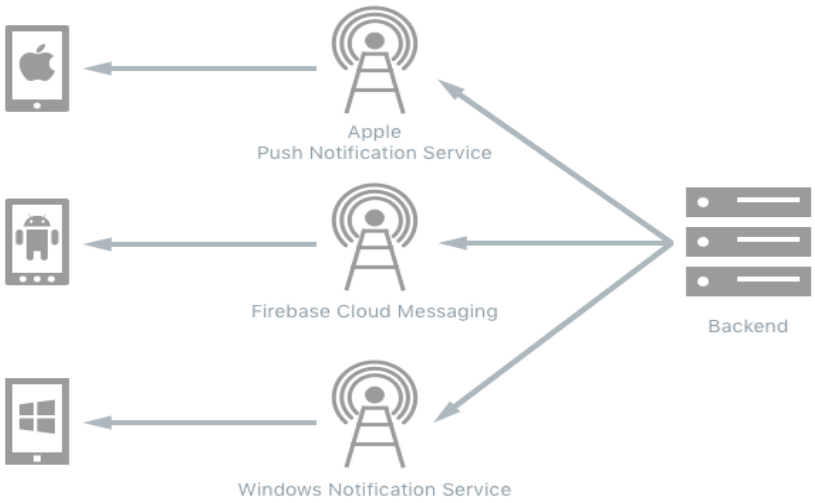
- Análisis: generalmente se inicia con una idea y se perfecciona en una base sólida para una aplicación.
- Diseño: definir la experiencia del usuario (UX), así como convertir esa experiencia del usuario en un diseño de interfaz de usuario (UI) adecuado, normalmente con la ayuda de un diseñador gráfico.
- Desarrollo: se realiza la creación real de la aplicación mediante su programación.
- Estabilización: Ejecución del control de calidad empieza por probar la aplicación y se corrigen los errores.
- Implementación: publicación en las tiendas de Apps.

En ocasiones, se realizan al mismo tiempo tareas relacionadas a fases diferentes dentro del ciclo de vida del software, es así como una aplicación puede estar en una fase de estabilización y al mismo tiempo que se agregan nuevas características a una nueva versión. De igual forma, estas fases se pueden usar con varias metodologías de SDLC como Agile, Spiral, Waterfall, etc.(Microsoft Xamarin, 2016)

5.2.8. Notificaciones Push

Las notificaciones permiten comunicar mensajes pequeños a los usuarios mediante un canal activo en las aplicaciones. Las notificaciones Push en términos generales son una forma de alertar a los usuarios acerca de información en aplicaciones y servicios. En ocasiones los usuarios no recuerdan aplicaciones instaladas, por medio de notificaciones las aplicaciones pueden brindar información oportuna y relevante manteniendo un activa y constante interacción con el usuario. (OneSignal, 2018)

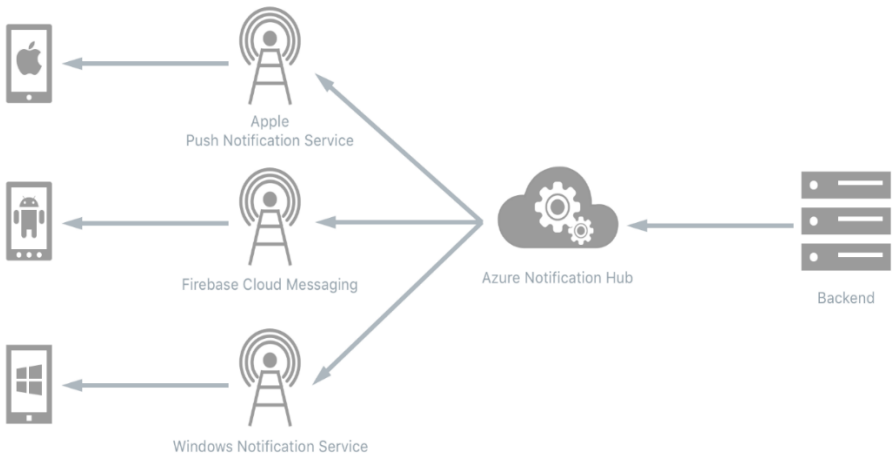
Él envió de notificaciones automáticas a los dispositivos móviles se realiza por medio del Sistema de Notificaciones de Plataforma (PNS), por lo general cada sistema operativo tiene un propio PNS aun así existen Plataformas de notificaciones que soportan varios sistemas, la siguiente imagen ilustra lo descrito:



Recuperado de: *https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/push-notifications*

Ilustración 7. Arquitectura envió de notificaciones en diferentes plataformas

Un problema al implementar notificaciones que soporten varias plataformas es la complejidad en el sistema para gestionar él envió de notificaciones para cada PNS. Los Centros de Notificación eliminan esa complicación al abstraer los detalles de los diferentes Sistemas de Notificación de la Plataforma permitiendo enviar una notificación multiplataforma con una sola llamada API.



Recuperado de: *https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/push-notifications/*

Ilustración 8. Arquitectura envió de notificaciones de un Notification Hub

El Centro de Notificaciones gestiona el contacto con el PNS específico de la plataforma para enviar notificaciones a una instancia de aplicación del cliente. Esto disminuye significativamente la complejidad del Back-End porque se utiliza solo una API para comunicar los mensajes para cada una de las plataformas.

6. Marco Conceptual

App: Una aplicación móvil consiste en un software que funciona en un dispositivo móvil (teléfonos y tabletas) y ejecuta ciertas tareas para el usuario. (Mobile Marketing Association, 2011) Se generará como resultado del proyecto el cual tiene como nombre UH Notas.

Smartphone: Dispositivo móvil que ofrece capacidades más avanzadas que un teléfono móvil común como la conexión a Internet, capacidades multimedia, entre otros.(Mobile Marketing Association, 2011) Esta clase de dispositivo permitirá la ejecución de la aplicación móvil "App".

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación u otra clase de software, que puedan funcionar en diversas plataformas. Existen dos plataformas móviles principales, la familia Apple de iPhones e iPads con el sistema operativo iOS, y el sistema operativo Android que se ejecuta en una variedad de teléfonos y tabletas.(Android, 2017) Un proveedor de software que desea enfocarse en estas tres plataformas debe lidiar con diferentes paradigmas de interfaz de usuario, tres entornos de desarrollo diferentes, tres interfaces de programación diferentes y quizás lo más extraño, tres lenguajes de programación diferentes: Objective-C para iPhone y iPad, Java para Android y C # para Windows. (Apple, 2017a) Es así como se planifica la aplicación móvil UH Notas bajo este mismo sistema donde crearemos la app para Android dando libertad para la misma implementación en iOS o Windows Phone, conseguido por medio de la plataforma tecnológica Xamarin.

API: Interfaz de programación de aplicaciones app. Conjunto de funciones y procedimientos que ofrece ciertas librerías para ser utilizadas por otro software e interactuar con él. (Mobile Marketing Association, 2011) Es así como se crea la implantación de UH Notas API.

Modelo Vista Vista-Modelo (MVVM): El modelo Model-View-ViewModel (MVVM) ayuda a separar la lógica de negocios y la presentación de una aplicación desde la interfaz de usuario (UI) de forma clara. Mantener una separación clara entre la lógica de la aplicación y la interfaz de usuario ayuda a solucionar problemas de desarrollo numerosas y puede hacer que una aplicación sea más fácil probar, mantener y desarrollar. Se pueden mejorar enormemente oportunidades de reutilización de código y permite a los desarrolladores y diseñadores de interfaz de usuario más puedan colaborar fácilmente al desarrollar sus respectivos partes de una aplicación. (Xamarin MVVM, 2018)

Android: Sistema operativo de código abierto para dispositivos móviles basados en Linux propiedad de Google. (Libro Blanco Apps)

Xamarin: Es la plataforma de desarrollo de aplicaciones móviles multiplataforma que proporciona la capacidad de compartir código en todas las plataformas (iOS, Android y Windows) y ayuda a reducir el costo total del producto. Las aplicaciones pueden compartir código de interfaz de usuario y lógica de negocio, conservando el aspecto y la sensación de la plataforma nativa. (Xamarin, 2016)

Código abierto: Conocido también como Open Source, describe software desarrollando y distribuido libremente. (Libro Blanco Apps). En relación a la plataforma Xamarin escogida para crear la aplicación móvil UH Notas permitiendo crear nuevo contenido sin restricciones.

Notificación Push: Notificación automática desde un servidor a una aplicación cliente informando de algún tipo de información nueva disponible. (Libro Blanco Apps) Implementado para la gestión de notificaciones relacionadas a notas académicas dentro de la aplicación UH Notas.

7. Metodología del proyecto

7.1.Metodología de investigación

Se propone una investigación exploratoria encargada de buscar o examinar el problema generado para no equivocarse en la solución del problema, tiene en cuenta algunos estudios realizados que puedan aparecer en el trascurso de la investigación para así enriquecer y respaldar por medio de las relaciones de conceptos o variables la importancia de cara al problema relacionado. (Cazau, 2006) El cual busca analizar y aprender sobre los dispositivos y aplicaciones móviles, plataformas de desarrollo, entre otros y así en conjunto a lo planificado poder ver como a través de la historia se ha generado conocimiento de importancia para dar solución relacionada al problema planteado, que busca por medio de una aplicación móvil mejorar o solucionar un problema existente dentro de la Corporación Universidad del Huila "CORHUILA".

Teniendo en cuenta las diferentes cuestiones, conceptos o variables conseguidos por medio de la investigación exploratoria, se implementa de igual manera un estudio o investigación descriptiva, el cual permite medir o describir las variables o conceptos correspondientemente de cada una de ellas independientemente de las demás, especificando propiedades importantes de personas, grupos o algún fenómeno en común. (Cazau, 2006) Esto nos lleva a evidenciar la cantidad de dispositivos móviles existentes y su uso teniendo en cuenta un amplio detalle en cuanto a dispositivos, sistemas operativos, etc. Una gran ventaja para la creación de la aplicación móvil, teniendo en cuenta los conceptos aprendidos para el desarrollo de la App, esto es algo satisfactorio ya que se pudo analizar las ventajas de la tecnología a utilizar en relación a la población planteada según su comportamiento, siendo escrupulosos con el fin de la solución a implementar ya que conoceremos la cantidad de personas beneficiadas teniendo datos específicos descritos y reales.

7.2.Metodología de desarrollo

"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." (Avison & Fitzgerald, 2006) Sin embargo, también podemos hablar de metodología de forma filosófica, algo que va más allá de los fases o técnicas a seguir. Es así como las metodologías tienen diferentes fases, el contenido de la fase o en su base filosófica, todo esto se aplica, dependiendo del contexto de desarrollo, tamaño del proyecto o del equipo de trabajo, cultura organizacional, entre otros aspectos. (Amaya Balaguera, 2013)

7.2.1. Scrum

Con la masificación de los teléfonos inteligentes ha incrementado el desarrollo de aplicaciones con características distintas a otros tipos de proyectos. Comúnmente las metodologías agiles se implementaron en proyectos Web o de escritorio. No obstante, se han adaptado bien al desarrollo de soluciones móviles, en particular el marco ágil Scrum. Es así como en este proyecto optamos por esta metodología donde prima la comunicación entre los interesados del proyecto. (Leiva Mundaca & Villalobos Abarca, 2015)

Scrum se implementa principalmente en proyectos de corto tiempo donde el entorno de la aplicación es cambiante y se desea mantener la cálida. En el escenario del desarrollo del producto se busca generar continuamente versiones funcionales mediante incrementos conocidos como "Sprints" donde el equipo "Scrum Team" está caracterizado por ser autoorganizado y multifuncional, se conforma por un dueño de producto "Product Owner", el equipo de desarrollo "Development Team" y un Scrum Master. Inicialmente se gestiona una

lista de producto "Product Backlog" donde se definen las funcionalidades a desarrollar del producto. Los Sprint son planeados y generalmente tienen una duración de 10 a 20 días en los cuales se realizan tareas asociadas al desarrollo del producto, así mismo el objetivo es entregar al dueño de producto nuevas funcionalidades y hacer una retroalimentación entre todos los interesados del proyecto. (Guía & Scrum, 2016)

7.2.1.1. Sprints

En Scrum durante los Sprints existen cuatro eventos formales para la inspección y adaptación. (Guía & Scrum, 2016)

- Planificación del Sprint "Sprint Planning"
- Scrum Diario "Daily Scrum"
- Revision del Sprint "Sprint Review"
- Retrospectiva del Sprint "Sprint Retrospective"

7.2.1.2. Roles

En el flujo de trabajo con Scrum interactúan los siguientes perfiles:

