UNIVERSIDAD NACIONAL
DE
CATAMARCA

FACULTAD DE
TECNOLOGIA Y CIENCIAS
APLICADAS

Desarrollo de una aplicación móvil de test de Orientación Vocacional

Tesis de Grado para la obtención del Título de Ingeniero en Informática

Tesista
Segura, Andrea Valeria

Director
Dra. Lazarte, Ivanna

Co- Director
Lic. Poliche, Valeria

Contenido

Índice de Tablas

Índice de Ilustraciones

Agradecimientos

La finalización de cualquier proyecto emprendido sin lugar a dudas se da por un cumulo de acciones, de personas, situaciones y de circunstancias adversas que supimos sortear. Independientemente del tiempo que nos haya tomado llegar al final de ese proyecto. Creo firmemente en que la persistencia, la tolerancia, la templanza, la fortaleza y sobre todo la paciencia con uno mismo son los pilares para cumplir cualquier meta que nos propongamos. Hoy, al culminar mi formación de grado quiero agradecer a cada uno de mis profesores con los que tuve la posibilidad de compartir, creo que una de las vocaciones más nobles es la docencia. Por eso hoy los reconozco y les agradezco a cada uno de los que me brindó su apoyo. A los miembros del jurado, por su tiempo, su dedicación y por la claridad en sus correcciones, las que ayudaron a que mi trabajo sea mejor.

Hablando un poco sobre el acompañamiento que un tesista debe recibir durante el proceso de elaboración, puedo decir que reconozco en la directora del presente proyecto final la Dra. Ivanna Lazarte y de la Co - Directora la Lic. Valeria Poliche el compromiso de asumir esta responsabilidad aportando su tiempo, experiencia y paciencia en cada entrega que realicé.

Durante el tiempo que cursé, tuve algunos contratiempos típicos de cualquier alumno regular y puedo decir que siempre me sentí acompañada, siempre tuve una respuesta positiva por parte del departamento de Informática, de Sección Alumnos y como así también de la Secretaria Académica la Lic. Natalia Fernández. Hoy les agradezco porque sentirnos respaldados por parte de la institución que elegimos para formarnos nos da ese plus de saber que no estamos solos y de que no somos solo un número más dentro de la facultad.

Retomando un poco lo que dije al principio sobre que la finalización de un proyecto es también un cumulo de personas, durante el cursado de esta carrera encontré enormes personas con las que disfrute y padecí en algunos momentos nuestra elección, algunos ya se recibieron y otros estamos en eso. Les agradezco a todos los que alguna vez me brindaron su ayuda y me dieron aliento para no abandonar.

A mis amigos y amigas que me entendieron en este estado de “elaboración de tesis” y eliminaron la pregunta: ¿Para cuándo?

A mis padres Leticia y Víctor, fundamentalmente por no dejarme abandonar la carrera y por inculcarme desde siempre que la educación es imprescindible para el desarrollo de una persona.

A mi hermana Victoria, que escucho explicaciones de diferenciales, librerías, ondas sinusoidales, sistemas de tiempo real, etc. Y aunque no sabía de qué hablaba ella me atendía y me brindaba su tiempo.

Hoy finalizo mi carrera de grado, hoy obtengo mi título. Quiero agradecerle a Damiana, mi hija por entender lo importante que era esto para mí, valorando cada uno de mis esfuerzos.

Para finalizar no quería dejar de mencionar que no sería posible que yo, mujer de 33 años, madre, de clase media, laburante, hoy pueda recibir mi título de grado si la educación no fuera pública y gratuita. La Universidad Pública es un techo que nos cobija a los que no tenemos el camino tan allanado.

Resumen

La Universidad Nacional de Catamarca (UNCA) cuenta, entre sus secretarías, con la Secretaría de Bienestar Universitario y Asuntos Estudiantiles, la cual se encarga de ofrecer distintos servicios a los estudiantes como ser: alojamiento, alimentación, salud, deporte, ayuda económica y de trabajo, a través de becas u otros programas. También brinda asesoramiento pedagógico y de orientación vocacional por medio de la Dirección de Orientación Vocacional perteneciente a esta.

En la mencionada Dirección se realizan distintas acciones, entre ellas, un proceso de orientación vocacional en el que se brinda asesoramiento a alumnos de los últimos años del Nivel Secundario sobre cuales son aquellas carreras afines a su persona.