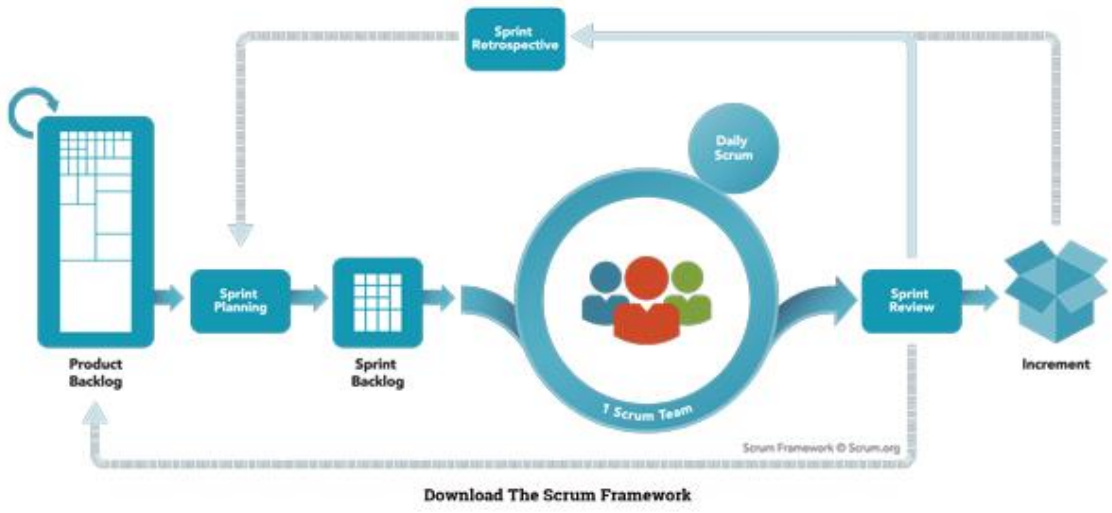
- El Dueño de Producto es el responsable de maximizar el valor del producto y el trabajo del Equipo de Desarrollo. Además, es la única persona responsable de gestionar la Lista del Producto (Guía & Scrum, 2016)
- El Equipo de Desarrollo está integrado por profesionales que realizan el trabajo de entregar un incremento del producto en cada Sprint. (Guía & Scrum, 2016)
- El Scrum Master es responsable de gestionar y manejar todos los procesos en Scrum de forma correcta de acuerdo a la teoría, prácticas y reglas. Así mismo es un líder que está al servicio del Equipo e interviene las veces que sea necesarias para ayudar en la creación del producto. (Guía & Scrum, 2016)

7.2.1.3. Artefactos

En Scrum los artefactos ayudan a maximizar la transparencia de la información entre todos los interesados del proyecto para la inspección y adaptación de los procesos. Es así como se diseñan los siguientes artefactos:

- Lista de Producto "Product Backlog" es una lista ordenada de las características y funcionalidades que debe cumplir el producto. El Dueño de Producto el responsable mantener esta lista. (Guía & Scrum, 2016)
- Lista de Pendientes del Sprint "Sprint Backlog" son una serie de ítems de la Lista de Producto escogidos por el Equipo de Desarrollo para ser ejecutados en un Sprint, que al finalizar son entregados como un incremento del producto. (Guía & Scrum, 2016)

7.2.1.4. Flujo de trabajo



Recuperado de: *https://www.scrum.org/resources/blog/que-es-scrum*

Ilustración 9. Flujo de trabajo con Scrum

8. Desarrollo

8.1.Análisis

8.1.1. Generalidades tecnológicas

De manera general se describen aspectos tecnológicos del proyecto. El sistema se compone de una aplicación móvil que es desarrollada con la plataforma Xamarin para el sistema operativo Android y una Api Rest construida con el Framework Laravel de Php que permite la comunicación entre aplicación móvil y base de datos Oracle del sistema académico de la institución. Además, se integra MySQL como repositorio auxiliar de la información que consume la App.

8.1.2. Planeación

Inicialmente se determinaron las fases de desarrollo del proyecto con cada una de sus tareas asociadas y los días que tomara llevar a cabo dicha tarea. La planeación en esta etapa se realiza en reunión con el equipo de desarrollo y Scrum Master siguiendo el proceso de desarrollo iterativo e incremental que propone Scrum.

Tabla 1. Planeación de la fase de desarrollo

Proyecto UH - Notas	**134 días**
Análisis	**28 días**
Información del proyecto	5 días
Especificación de requisitos	5 días
Priorización de requisitos	2 días
Capacitación	15 días
Validación de requisitos	1 día
Diseño	**31 días**
Prototipo funcional	10 días
Arquitectura del software	5 días
Diagrama entidad-relación	4 días
Capacitación	10 días
Diagrama de clases	2 días
Desarrollo	**51 días**
Configuración de ambientes	5 días
Implementación DB	1 día
Codificación API	20 días
Codificación App	25 días

Pruebas	**17 días**
Características	5 días
Interfaz de usuario	12 días

Nota fuente: *Polo, A y Estupiñán, S (2018)*

8.1.3. Requerimientos funcionales y no funcionales

En una etapa temprana los requisitos del software no se tienen claros aun, es por esto que se realiza un análisis de las funcionalidades y comportamientos para definir el sistema a desarrollar, las prioridades y alcances del proyecto. Con todos los interesados en el proyecto se establece una lista de los requisitos funcionales que definen lo que permite hacer el sistema al usuario y no funcionales que definen restricciones operacionales en el sistema.

En la Lista de Producto se organizan todas las funcionalidades requeridas para el desarrollo del producto, en conjunto con todos los interesados en el proyecto se genera esta lista, es importante comentar que el Dueño de Producto "Product Owner" es el responsable de mantener la Lista de Producto. Los elementos de la Lista de Producto tienen como atributos el código de requerimiento, la descripción, titulo, estado he importancia.

El Estado puede tener el valor de Revisión o Completado. En revisión indica que el requerimiento está en refinamiento, aún no está claro o definido completamente. Una vez el requerimiento está bien definido, a decisión y aprobación de todos los interesados del proyecto Product Owner, Equipo de desarrollo y Scrum Master pasa a estar al estado completado.

La Importancia se determina el valor que el requerimiento aporta al proyecto, se puede asignar de acuerdo a los numero 2, 5 o 10. Donde el número más alto tiene el mayor grado de importancia y por lo tanto ese requerimiento debe ser atendido primero antes que otro con menor número.

Tabla 2. Requerimiento número uno

Requerimiento R1			
Título		Notificaciones de las calificaciones a estudiantes	
Descripción		El Api Rest debe enviar notificaciones de las nuevas calificaciones registradas en el sistema académico de la Universidad a él Smartphone del estudiante.	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 3. Requerimiento número dos

Requerimiento R2			
Título		Horario de clases	
Descripción		La App debe mostrar el horario de clases correspondiente al periodo actual que cursa el estudiante	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 4. Requerimiento número tres

Requerimiento R3			
Título		Lista de materias semestre actual	
Descripción		La App debe mostrar en una lista las materias del semestre actual que cursa el estudiante	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 5. Requerimiento número cuatro

Requerimiento R4

Título		Perfil del estudiante	
Descripción		La App debe mostrar la información de perfil académico del estudiante	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 6. Requerimiento número cinco

Requerimiento R5			
Título		Pensum carrera	
Descripción		La App debe mostrar un archivo pdf con el pensum de la carrera profesional que cursa el estudiante	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 7. Requerimiento número seis

Requerimiento R6			
Título		Lista historial semestres cursados	
Descripción		La App debe mostrar una lista con los semestres cursados por el estudiante durante toda la carrera universitaria	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 8. Requerimiento número siete

Requerimiento R7	
Título	Lista materias por semestre

Descripción		La App debe mostrar una lista de las materias asociadas a un semestre cursado por el estudiante	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 9. Requerimiento número ocho

Requerimiento R8			
Título		Colores institucionales	
Descripción		La App debe implementar en la interfaz gráfica los colores que representa la institución	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 10. Requerimiento número nueve

Requerimiento R9			
Título		Calcular calificación	
Descripción		La App debe calcular la calificación que debe obtener el estudiante en el tercer corte para aprobar una materia	
Estado	Completado	**Importancia**	2

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 11. Requerimiento número diez

Requerimiento R10	
Título	Autentificación de estudiantes
Descripción	La App debe autentificar a los estudiantes mediante un inicio de sesión que recibe los

		datos de entrada Nombre Usuario y Contraseña	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 12. Requerimiento número once

Requerimiento R11			
Título		Detalle materia	
Descripción		La App debe mostrar la información correspondiente a una materia	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 13. Requerimiento número doce

Requerimiento R12			
Título		Detalle semestre	
Descripción		La App debe mostrar la información de detalle correspondiente al semestre	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 14. Requerimiento número trece

Requerimiento R13			
Título		Promedio semestres	
Descripción		La App debe mostrar en una gráfica de barras los promedios por semestre del estudiante	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 15. Requerimiento número catorce

Requerimiento R14			
Título		Cambiar contraseña estudiante	
Descripción		La App debe permitir cambiar la contraseña actual al estudiante	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 16. Requerimiento número quince

Requerimiento R15			
Título		Tokens de sesión	
Descripción		La comunicación entre App y Api Rest debe contener un Token autentificación en cada solicitud	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 17. Requerimiento número dieciséis

Requerimiento R16			
Título		Comprobar conexión a Internet	
Descripción		Antes de realizar cualquier solicitud a la Api Rest en la App se debe comprobar la conexión a Internet	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 18. Requerimiento número diecisiete

Requerimiento R17	
Título	Cache de información en la aplicación móvil

Descripción		La App debe implementar un mecanismo de chache para la disponibilidad de la información sin conexión a Internet	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 19. Requerimiento número dieciocho

Requerimiento R18			
Título		Encriptar la comunicación	
Descripción		La comunicación entre App y Api Rest debe ser mediante el protocolo seguro Https	
Estado	Completado	**Importancia**	2

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 20. Requerimiento número diecinueve

Requerimiento R19			
Título		Contraseña por defecto	
Descripción		Al iniciar sesión la contraseña del estudiante debe ser el número de código	
Estado	Completado	**Importancia**	5

Nota fuente: *Polo, A y Estupiñán, S (2018)*

Tabla 21. Requerimiento número veinte

Requerimiento R20			
Título		Diseño nativo aplicación móviles	
Descripción		Los componentes de interfaz gráfica se deben mostrar de forma nativa según el sistema operativo móvil	
Estado	Completado	**Importancia**	10

Nota fuente: *Polo, A y Estupiñán, S (2018)*

8.1.4. Planificación de Sprints

Luego de determinar los requerimientos funcionales y no funcionales en la Lista de Producto el Equipo de Desarrollo trabaja para proyectar la funcionalidad que se desarrollará durante el Sprint.

Tabla 22. Planeación de Sprint

Sprint	Historias Planificadas
Sprint 0	Información del proyecto
	Especificación de requerimientos
	Priorización de requisitos
	Capacitación
	Validación de requisitos
	Prototipo funcional
	Arquitectura del software
Sprint 1	Diagrama entidad relación
	Capacitación
	Diagrama de clases
	Notificaciones de las calificaciones a estudiantes
Sprint 2	Lista de materias semestre actual
	Perfil del estudiante
	Lista historial semestre cursados
Sprint 3	Lista materias por semestre
	Autentificación de estudiantes
	Detalle materia
	Detalle semestre
Sprint 4	Comprobar conexión a Internet
	Diseño nativo aplicaciones móviles
	Contraseña por defecto
	Caché de información en la aplicación móvil

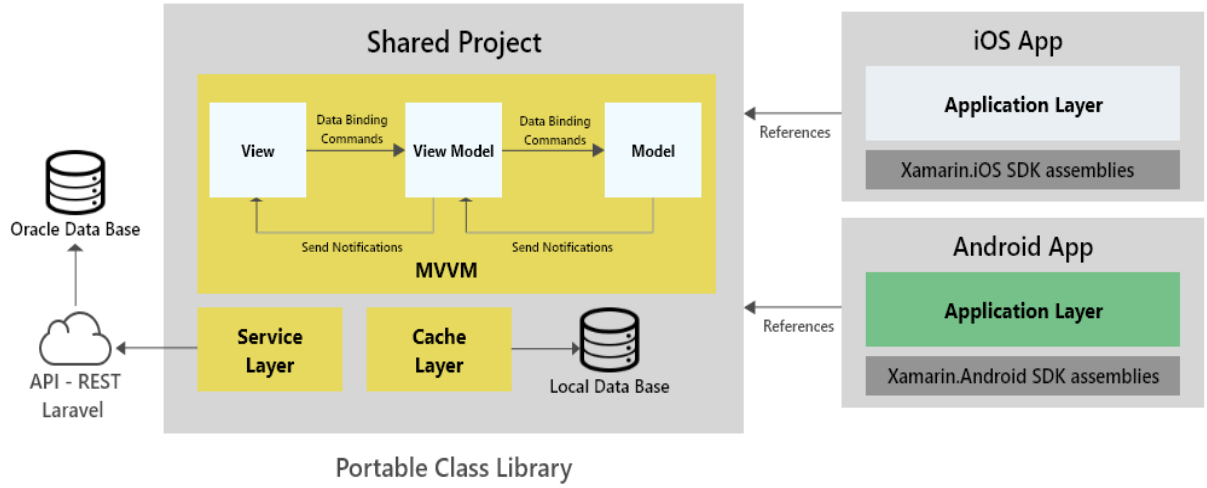
	Multiplataforma
	Token de sesión
Sprint 5	Cambiar contraseña estudiante
	Promedio semestre
	Colores institucionales
	Pensum carrera
Sprint 6	Horario de clases
	Calcular calificación

Nota fuente: *Polo, A y Estupiñán, S (2018)*

8.2.Diseño

8.2.1. Arquitectura

Xamarin propone una arquitectura construida principalmente para compartir código entre plataformas. La arquitectura de los dos enfoques Xamarin es muy similar con la diferencia de Xamarin.forms que además de compartir código de negocio comparte código de interfaz de usuario. El código compartido tiene lugar en el proyecto Core de la solución donde residen los patrones de diseño junto al código asociado a las capas lógicas arquitectónicas de negocio, servicio, acceso a datos y datos, referenciado por los proyectos de plataforma específica (Xamarin.iOS, Xamarin.Android o Windows) mediante una Portable Class Library (PCL) o .NET Standard Library, además los proyectos específicos hacen referencia a su SDK correspondiente de su plataforma. (Xamarin Platform,2018)



Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 10. Arquitectura de la aplicación móvil UH Notas

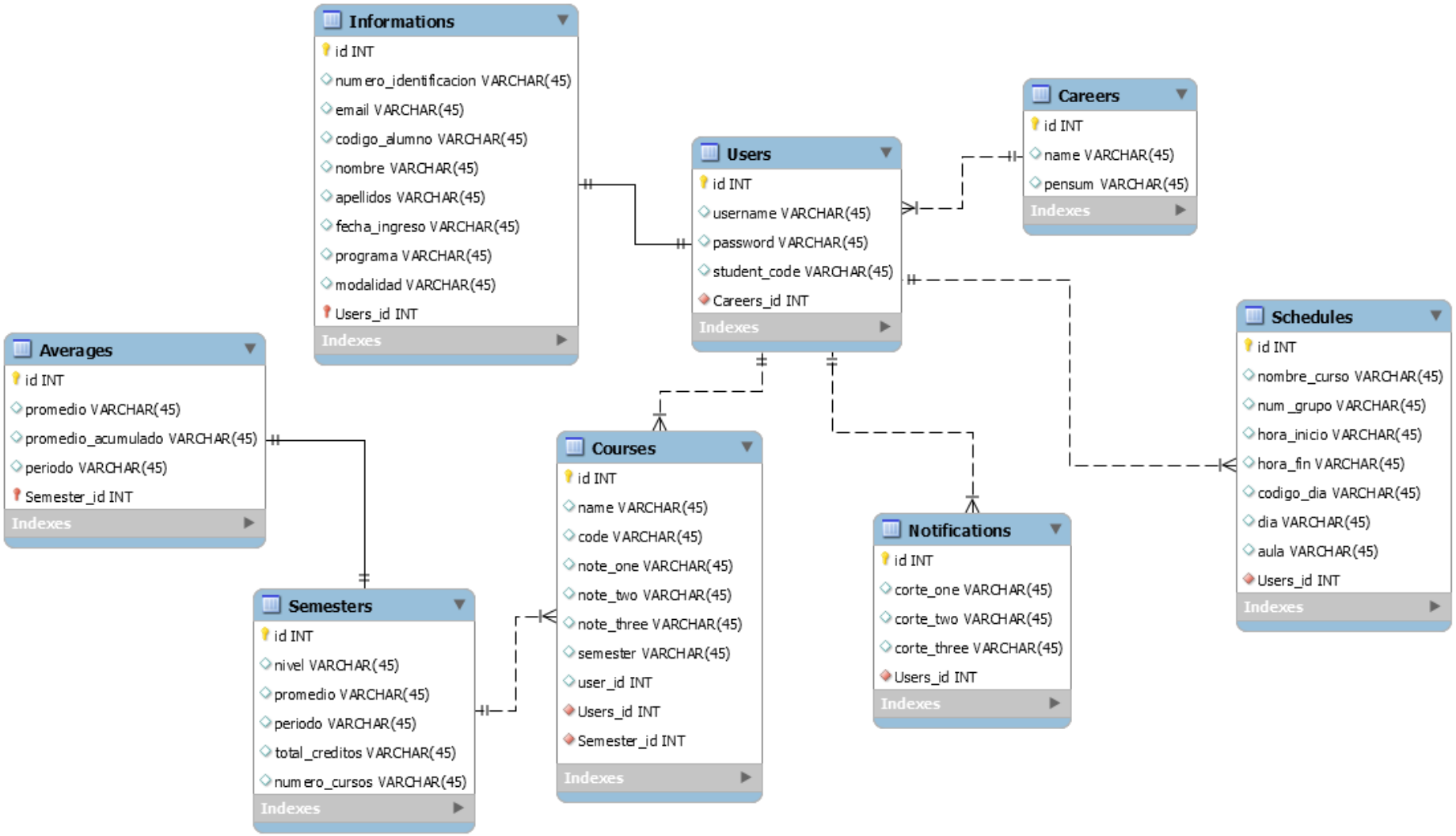
8.2.2. Modelo relacional de base de datos

El modelo relacional se ha establecido actualmente como el principal modelo de datos para las aplicaciones. Ha conseguido la posición principal debido a su simplicidad, que facilita el trabajo del programador en comparación con otros modelos. (Katz, 2018)

Proceso de diseño de bases de datos:

- **Análisis de requisitos** Recabar información sobre el uso que se piensa dar a la base de datos (elicitación de requisitos del sistema).
- **Diseño conceptual (modelo E/R)** Creación de un esquema conceptual de la base de datos independiente del DBMS que se vaya a utilizar.
- **Elección del sistema gestor de bases de datos** Elección del modelo de datos (tipo de DBMS) y del DBMS concreto.
- **Diseño lógico** Creación del esquema conceptual para el modelo de datos del DBMS elegido (p.ej. paso del modelo E/R a un conjunto de tablas).
- **Diseño físico** Creación de la base de datos utilizando el DDL (lenguaje de definición de datos del DBMS). (Katz, 2018)

El proyecto UH Notas en particular consume información académica relacionada al estudiante de la base de datos en Oracle de la institución. De igual forma utiliza una base de datos MySql como respaldo y repositorio de información auxiliar para los procesos que realiza la aplicación móvil. Con el fin de mejorar el entendimiento de la base de datos en general, se diseña un solo diagrama con los modelos utilizados de Oracle y MySql.



Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 11. Modelo relacional de base de datos

8.2.3. Api Rest

En el año 2000, Roy Fielding describió “Representational State Transfer” - REST como un estilo arquitectónico con principios de ingeniería de software y restricciones aplicadas a los elementos dentro de la arquitectura para la construcción de sistemas distribuidos de hipermedia. (Fielding, 2000)

Al día de hoy REST se aplica comúnmente al diseño de APIs modernas. En términos generales, una API es la interfaz de comunicación que permite exponer funciones, información o servicios entre programas. (Fallis, 2013)

Las restricciones o reglas de REST permiten diseñar una API coherente para los clientes que la utilizan. Principalmente las reglas proponen las mejores prácticas asociadas a:

- El nombre correcto de segmentos de ruta URI.
- Correlación de las operaciones CRUD con las URI.
- Códigos de respuesta HTTP apropiados para un escenario dado.
- Estructura del formato JSON.
- Versionamiento.

8.2.3.1. Principios REST

Algunos de los principios de diseño principales de las API REST que usan HTTP son:

- Las API REST están diseñadas en torno a recursos, que son cualquier tipo de objeto, datos o servicio al que puede acceder el cliente.
- Un recurso tiene un identificador, que es un URI que identifica de manera única ese recurso.
- Los clientes interactúan con un servicio intercambiando representaciones de recursos en formatos JSON o XML.

- Las API REST basadas en HTTP, la interfaz uniforme incluye el uso de verbos HTTP estándar para realizar operaciones en los recursos. Las operaciones más comunes son GET, POST, PUT, PATCH y DELETE.(Azure, 2018)

Leonard Richardson propuso el siguiente modelo de madurez para las API Web:

- Nivel 0: defina un URI y todas las operaciones son solicitudes POST para este URI.
- Nivel 1: crear URI separados para recursos individuales.
- Nivel 2: use los métodos HTTP para definir operaciones en los recursos.
- Nivel 3: use hipermedia (HATEOAS, descrito a continuación). (Fowler, 2010)

Dentro del proyecto se utilizaron algunas de las reglas del marco "Lenguaje de Modelado de Recursos Web" – WRML, que propone Mark Massé tomando como referencia los principios de diseño en REST. (Fallis, 2013)

8.2.3.2. Identificador de recursos URI

Los identificadores de recursos uniformes son utilizados por REST para comunicar recursos o modelos de información.

Regla: el separador de barra inclinada (/) debe usarse para indicar una jerarquía o Relación entre los recursos.

Regla: los guiones (-) se deben usar para mejorar la legibilidad de los URI
Para hacer que sus URI sean fáciles de escanear e interpretar.

Regla: las letras minúsculas deben preferirse en las rutas URI. ya que las mayúsculas pueden a veces causar problemas.

Regla: mantener el formato consistente y usar siempre un plural en cada segmento de la URI.

8.2.3.3. Interacción con HTTP

Cada método HTTP tiene una semántica específica y bien definida dentro del contexto del modelo de recursos de una API REST. (Fallis, 2013)

- GET recuperar una representación del estado de un recurso.
- HEAD se utiliza para recuperar los metadatos asociados a un recurso.
- PUT debe usarse para agregar un nuevo recurso o actualizar un recurso.
- DELETE elimina un recurso existente.
- POST debe usarse para crear un nuevo recurso dentro de una colección.

8.2.3.4. Códigos de estado de respuesta HTTP

Las API REST utilizan los mensajes de estado de respuesta HTTP para informar a los clientes sobre el resultado general de su solicitud. HTTP define cuarenta códigos de estado estándar que se pueden usar para transmitir los resultados de una solicitud del cliente Los códigos de estado se dividen en las cinco categorías: (Fallis, 2013)

- **Informativo**: comunica la información del nivel de protocolo de transferencia.
- **Exitoso**: Indica que la solicitud del cliente fue aceptada con éxito.
- **Redireccionamiento**: indica que el cliente debe realizar alguna acción adicional para completar su solicitud.
- **Error de cliente**: esta categoría de códigos de estado de error señala con el dedo a los clientes.
- **Error del servidor:** el servidor asume la responsabilidad de estos códigos de estado de error.

Tabla 23. Códigos HTTP

Código	Nombre	Descripción
200	Ok	Indica solicitud realizada con éxito
201	Creado	Indica que se ha creado un nuevo recurso
204	Sin contenido	Indica que el cuerpo se ha dejado intencionalmente en blanco
400	Solicitud mal habida	Indica un error de cliente no específico
401	No autorizado	El cliente proporcionó credenciales no válidas u olvidó enviarlas
402	Prohibido	Denegación del acceso a un recurso protegido
404	No encontrado	Cuando el cliente intentó interactuar con un URI que la API REST no pudo mapear a un recurso
405	Método no permitido	Cuando el cliente intentó interactuar utilizando un método HTTP no compatible
500	Error interno del servidor	Indica al cliente que la API tiene problemas propios

Nota fuente: *Mark Massé (2012)*

Libro fuente: *REST API Design Rulebook*

8.2.3.5. Respuesta JSON

Toda solicitud a la API UH Notas debe representar los recursos en formato JSON en la repuesta. Con el propósito de estandarizar el diseño de API se tomó como base la especificación API JSON que define como un servidor debe responder a las solicitudes.

La estructura de un recurso en la especificación API JSON, se identifica por el tipo de medio application/ json. Los recursos de la API están definidos en JavaScript Object Notation – JSON. Un objeto JSON debe estar en la raíz de cada respuesta de la API que contenga datos. Este objeto define el "nivel superior" de un recurso. (Katz, 2018)

Un recurso debe contener al menos uno de los siguientes miembros de nivel superior:

- **data**: los "datos primarios" del recurso
- **errors**: una matriz de objetos de error
- **meta**: contiene meta información no estándar.