En el proceso de orientación vocacional no se utilizan herramientas tecnológicas que permitan sistematizar las pruebas, recabar y organizar la información sobre los alumnos que las realizan.

Por ello surge la necesidad de desarrollar una aplicación móvil para la Dirección de Orientación Vocacional que automatice el test de orientación vocacional y que sirva de instrumento para ayudar a los estudiantes del último año del Nivel Secundario a elegir una carrera, basada en información sobre sus gustos, intereses, habilidades y aptitudes. El propósito de contar con esta herramienta es, por un lado, promocionar las carreras que se dictan en las distintas unidades académicas de la UNCA y, por otro lado, ayudar a disminuir la deserción de los estudiantes de los diferentes programas académicos de la UNCA.

Palabras claves: Aplicación móvil, Orientación vocacional, Promoción de carreras

Capítulo 1:

Introducción

1.1. Introducción

Este Trabajo Final se encuentra enmarcado dentro del Proyecto de I+D “Desarrollo de una aplicación móvil de test de orientación vocacional para la Universidad Nacional de Catamarca", bajo la dirección de la Dra. Ivanna Lazarte y la co-dirección de la Lic. Valeria Póliche. Cuenta con aval institucional otorgado mediante Resolución Rectoral N° 494/2018. La línea de investigación del mencionado Proyecto está orientada al estudio del efecto que tiene la orientación vocacional en la elección de carrera, como así también la incidencia que tiene la elección de la carrera de acuerdo al resultado del test en el éxito escolar del alumno.

El presente Trabajo Final se enfoca en el desarrollo de una aplicación para dispositivos móviles que permita realizar un test de orientación vocacional a estudiantes del último año del Nivel Secundario. La aplicación tiene como finalidad servir como un instrumento para que el alumno tenga una orientación de cuáles son sus áreas ocupacionales[1] de preferencia a la hora de optar por una carrera y proveerle información sobre las carreras que se imparten en la Universidad Nacional de Catamarca (UNCA) relacionadas a dichas áreas.

Para el desarrollo de esta aplicación se eligió la plataforma Android [1], el cual es el sistema operativo predominante en el mercado de telefonía móvil, y tiene a Google como su referente y gran promotor. Este sistema operativo está basado en el núcleo Linux y fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas y también para relojes inteligentes, televisores y automóviles. En la actualidad existe una cantidad innumerable de distintos modelos de teléfonos que cuentan con el sistema operativo Android en el mercado de más de diez fabricantes.

1.2. Planteamiento del Problema

La deserción universitaria es uno de los grandes problemas que afecta a muchas universidades de América Latina [51] [52] [53] [54]. Las causas de la deserción son múltiples y variadas por lo que esta problemática debe analizarse teniendo en cuenta factores de índole personal, socioeconómico y académico, entre otros. Los factores académicos tienen que ver, entre otras causas, con la deficiente orientación vocacional, la insatisfacción con la calidad de la carrera, y la insuficiente preparación para ingresar a una carrera [54] [55] [57].

La orientación vocacional es reconocida como un instrumento de gran valor para la retención de estudiantes en los centros educativos, particularmente en la Universidad [51] [55] [56] [58] [59] [60]. La orientación vocacional es un proceso de ayuda para el estudiante, destinado a conseguir una comprensión adecuada de las distintas opciones profesionales que existen en el mundo de la educación, eligiendo aquella que cumpla con sus intereses y objetivos personales [59] [60]. Se considera que un proceso de orientación vocacional es satisfactorio cuando colabora para que el estudiante elija el camino a seguir lo más acertadamente posible, reduciendo frustraciones en las elecciones profesionales y vocacionales, y contribuyendo a alcanzar una mejor calidad de vida [61].

En el I Congreso Iberoamericano de Orientación, celebrado en Argentina en el año 2004, se propuso la participación activa que deben tener las Universidades para que las actividades

[1] Son conjuntos de carreras u ocupaciones afines, que comparten principios técnicos o científicos, o los ámbitos en los que se lleva a cabo el trabajo. Por ejemplo, las ocupaciones relacionadas con las ciencias médicas ó ciencias de la salud.

de orientación vocacional sean definitivamente incorporadas a las actividades normales y propias de las mismas. En este sentido, la UNCA, a través de la Dirección de Orientación Vocacional, ofrece dos procesos institucionales de Orientación y Reorientación Vocacional, cuya modalidad es grupal o individual (dependiendo de las características específicas de cada caso) donde se guía al estudiante hacia la carrera que más se identifique con su vocación.