  El miembro data y errors no deben coexistir en el mismo documento.

  Un recurso puede contener alguno de estos miembros de nivel superior:

- **jsonapi**: un objeto que describe la implementación del servidor
- **links**: un objeto de enlaces relacionado con los datos primarios.
- **included**: una matriz de objetos de recursos que están relacionados con los datos primarios y / o entre ellos ("recursos incluidos").

  El objeto de enlaces de nivel superior puede contener los siguientes miembros:

- **self**: el enlace que generó el documento de respuesta actual.
- **related**: un enlace de recurso relacionado cuando los datos primarios representan una relación de recursos.

  Ejemplo:


```
{
    "data": [
        {
            "nombre": "COSTOS PRECIOS Y COTIZACIONES INTERNACIONALES",
            "codigo": "69038",
            "primer_corte": "4.2",
            "segundo_corte": "-",
            "tercer_corte": "-",
            "links": {
                "detalle": "http://uhnotasapi.loc/api/students/courses/69038"
            }
        },
        {
            "nombre": "INGLES TECNICO III",
            "codigo": "10071",
            "primer_corte": "5",
            "segundo_corte": "-",
            "tercer_corte": "-",
            "links": {
                "detalle": "http://uhnotasapi.loc/api/students/courses/10071"
            }
        },
```


Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 12. Ejemplo de respuesta en formato JSON

8.2.3.6. Versionamiento

El control de versiones ayuda a iterar más rápido y evita que las solicitudes no válidas lleguen a los puntos finales actualizados. También ayuda a suavizar las principales transiciones de versión de API, además puede continuar ofreciendo versiones antiguas de API durante un período de tiempo. (Sahni, 2015)

Una API nunca va a ser completamente estable. El cambio es inevitable. Lo importante es cómo se maneja ese cambio. Los calendarios de desaprobación de varios meses bien documentados y anunciados pueden ser una práctica aceptable para muchas API. Todo se reduce a lo razonable dada la industria y los posibles consumidores de la API. (Sahni, 2015)

Existen opiniones mixtas sobre si una versión de API debe incluirse en la URL o en un encabezado. Académicamente hablando, probablemente debería estar en un encabezado. Sin embargo, la versión debe estar en la URL para garantizar la capacidad de exploración del navegador de los recursos en todas las versiones. (Sahni, 2015)

8.2.3.7.Modelo REST

Siguiendo el estilo arquitectónico de REST para API se diseñó un formato que define cada recurso expuesto por la interfaz de programación de UH Notas.

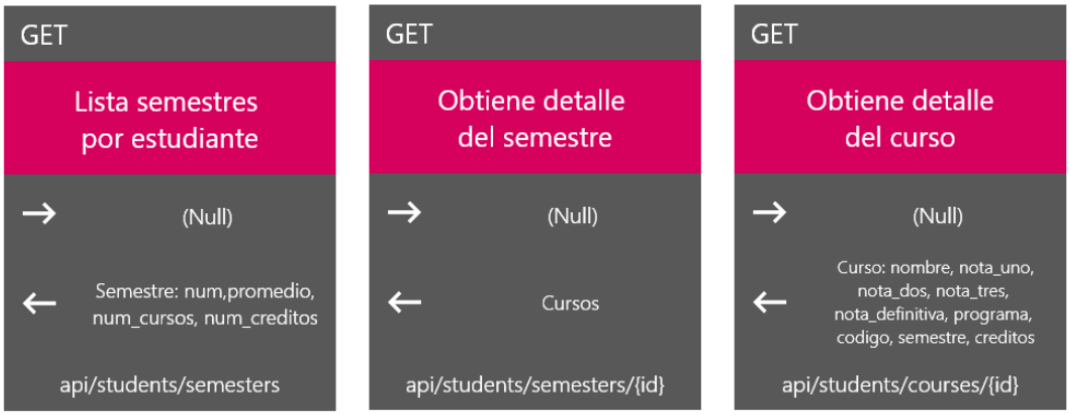
Campos:

- **Método**: Nombre del verbo o método HTTP
- **Entrada**: Datos de la solicitud (form-data).
- **Salida**: Datos de la respuesta.
- **End-Points**: Dirección del recurso.

Tabla 24. Formato definición de un recurso API REST

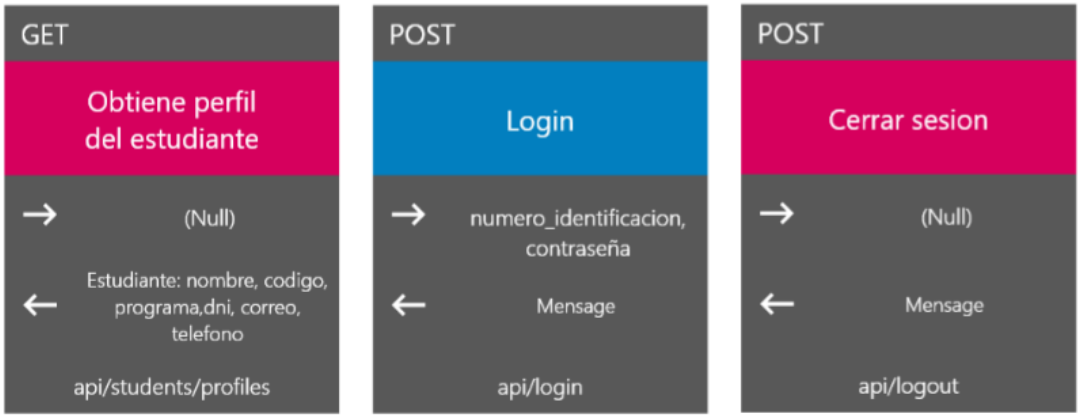
Método	
Descripción	
Entrada	Datos de entrada
Salida	Datos de entrada
URI o End-Point	

Recuperado de: *RESTful Web Services with PHP and Laravel*
Nota fuente: *Max Schwarzmueller*



Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 13. Definición de recursos API REST número uno



Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 14. Definición de recursos API REST número dos

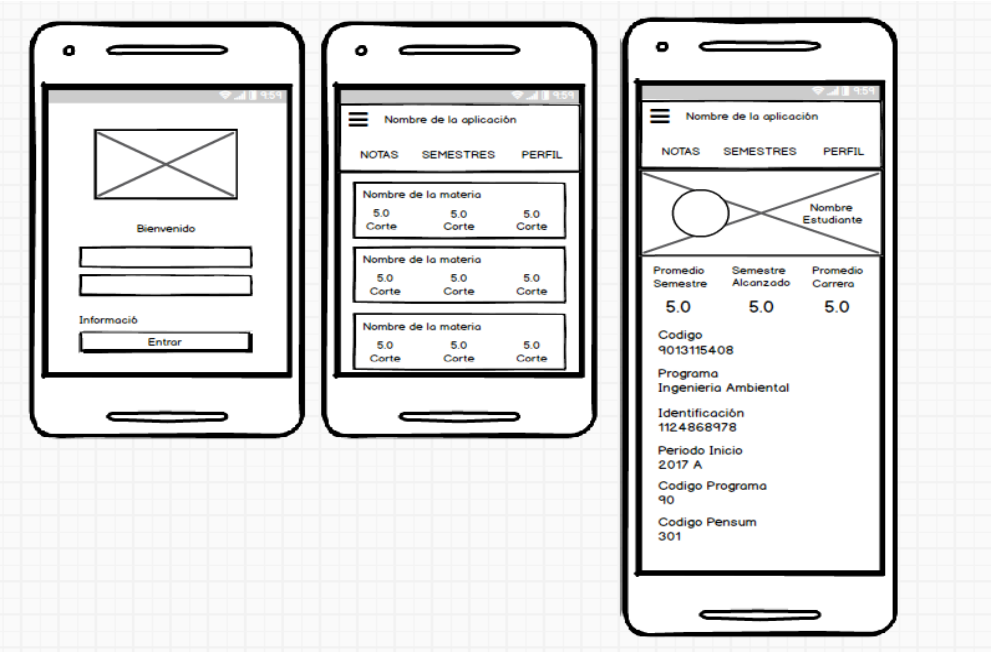
Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 15. Definición de recursos API REST número tres

Nota fuente: *Polo, A y Estupiñán, S (2018)*

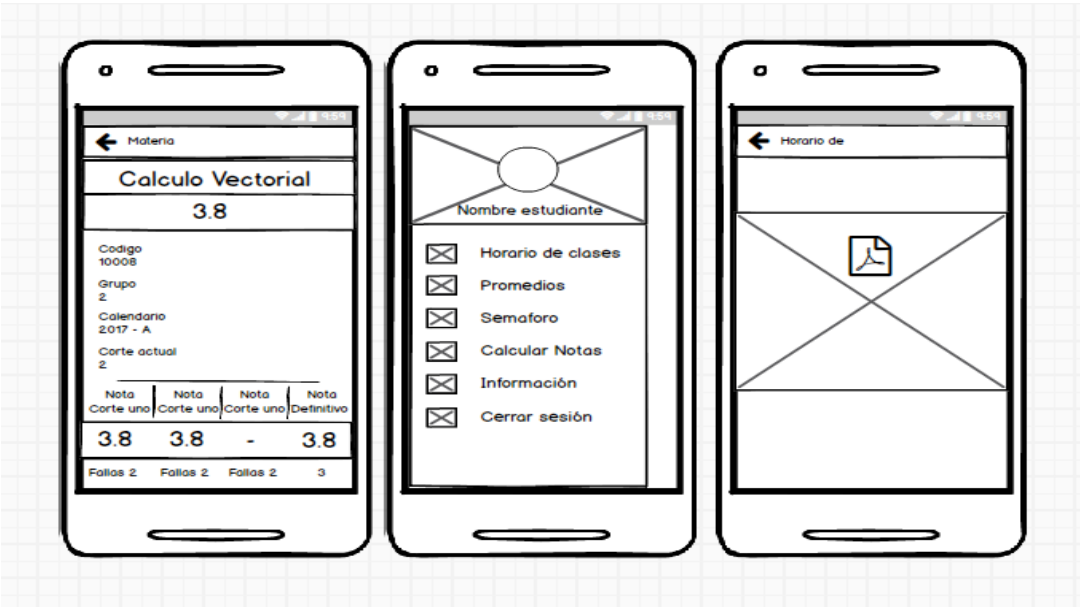
Ilustración 16. Definición de recursos API REST número cuatro

8.2.4. Wireframes



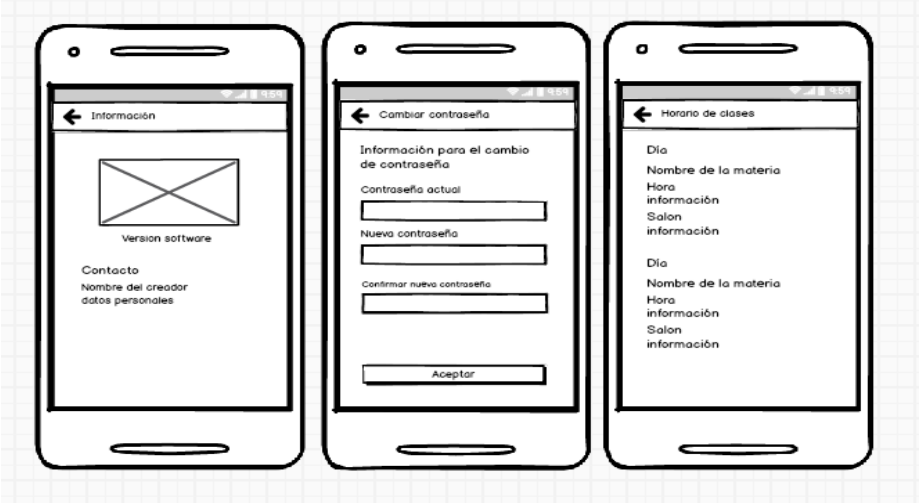
Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 17. Wireframes, inicio de sesión, menú principal y perfil usuario



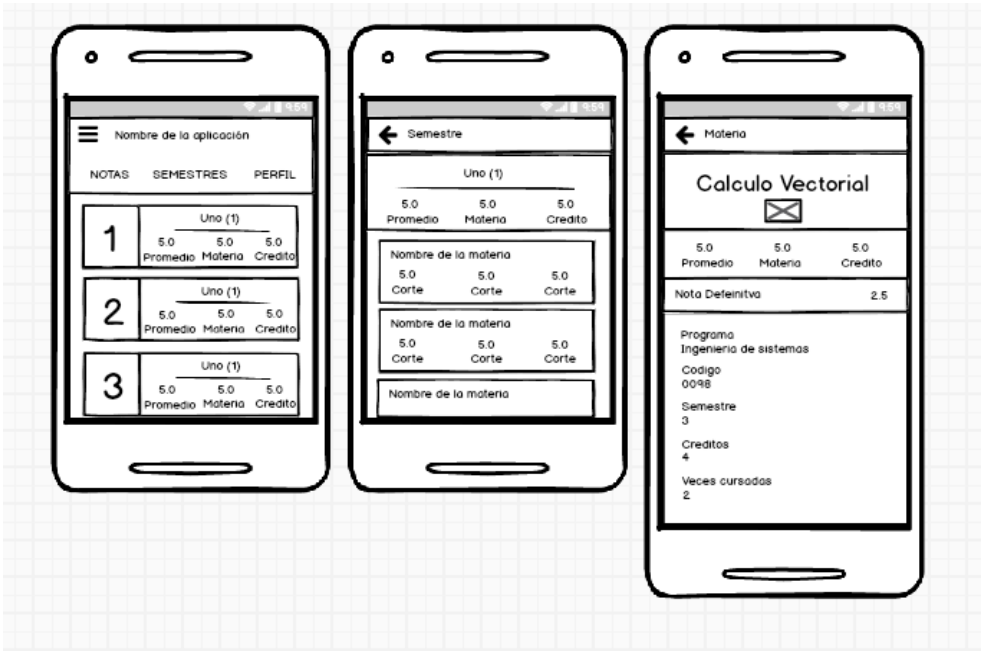
Nota fuente: *Polo, A y Estupiñán, S (2018)*

Ilustración 18. Wireframes, información materia, menú desplegable, horario de clase uno



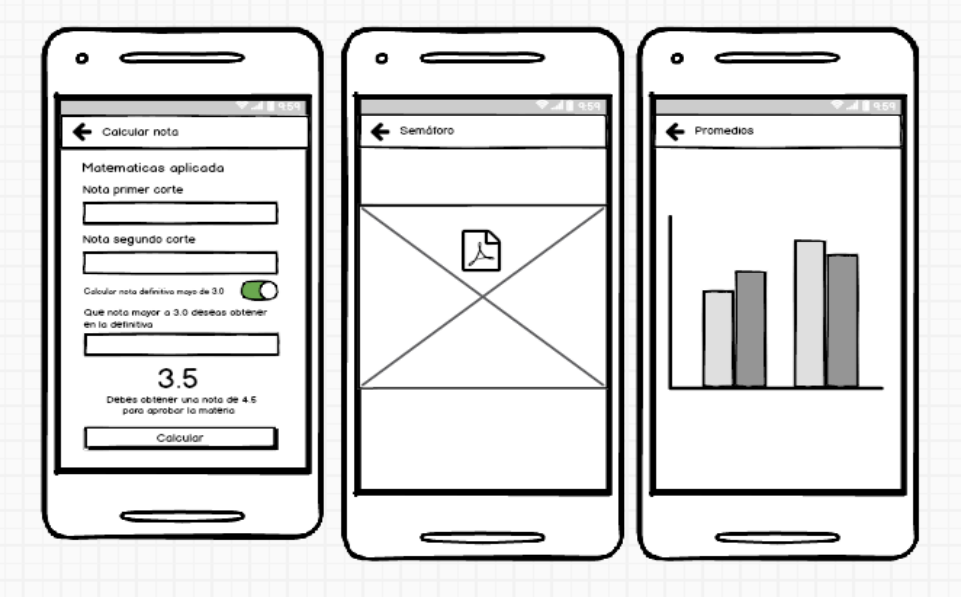
***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 19. Wireframes, información App, cambio contraseña, horario de clases dos



***Nota fuente**: Polo, A y Estupiñán, S (2018)*

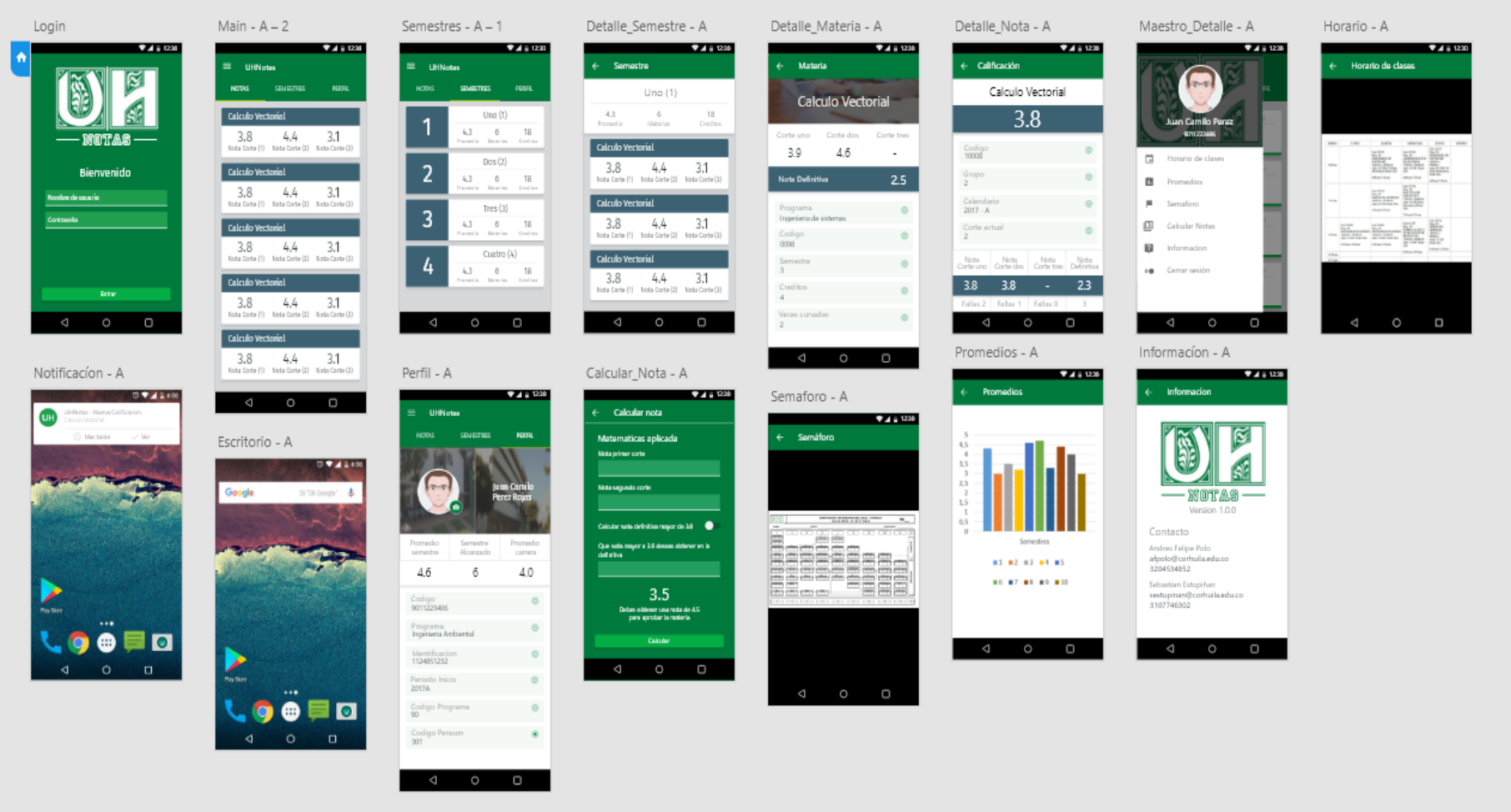
Ilustración 20. Wireframes, semestres generales, información semestre y materia



***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 21. Wireframes, calcular nota, plan de estudio, promedios generales

8.2.5. Prototipo



***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 22. Prototipo aplicación móvil UH Notas

8.3. Codificación

8.3.1. Estándar de programación

Un estilo de codificación permite a todos los miembros del equipo de desarrollo poder leer y entender más fácil el código mediante el seguimiento de reglas. De esta forma se evita que cada programador escriba código diferente en el sentido de la sintaxis, estructura y nomenclaturas.

En la programación de la aplicación móvil y Api Rest se utilizó las convenciones de nombres. Conocidos como Pascal-Case (primera letra de cada palabra empieza en mayúscula) y Camel-Case. (primera letra de la primera palabra empieza en minúscula y sus consecuentes en mayúsculas). Simplemente son la forma en que damos nombre a nuestras clases, atributos, variables, métodos, funciones o constantes de nuestro proyecto.

```php
trait ApiResponser
{
    /**
     * @param $data
     * @param int $code
     * @return \Illuminate\Http\JsonResponse
     */
    protected function successResponse($data, $code = 200)
    {
        return response()->json(['success'=>$data],$code);
    }

    /**
     * @param $error
     * @param $code
     * @param null $errors
     * @return \Illuminate\Http\JsonResponse
     */
    protected function errorResponse($error, $code, $errors = null)
    {
        $error = $this->errorFormatter($error,$code,$errors);
        return response()->json(['error' => $error], $code);
    }
```

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 23. Implementación de estándar de codificación

En la documentación de código se utiliza las sintaxis de comentario en bloque y se documentan solo las clases, atributos y métodos del proyecto. Además, en la API se implementa Swagger como generador de documentación y la especificación Open Api.

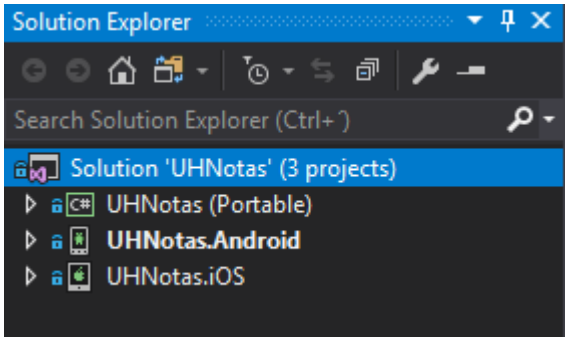
```php
<?php

namespace App\Http\Controllers;

use App\Traits\ApiNullResource;
use App\Traits\ApiResponser;

/**
 * @SWG\Swagger(
 *   basePath="/api",
 *   @SWG\Info(
 *     title="UHNotas API",
 *     version="1.0.0",
 *     @SWG\Contact(
 *          name="andresfvpclap@gmail.com"
 *     )
 *   )
 * )
 */
class ApiController extends Controller
{
    use ApiResponser;
    use ApiNullResource;
}
```

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 24. Implementación del documentador de la API

8.3.2. Estructura del proyecto Xamarin

Es fundamental detallar cómo se compone y estructura una aplicación Xamarin.forms. Visual Studio de Microsoft es el IDE por defecto para trabajar en la construcción de aplicaciones con Xamarin, es importante aclarar que Visual Studio organiza el código en soluciones y proyectos. Una solución es un contenedor que puede incluir uno o varios proyectos. Un proyecto puede ser una aplicación, una biblioteca auxiliar o una aplicación de prueba, entre otros. De esta misma forma se organiza una aplicación de Xamarin.forms.



***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 25. Solución con el nombre de la App que contiene los proyectos

- **UHNotas:** este proyecto es el de la biblioteca de .NET Standard o Portable Class Library que incluye todo el código compartido y la interfaz de usuario compartida, conocido como proyecto Core o Shared Project.
- **UHNotas.Android:** este proyecto incluye el código específico de Android y es el punto de entrada de la aplicación Android.

- **UHNotas.iOS:** este proyecto incluye el código específico de iOS y es el punto de entrada de la aplicación de iOS.
- **UHNotas.UWP:** este proyecto incluye el código específico de plataforma universal de Windows (UWP) y es el punto de entrada de la aplicación de UWP.(Microsoft Xamarin, 2018)

  El proyecto compartido UHNotas consta de los archivos.

  - **App.xaml:** el marcado XAML para la clase App, que define un diccionario de recursos para la aplicación.
  - **App.xaml.cs:** el código subyacente para la clase App, que es el responsable de crear instancias de la primera página que se mostrarán mediante la aplicación en cada plataforma, y para controlar los eventos del ciclo de vida de la aplicación.

- **MainPage.xaml:** el marcado XAML para la clase MainPage, que define la interfaz de usuario para la página que se muestra cuando se inicia la aplicación.
- **MainPage.xaml.cs:** el código subyacente para la clase MainPage, que contiene la lógica de negocios que se ejecuta cuando el usuario interactúa con la página.

En Shared Project existe una clase de nombre App encargada de instanciar la página principal que se va a mostrar en cada plataforma.

```
namespace UHNotas
{
    public partial class App : Application
    {
        public static MasterPage Master { get; internal set; }

        public App()
        {
            InitializeComponent();

            //MainPage = new UHNotas.LoginPage();
            MainPage = new UHNotas.Pages.MasterPage();
        }
    }
}
```

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 26. Código correspondiente a la clase App

En cada uno de los proyectos de plataforma específica se inicializa la página principal instanciada en el proyecto Core de Xamarin.forms.