Los procesos de orientación vocacional llevados a cabo tienen como fin generar un espacio de encuentro donde el estudiante pueda plantear sus dudas, incertidumbres, pensamientos y deseos. La duración estimada del proceso es de ocho encuentros (alrededor de un mes), con una frecuencia de dos reuniones semanales, de una hora de duración cada una. Están dirigidos a estudiantes del último año del Nivel Secundario (tanto de la capital, como del interior provincial) o para adultos que decidan iniciar o modificar un proyecto de vida asociado a la elección de una carrera. Se cuenta con profesionales psicopedagogas que brindan asesoramiento, por medio de charlas informativas sobre la vida universitaria, acompañando en visitas guiadas a Facultades y Escuelas de la UNCA.

Además, el equipo de trabajo de la Dirección de Orientación Vocacional participa en el Programa la UNCA+Cerca (promovido por la Secretaria de Extensión Universitaria), también brindan charlas informativas en escuelas que requieren dicha participación, pensando en la matrícula potencial de la universidad y una inserción institucional efectiva. También se trabaja en simultáneo con las escuelas secundarias del interior provincial, cuya agenda de trabajo incluye diferentes actividades académicas, de difusión de las carreras de la UNCA, deportivas y competencias, artísticas, entre otras. La relevancia de esta política del gobierno universitario reside en que se propicia el acercamiento y trabajo con cada comunidad no sólo para la difusión y acercamiento universidad-comunidad de cada localidad sino que también permite la iniciación de otras actividades de vinculación de la UNCA con su contexto provincial de inserción.

Actualmente, Dirección de Orientación Vocacional realiza el test de orientación vocacional mediante formularios impresos y en forma presencial en las instalaciones de dicha dependencia, lo que dificulta que los estudiantes del interior accedan a los beneficios de realizar un test vocacional.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar una aplicación móvil que permita obtener información relevante sobre los intereses y habilidades de los estudiantes del Nivel Secundario, mediante un test de orientación vocacional, y que sirva de herramienta para ayudarlos al momento de elegir una carrera.

1.3.2. Objetivos específicos

- Indagar sobre los antecedentes tecnológicos y temáticos vinculados a la Orientación Vocacional.
- Adquirir conocimientos sobre la utilización de herramientas y lenguajes de programación para el desarrollo de aplicaciones móviles para el Sistema Operativo Android.

- Desarrollar una aplicación móvil de test de orientación vocacional destinado a los estudiantes de Nivel Secundario.
- Proveer una herramienta a la Dirección de Orientación Vocacional para ser utilizada durante el proceso de orientación vocacional.
- Ayudar a la Secretaría de Bienestar Universitario y Asuntos Estudiantiles en la promoción de carreras de la UNCA.

1.4. Alcances y limitaciones

La aplicación móvil propuesta brindará la posibilidad de realizar un test de orientación vocacional al usuario y de acuerdo al resultado mostrará información sobre carreras de grado y pregrado dictadas en la UNCA que se relacionen con dicho resultado como así también la unidad académica en la que se dictan. No brinda otro tipo de información como ser planes de estudio, fechas de inscripción, etc.

La aplicación móvil se desarrollará para su utilización en la Dirección de Orientación Vocacional de la UNCA.

Cabe aclarar que esta aplicación es solo un primer paso en la búsqueda vocacional, por lo cual se aconsejará al usuario participar del Proceso de Orientación Vocacional que brinda la Dirección de Orientación Vocacional para afianzar su decisión en la elección de una carrera.

1.5. Justificación

La importancia del presente Trabajo reside en que la población destinataria son los estudiantes del Nivel Secundario de las escuelas pre-universitarias y de las escuelas dependientes del Ministerio de Educación, Ciencia y Tecnología de la Provincia, quienes son potenciales miembros de la comunidad universitaria.

Otra cuestión a destacar refiere a la problemática que se pretende abordar con la implementación de este Trabajo, ya que, como se mencionó anteriormente, la orientación vocacional es reconocida como un instrumento de gran valor para la retención estudiantil en el nivel universitario. La aplicación móvil permitirá que la amplia y heterogénea población destinataria tenga la posibilidad de descubrir cuáles son las áreas ocupacionales más afines y las carreras que ofrece la UNCA en dichas áreas.