En el proyecto Android en la clase MainActivity inicializa el marco de Xamarin Forms y carga la clase App del proyecto compartido.

```
namespace UHNotas.Droid
{
    [
        Activity(Label = "UHNotas",
        Icon = "@drawable/ic_launcher",
        Theme = "@style/Theme.MyTheme",
        MainLauncher = false,
        ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation)
    ]
    public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void OnCreate(Bundle bundle)
        {
            TabLayoutResource = Resource.Layout.Tabbar;
            ToolbarResource = Resource.Layout.Toolbar;

            base.OnCreate(bundle);

            global::Xamarin.Forms.Forms.Init(this, bundle);
            LoadApplication(new App());
        }
    }
}
```

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 27. Clase MainActivity donde se inicia la App para Android

En el proyecto iOS en la clase AppDelegate en el método FinishedLaunching inicializa Xamarin.Forms y carga la clase App del proyecto compartido mediante una instancia de la misma.

```
namespace UHNotas.iOS
{

    [Register("AppDelegate")]
    public partial class AppDelegate : global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
    {

        public override bool FinishedLaunching(UIApplication app, NSDictionary options)
        {
            global::Xamarin.Forms.Forms.Init();
            LoadApplication(new App());

            return base.FinishedLaunching(app, options);
        }
    }
}
```

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 28. Clase AppDelagate donde se inicia la App para iOS

De esta forma es como en Xamarin.forms es posible crear aplicaciones móviles que se ejecuten en diferentes sistemas operativos y compartan una cantidad considerable de código, hasta el momento hemos mencionado solo las plataformas móviles, pero Xamarin en su abanico de opciones posibilita el desarrollo de Apps multiplataforma orientadas a Desktop o Smart TVs.

8.3.3. Codificación interfaz grafica

En Xamarin.Forms la interfaz de usuario es escrita con XAML y es mostrada de forma nativa para dispositivos móviles iOS, Android y UWP. Dentro de un archivo XAML, el desarrollador puede definir interfaces de usuario utilizando todas las vistas, diseños y páginas de Xamarin.Forms, así como clases personalizadas. El archivo XAML puede compilarse o incorporarse en el ejecutable. De cualquier manera, la información XAML se analiza en tiempo de compilación para localizar objetos con nombre, y nuevamente en tiempo de ejecución para instanciar e inicializar objetos, y para establecer enlaces entre estos objetos y el código de programación. (XAML, 2018)

XAML tiene varias ventajas sobre el código equivalente:

- XAML es a menudo más breve y legible que el código equivalente.
- La jerarquía padre-hijo inherente en XML permite a XAML imitar con mayor claridad visual la jerarquía padre-hijo de los objetos de la interfaz del usuario.
- XAML puede ser escrito a mano fácilmente por los programadores, pero también se presta a ser procesable y generado por herramientas de diseño visual.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
                  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                  xmlns:menu="clr-namespace:UHNotas.Pages"
                  xmlns:pages="clr-namespace:UHNotas.Pages"
                  x:Class="UHNotas.Pages.MasterPage">
    <MasterDetailPage.Master>
        <menu:MenuMasterPage Title="Menu"/>
    </MasterDetailPage.Master>
    <MasterDetailPage.Detail>
        <NavigationPage>
            <x:Arguments>
                <pages:TabbedMasterPage Title="UHNotas"/>
            </x:Arguments>
        </NavigationPage>
    </MasterDetailPage.Detail>
</MasterDetailPage>
```

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 29. Código XAML del menú de la App

```xaml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:control="clr-namespace:UHNotas.Controls"
             x:Class="UHNotas.Pages.MenuMasterPage">
    <ContentPage.Content>
        <StackLayout>
            <Grid BackgroundColor="Transparent">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                </Grid.RowDefinitions>
                <Grid>
                    <Image Source="bg_uh.png" Aspect="AspectFill" />
                    <StackLayout Padding="0,20,0,0"
                                 HorizontalOptions="CenterAndExpand"
                                 VerticalOptions="CenterAndExpand">
                        <Image Source="img_user.png"
                               Aspect="AspectFit"
                               WidthRequest="90"
                               HeightRequest="90" />
                        <Label Text="Andres Felipe Polo"
                               TextColor="White"
                               FontSize="Medium" />
                    </StackLayout>
                </Grid>
            </Grid>
```

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 30. Código XAML detalle del menú de la App

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

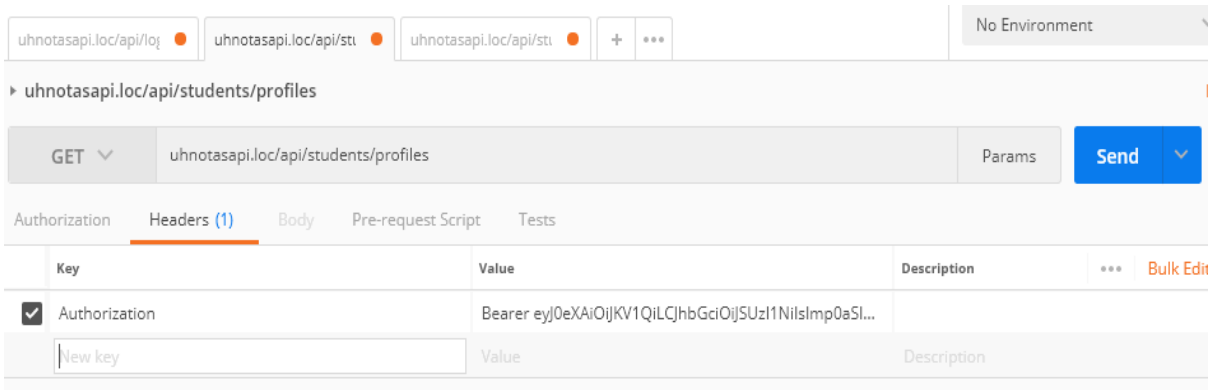
Ilustración 31. Simulación de la App en iOS y Android correspondientemente

8.4. Pruebas

En esta etapa procedemos a demostrar que el sistema no presenta errores y realiza la funcionalidad esperada. Así mismo se busca encontrar cualquier defecto, falla o error que el sistema pueda presentar en su funcionamiento para finalmente ejecutar tareas de corrección.

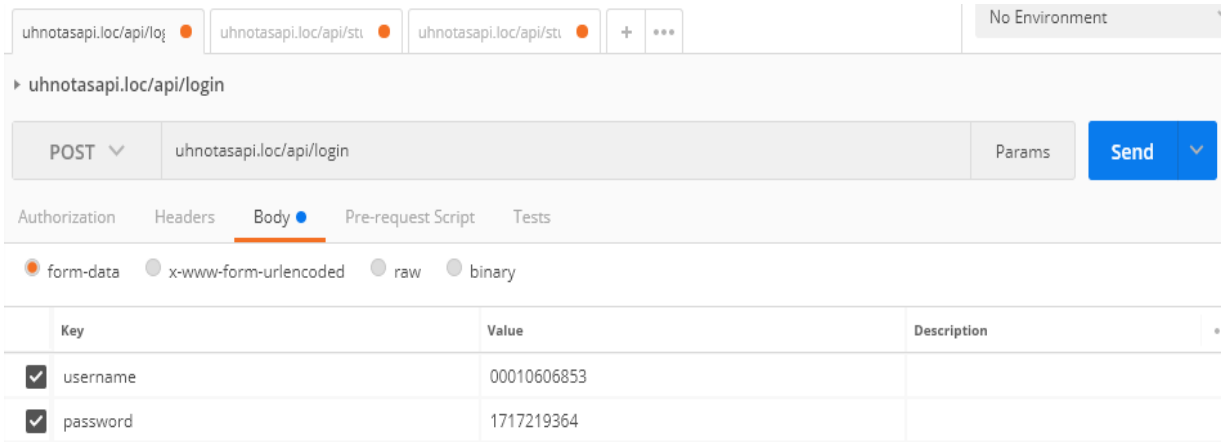
PostMan es una herramienta de Testing que permite realizar pruebas funcionales a la API REST. Mediante una interfaz de usuario es posible crear solicitudes Http. Por ejemplo, una solicitud POST a la cual se puede agregar valores en el cuerpo y en las cabeceras de cada request.

Es posible construir las solicitudes Http correspondientes al proyecto seleccionando el tipo de solicitud (POST, GET, PUT o DELETE) y apuntar al DNS o IP donde se aloja la API REST, así como establecer variables en el cuerpo de la solicitud, con todo listo se ejecuta y obtienen los resultados.



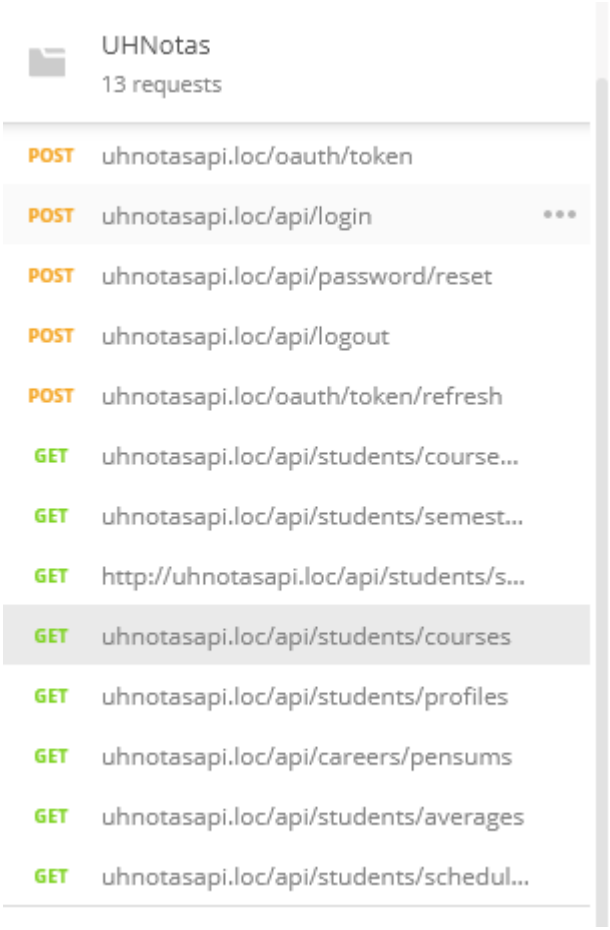
***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 32. Prueba de solicitud Http al recurso API de perfil del usuario



***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 33. Prueba de solicitud Http inicio de sesión del usuario



***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 34. End-Point de la API REST UH Notas

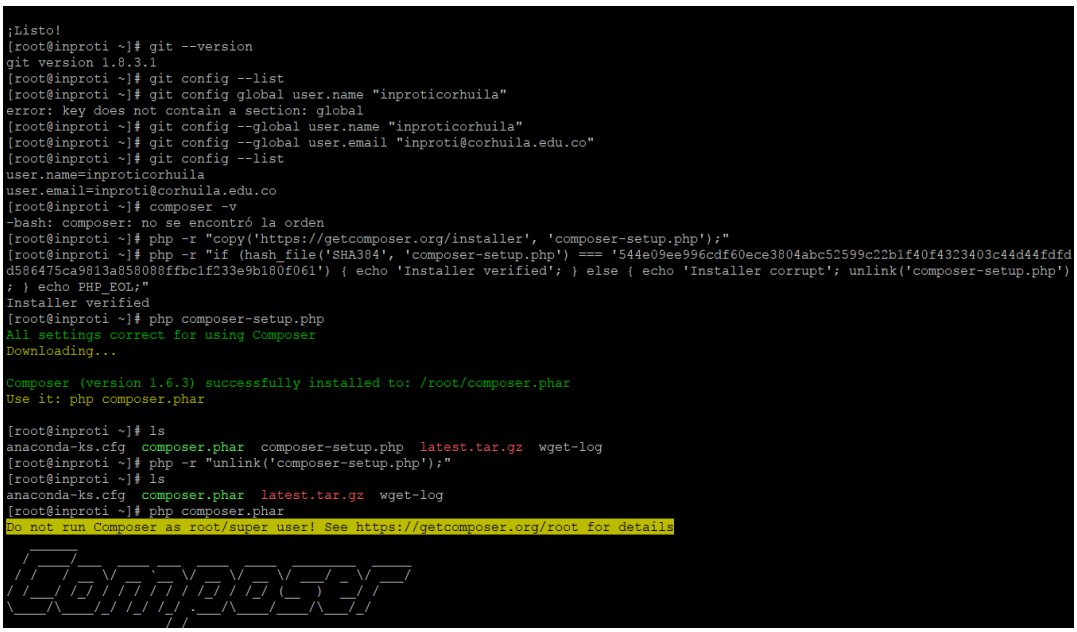
8.5. Implementación

El servidor de producción provisto para alojar la API REST opera bajo el sistema operativo CentOS 7. Teniendo en cuenta que el proyecto está construido con Laravel, se debe aprovisionar el servidor con Php en su última versión disponible. Además de un servidor Web como Apache o Nginx. Laravel 5.5 en particular requiere:

- PHP > = 7.0.0
- Extensión PHP OpenSSL
- Extensión PDO PHP
- Extensión PHP Mbstring
- Extensión de Tokenizer PHP

- Extensión XML PHP

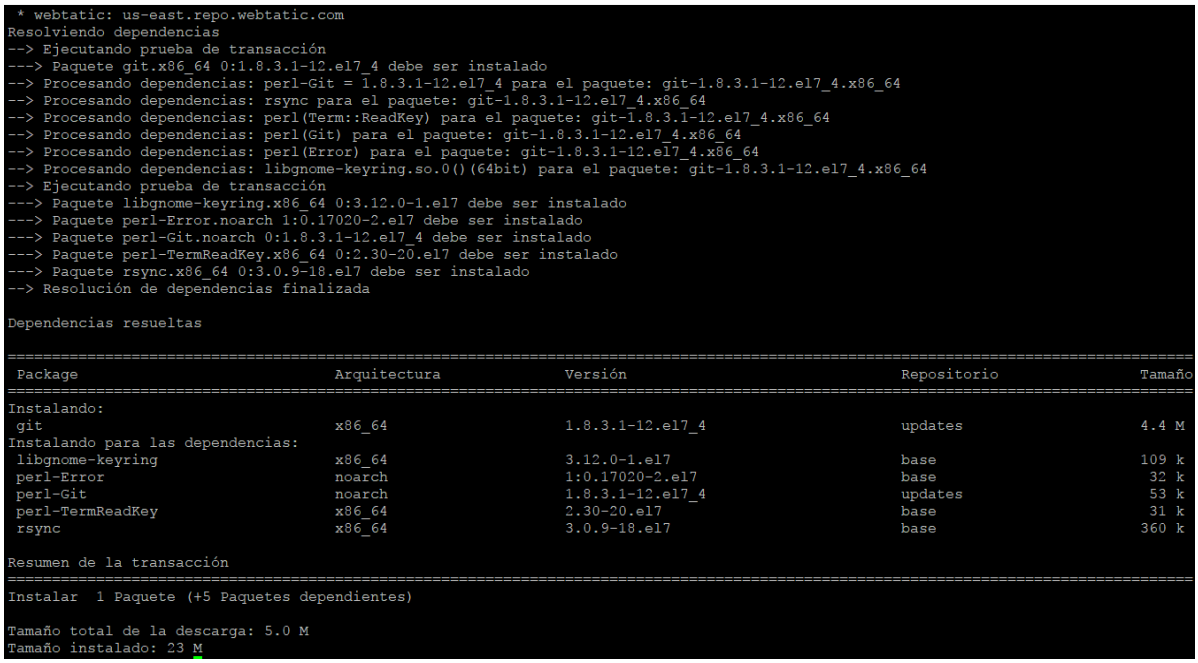
La instalación de Composer como gestor de paquetes es necesaria debido a que Laravel por este medio instala cada una de sus dependencias. Paso siguiente realizamos las respectiva instalación y configuración.



***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 35. Configuración de Git e instalación de Composer

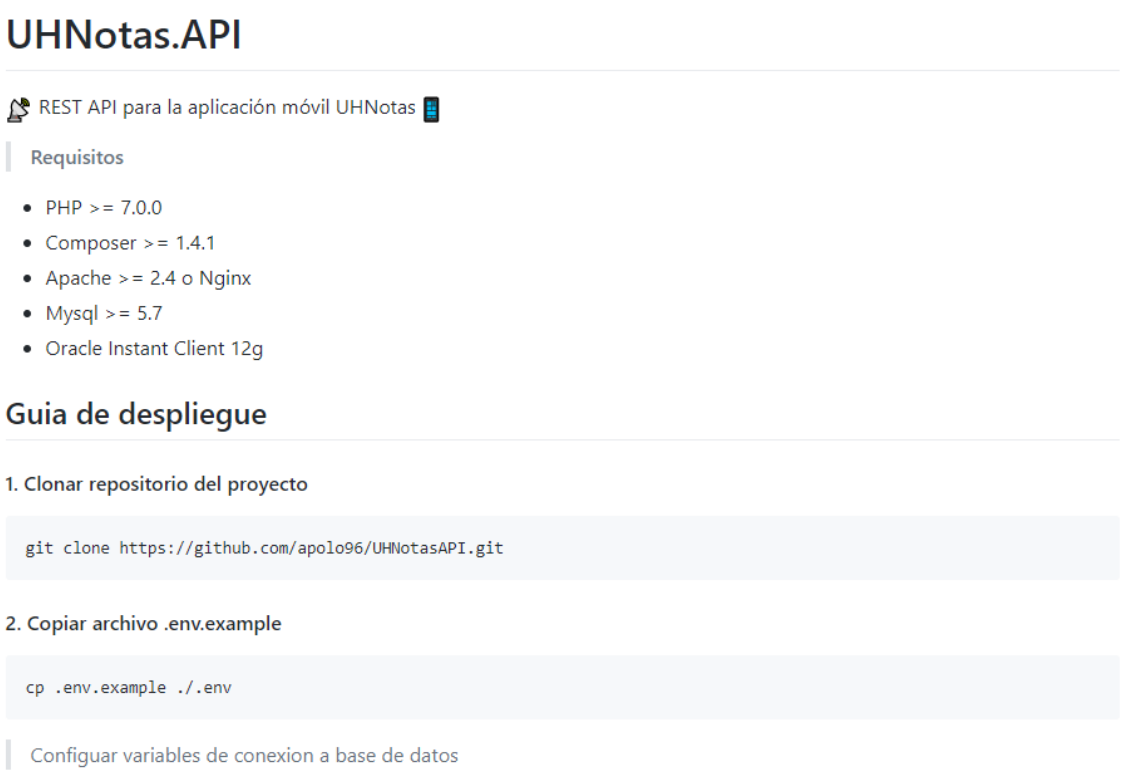
Una vez el servidor cumpla con todos los requisitos, se procede a realizar el despliegue del proyecto. Que se encuentra alojado en un repositorio de GitHub, para ellos requerimos del sistema de control de versiones Git.



```
 * webtatic: us-east.repo.webtatic.com
Resolviendo dependencias
--> Ejecutando prueba de transacción
---> Paquete git.x86_64 0:1.8.3.1-12.el7_4 debe ser instalado
--> Procesando dependencias: perl-Git = 1.8.3.1-12.el7_4 para el paquete: git-1.8.3.1-12.el7_4.x86_64
--> Procesando dependencias: rsync para el paquete: git-1.8.3.1-12.el7_4.x86_64
--> Procesando dependencias: perl(Term::ReadKey) para el paquete: git-1.8.3.1-12.el7_4.x86_64
--> Procesando dependencias: perl(Git) para el paquete: git-1.8.3.1-12.el7_4.x86_64
--> Procesando dependencias: perl(Error) para el paquete: git-1.8.3.1-12.el7_4.x86_64
--> Procesando dependencias: libgnome-keyring.so.0()(64bit) para el paquete: git-1.8.3.1-12.el7_4.x86_64
--> Ejecutando prueba de transacción
---> Paquete libgnome-keyring.x86_64 0:3.12.0-1.el7 debe ser instalado
---> Paquete perl-Error.noarch 1:0.17020-2.el7 debe ser instalado
---> Paquete perl-Git.noarch 0:1.8.3.1-12.el7_4 debe ser instalado
---> Paquete perl-TermReadKey.x86_64 0:2.30-20.el7 debe ser instalado
---> Paquete rsync.x86_64 0:3.0.9-18.el7 debe ser instalado
--> Resolución de dependencias finalizada

Dependencias resueltas

================================================================================================================================
 Package                        Arquitectura            Versión                       Repositorio               Tamaño
================================================================================================================================
Instalando:
 git                            x86_64                  1.8.3.1-12.el7_4              updates                  4.4 M
Instalando para las dependencias:
 libgnome-keyring               x86_64                  3.12.0-1.el7                  base                     109 k
 perl-Error                     noarch                  1:0.17020-2.el7               base                      32 k
 perl-Git                       noarch                  1.8.3.1-12.el7_4              updates                   53 k
 perl-TermReadKey               x86_64                  2.30-20.el7                   base                      31 k
 rsync                          x86_64                  3.0.9-18.el7                  base                     360 k

Resumen de la transacción
================================================================================================================================
Instalar  1 Paquete (+5 Paquetes dependientes)

Tamaño total de la descarga: 5.0 M
Tamaño instalado: 23 M
```

***Nota fuente**: Polo, A y Estupiñán, S (2018)*

Ilustración 36. Instalación de dependencias del proyecto



UHNotas.API

REST API para la aplicación móvil UHNotas

> Requisitos

- PHP >= 7.0.0
- Composer >= 1.4.1
- Apache >= 2.4 o Nginx
- Mysql >= 5.7
- Oracle Instant Client 12g

Guia de despliegue

1. Clonar repositorio del proyecto

```
git clone https://github.com/apolo96/UHNotasAPI.git
```

2. Copiar archivo .env.example

```
cp .env.example ./.env
```

> Configuar variables de conexion a base de datos

***Recuperado de**: https://github.com/apolo96/UHNotasAPI*

Ilustración 37. GitHub requisitos y guía despliegue de la API número uno

5. Ejecutar migraciones y semillas (seeder)

```
php artisan migrate --seed
```

6. Generar llaves y crear clientes para Passport - Oauth2

```
php artisan passport:install
```

7. Crear cliente UHNotas.App en Passport

```
php artisan passport:client --password
```

> Copiar las credenciales CLIENT ID y CLIENT SECRET en las variables PASSPORT_CLIENT_ID y PASSPORT_CLIENT_SECRET respectivamente en el archivo .env

8. Configuración variables de entorno

En el archivo** .env** establecer la variable **APP_ENV** con el valor igual a **production** y **APP_DEBUG** a **false**

***Recuperado de**: https://github.com/apolo96/UHNotasAPI*

Ilustración 38. GitHub requisitos y guía para el despliegue de la API número dos

3. Instalar dependencias del proyecto

```
composer install --optimize-autoloader
```

4. Generar APP_KEY de Laravel

```
php artisan key:generate
```

5. Ejecutar migraciones y semillas (seeder)

```
php artisan migrate --seed
```

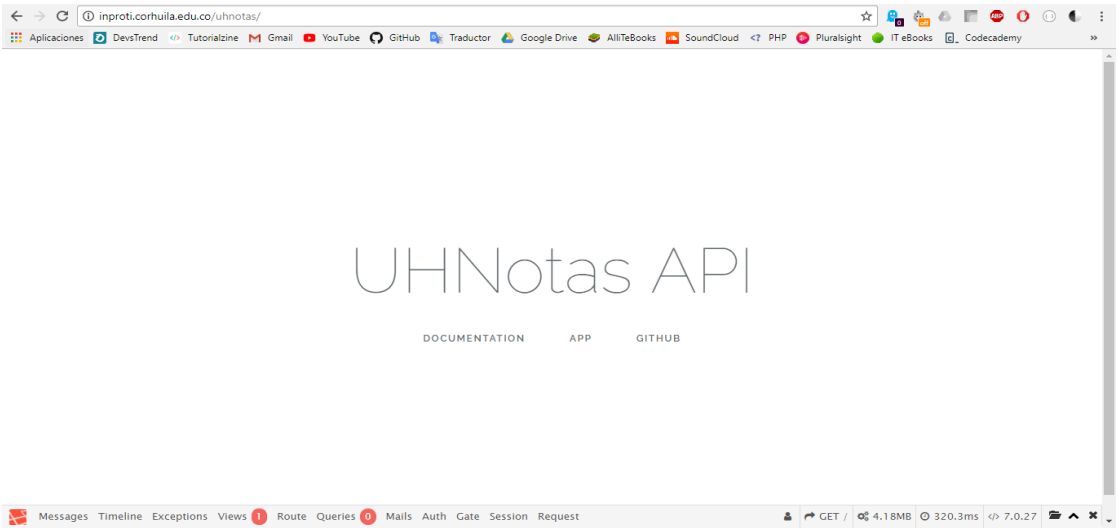
6. Generar llaves y crear clientes para Passport - Oauth2

```
php artisan passport:install
```

***Recuperado de**: https://github.com/apolo96/UHNotasAPI*

Ilustración 39. GitHub requisitos y guía para el despliegue de la API número tres

Una vez realizado todos los pasos procedemos a ingresar al servidor mediante el navegador y dirigirnos a la URL donde se muestra la página principal de las API.



Recuperado de: *http://inproti.corhuila.edu.co/uhnotas/*

Ilustración 40. Página Web presentación de UH Notas API

9. Resultados Esperados

9.1. Dirigido a la apropiación social del conocimiento

Con el desarrollo de este proyecto se pretende transferir el conocimiento a la comunidad educativa a través de los resultados obtenidos, con un artículo investigativo de revisión: Desarrollo de aplicaciones móviles con enfoque multiplataforma.

Tabla 25. Resultados esperados del proyecto

Resultado/Producto esperado	Indicador	Beneficiario
Artículo investigativo de revisión	Desarrollo de aplicaciones móviles con enfoque multiplataforma: Un revisión sistemática	Docentes y estudiantes de las diferentes Universidades de Colombia.

Nota fuente: *Polo, A y Estupiñán, S (2018)*

9.2. Impactos esperados a partir del uso de los resultados

El impacto tecnológico que se va a generar, va a beneficiar a la comunidad CORHUILA (estudiantes), porque permitirá mejorar los procesos académicos de la institución y el acceso a la información en tiempo real.

Tabla 26. Impactos esperados a partir del uso de los resultados

Impacto esperado	Plazo (años) después de finalizado el proyecto: corto (1-4), mediano (5-9), largo (10 o más)	Indicador verificable	Supuestos*
Beneficiar a toda la comunidad CORHUILA (estudiantes, docentes y directivos)	A corto plazo de 1 a 4 años	80% de la comunidad CORHUILA	Se supone que el 80% de la población objeto de estudio utiliza un dispositivo móvil con Sistema Operativo Android.

Nota fuente: *Polo, A y Estupiñán, S (2018)*

10. Cronograma

Tabla 27. Cronograma de actividades del proyecto

Actividad	Meses									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Actividades de autoaprendizaje Xamarin	**X**	**X**	**X**	**X**						
Actividades de aprendizaje colaborativo Xamarin		**X**	**X**	**X**						
Actividades de investigación asociadas al proyecto de grado	**X**	**X**	**X**	**X**						
Generar la elicitación de requisitos para el desarrollo de la App			**X**	**X**						
Definir el diseño y la arquitectura de la App				**X**	**X**					
Desarrollar la App con sus respectivas pruebas funcionales				**X**	**X**	**X**	**X**			
Generar la documentación del proyecto			**X**	**X**	**X**	**X**	**X**			
Generación del Artículo: Estado actual del desarrollo de aplicaciones móviles para el sector educativo en Colombia						**X**	**X**	**X**		
Capacitación: Introducción al desarrollo de aplicaciones móviles con Xamarin.							**X**	**X**		

Nota fuente: *Polo, A y Estupiñán, S (2018)*

11. Presupuesto

Tabla 28. Presupuesto del proyecto

	FUENTES		
RUBROS	**CORHUILA**	**CONTRAPARTIDA GRUPO DE INV.**	**TOTAL**
PERSONAL DE PLANTA	**No financiable**		
COMPRA DE EQUIPOS	0		
SOFTWARE	330.000		
MATERIALES E INSUMOS	120.000		
SALIDAS DE CAMPO	0		
MATERIAL BIBLIOGRÁFICO	**No financiable**		
SERVICIOS TÉCNICOS	0		
DIVULGACIÓN DE LA INVESTIGACIÓN	1.000.000		
CONSTRUCCIONES	**No financiable**		
OTROS	**No financiable**		
IMPREVISTOS	**No financiable**		
MANTENIMIENTO	**No financiable**		
TOTAL	**1.450.000**		

Nota fuente: Polo, A y Estupiñán, S (2018)

12. Conclusiones

Como resultado de la investigación se demuestra que las aplicaciones móviles "Apps" y los dispositivos móviles han tenido un crecimiento considerable en los últimos años, siendo el futuro y presente dentro de las Tecnologías de la Información y la Comunicación (TIC), es por esta razón, que a la hora de generar un proyecto para dar solución por medios tecnológicos las aplicaciones móviles pueden ser una buena elección frente a otras plataformas, teniendo en cuenta todos los beneficios que las aplicaciones y dispositivos móviles nos brindan.

Al trabajar en el proyecto se llevaron a la práctica todos los conocimientos y habilidades adquiridas durante la carrera de Ingeniería de Sistemas, de igual forma se obtuvo experiencia y adquisición de nuevo conocimiento en el desarrollo de Apps multiplataforma con Xamarin y API REST, así como sus características, arquitectura y funcionamiento.

Se desarrolló e implemento UH Notas API en los servidores del Grupo de Investigación Innovación, investigación y desarrollo de proyectos de tecnología de la información “INPROTI”, permitiendo compartir información académica con la aplicación UH Notas. Igualmente, la aplicación móvil está disponible para su implementación en otras plataformas o sistemas operativos, además la API puede servir para futuros proyectos que quiera realizar la comunidad estudiantil en CORHUILA.

Finalmente, el resultado del proyecto fue la creación de la aplicación móvil UH Notas que significo un aporte en el avance tecnológico de la institución. Así mismo con la creación de la App se logró un impacto tecnológico dentro de CORHUILA obteniendo como resultado beneficiar a más del 80% de la comunidad de estudiantes que utilizan activamente su teléfono inteligente.

Referencias

Amaya Balaguera, Y. D. (2013). Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. *Revista de Tecnología | Journal Technology*, *12 número*, 111–124.

Android, D. (2017). Android Platform. Retrieved from https://developer.android.com/guide/platform/index.html?hl=es-419

Angélica, B., Briceño, P., Consuelo, N., & Salazar, O. (2016). Desarrolló Agil de una Aplicación para Dispositivos Móviles. Caso de Estudio: Taxímetro Móvil Agile Application Development for Mobile Devices. Case Study: Mobile Taximeter, *21*(3), 260–275. https://doi.org/10.14483/udistrital.jour.reving.2016.3.a01

Apple. (2017a). App Store. Retrieved from https://developer.apple.com/support/app-store/

Apple. (2017b). Learn how to identify your iPhone model by its model number and other details. Retrieved from https://support.apple.com/en-us/HT201296

Avison, D., & Fitzgerald, G. (2006). *Information systems development*. McGraw-Hill Education.

Azure, M. (2018). API Design. Retrieved from https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

Betancur, L. (2014). Las universidades se apuntan a la era de las "apps." Retrieved from http://www.eltiempo.com/archivo/documento/CMS-13661855

Cantillo Valero, C., Roura Redondo, M., & Sánchez Palacín, A. (2012). Tendencias actuales en el uso de dispositivos móviles en educación. *La Educación Digital*, *47*, 1–21.

Cazau, P. (2006). Introducción a la Investigación en Ciencias Sociales, 74–80.

ComScore, A. C. (2015). *Futuro Digital Colombia 2015*. https://www.comscore.com/lat/Prensa-y-Eventos/Presentaciones-y-libros-blancos/2015/Futuro-Digital-Colombia-2015.

ComScore, Marchant, I., & Valencia, F. (2015). Medición Multiplataforma en Colombia.

Developer, A. (2017). Start Developing iOS Apps. Retrieved from https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/index.html#//apple_ref/doc/uid/TP40015214-CH2-SW1

Ditrendia. (2014). Informe ditrendia : Mobile en España y en el Mundo.

Ditrendia. (2015). Informe Mobile en España y en el Mundo 2015, 66. Retrieved from http://www.ditrendia.es/wp-content/uploads/2015/07/Ditrendia-Informe-Mobile-en-España-y-en-el-Mundo-2015.pdf

Ditrendia. (2016). Informe ditrendia 2016: Mobile en España y en el Mundo. *Ditrendia*, 67.

Ditrendia. (2017). Informe ditrendia Mobile en España y en el Mundo 2017 - DITRENDIA, 84. Retrieved from https://www.amic.media/media/files/file_352_1289.pdf

Durall, E., Gros, B., Maina, M., Johnson, L., & Adams, S. (2012). Perspectivas tecnológicas : educación superior en Iberoamérica 2012-2017, 27.

Fallis, A. . (2013). *REST API Design Rule Book*. *Journal of Chemical Information and Modeling* (Vol. 53). https://doi.org/10.1017/CBO9781107415324.004

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. *Building*, *54*, 162. https://doi.org/10.1.1.91.2433

Fowler, M. (2010). Richardson Maturity Model API REST. Retrieved from https://martinfowler.com/articles/richardsonMaturityModel.html

Gartner. (2017). Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. Retrieved from http://www.gartner.com/newsroom/id/3609817

Guía, L., & Scrum, D. De. (2016). La Guía de Scrum.

Gutiérrez-Rubí, A. (2014). 6 rasgos clave de los millennials, los nuevos consumidores. Retrieved from https://www.forbes.com.mx/6-rasgos-clave-de-los-millennials-los-nuevos-consumidores/

Harvard University. (2018). Harvard Mobile. Retrieved from https://play.google.com/store/apps/details?id=edu.harvard.harvardmobile

Inc, A. (2017). About the iOS Technologies. Retrieved from https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

Katz, Y. (2018). JSON API ESPECIFICATION. Retrieved from http://jsonapi.org

Kılıçdağı, A., & YILMAZ, H. İ. (2014). *Laravel Design Patterns and Best Practices*.

Leiva Mundaca, I., & Villalobos Abarca, M. (2015). Método ágil híbrido para desarrollar software en dispositivos móviles. *Ingeniare. Revista Chilena de Ingeniería*, *23*(3), 473–488. https://doi.org/10.4067/S0718-33052015000300016

Microsoft. (2017). What is UWP. Retrieved from https://docs.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp

Microsoft. (2018). Arquitectura. Retrieved from https://docs.microsoft.com/es-es/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/architecture

Microsoft Xamarin. (2016). Introducción al ciclo de vida de desarrollo de software móvil. Retrieved from https://docs.microsoft.com/es-es/xamarin/cross-platform/get-started/introduction-to-mobile-sdlc

Microsoft Xamarin. (2018). Análisis detallado de Xamarin.Forms. Retrieved from https://docs.microsoft.com/es-es/xamarin/xamarin-forms/get-started/hello-xamarin-forms/deepdive?tabs=vswin#architecture-and-application-fundamentals

Microsoft Xamarin Docs. (2017). Introducción al desarrollo móvil. Retrieved from https://docs.microsoft.com/es-es/xamarin/cross-platform/get-started/introduction-to-mobile-development

Ministerio de Educación. (2016). Boletín Educación Superior en Cifras, 1–3.

Mobile Marketing Association. (2011). *Libro blanco de las apps. La Asociación de Marketing Movil*. Retrieved from http://www.mmaspain.com/wp-content/uploads/2015/09/Libro-Blanco-Apps.pdf

OneSignal. (2018). Push Notifications. Retrieved from https://onesignal.com/what-are-push-notifications

Patricio Henríquez Ritchie, Javier Organista Sandoval, G. L. (2015). Nuevos procesos de interactividad e interacción social: uso de smartphones en estudiantes y docentes universitarios. Retrieved from https://revistas.ucr.ac.cr/index.php/aie/article/view/12039

Richter, F. (2017). The Smartphone Platform War Is Over. Retrieved from https://www.statista.com/chart/4112/smartphone-platform-market-share/

Sahni, V. (2015). Best Practices for Designing a Pragmatic RESTful API. Retrieved from https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api#requirements

Stanford University. (2018). STANFORD MOBILE. Retrieved from https://ucomm.stanford.edu/mobileapp/

Tenjo Galarza, J. (2012). Demanda por educación superior: Proyecciones hasta 2025. *Colección Análisis Institucional*, 93.

Universidad de La Sabana. (2017). UniSabana. Retrieved from https://play.google.com/store/apps/details?id=lux.unisabana.sabanaviveenti&hl=es_419

Universidad Pontificia Bolivariana. (2017). UPB Colombia. Retrieved from https://play.google.com/store/apps/details?id=co.edu.upb.movilapp&hl=es

University of Oxford IT Services. (2017). Mobile Oxford. Retrieved from https://play.google.com/store/apps/details?id=uk.ac.ox.it.mobileoxford&hl=es

Xamarin. (2018). Xamarin Platform. Retrieved from https://www.xamarin.com/platform

Xamarin MVVM. (2017). MVVM. Retrieved from https://developer.xamarin.com/guides/xamarin-forms/enterprise-application-

patterns/mvvm/
XAML, X. (2018). Xamarin XAML.