Además, la realización de este Trabajo se fundamenta en su relevancia institucional y económica. En cuanto a su relevancia institucional, es importante destacar que los resultados que se obtengan de los test permitirán a las unidades académicas de la UNCA delinear líneas de trabajo y de articulación anticipando la población de potenciales ingresantes. En cuanto a su relevancia económica, la aplicación propuesta es de fácil acceso para los usuarios de dispositivos móviles y permitirá llegar a una vasta población estudiantil con cero/mínimo costo.

En relación a las aplicaciones para dispositivos móviles, la ventaja principal que se presenta es justamente su movilidad. La aplicación supone una forma más simple de acceder a la información para su lectura y respuesta de forma rápida y fácil, utilizando las ventajas de la movilidad, mediante la conexión a internet.

1.6. Metodología de trabajo

Para la realización del Trabajo Final se utilizaron dos metodologías, una para la investigación y otra para el desarrollo de la aplicación móvil.
La metodología de investigación aplicada es la denominada “Modelo de Investigación en Ingeniería del Software: Una propuesta de investigación tecnológica”, desarrollada en la Universidad San Buenaventura, Cali, Colombia. La misma consiste en tres grandes fases:

- **La investigación y desarrollo inicial:** Se delimita el área de trabajo investigativo que se busca desarrollar en la iniciativa.
- **La investigación aplicada:** es la etapa en donde se busca madurar la tecnología definida en la etapa inicial, trabajando en el desarrollo de la tecnología y en su aplicación en situaciones reales.
- **La transferencia:** es la etapa que busca amplificar el impacto de las nuevas tecnologías, prácticas y conocimientos, más allá de las empresas con las cuales se trabajó en el proceso de investigación aplicada. Para cumplir su objetivo, los grupos de investigación desarrollan otros tipos de actividades, tales como cursos, conferencias, y licenciamiento de la tecnología.

En la fase **La investigación aplicada** de la metodología de investigación, para el desarrollo de la aplicación, se empleó la “Metodología para el desarrollo de aplicaciones móviles” [9], desarrollada en la Universidad Distrital Francisco José de Caldas, Colombia, la cual rescata y aplica aspectos de las metodologías de desarrollo ágil, la evaluación del potencial de éxito para servicios de tercera generación denominada 6 M, y la ingeniería de software educativo con modelado orientado por objetos (ISE-OO).

Ambas metodologías se explican con más detalle en el Capítulo 2, en los incisos: 2.4.5 Metodología específica para el desarrollo de Aplicaciones Móviles y 2.5.1 Modelo de Investigación en Ingeniería del Software: Una propuesta de investigación tecnológica.

1.7. Resultados Esperados

Se espera obtener una aplicación móvil de test de orientación vocacional destinado a los estudiantes del Nivel Secundario, que sirva de herramienta a la Dirección de Orientación Vocacional para ayudar al estudiante al momento de elegir una carrera. También que la misma se utilice como herramienta para la promoción de las carreras de grado de la Universidad Nacional de Catamarca.

La aplicación móvil de test de orientación vocacional deberá permitir al usuario:

- Obtener información sobre las diferentes actividades que se realizan en la Dirección de Orientación Vocacional.
- Obtener información sobre los distintos programas de becas y otros servicios que ofrece la Secretaria de Bienestar Universitario y Asuntos Estudiantiles.
- Georeferenciar en un mapa las localizaciones de las distintas unidades de la UNCA.

También se espera que el uso de esta aplicación permita a la UNCA:

- Masificar la cantidad de estudiantes que puedan acceder al test, sobre todo de aquellos estudiantes que viven en el interior de la provincia y no pueden concurrir a la Dirección de Orientación Vocacional a realizar el test.
- Ayudar a reducir la deserción estudiantil, ya que diversos estudios han demostrado que un alumno que eligió su carrera considerando sus habilidades, gustos e intereses (información que se obtiene a través de los test de orientación vocacional), tendrá mayores posibilidades de éxito escolar.
- Delinear líneas de trabajo y de articulación anticipando la población de potenciales ingresantes, a partir de los resultados que se obtengan de los test.

1.8. Divulgación de resultados

Los resultados parciales del trabajo realizado han sido divulgados a través de las siguientes publicaciones:

- Segura, Andrea; Fernández, Gabriel; Doria, Vanesa; Flores, Carola; Moreno, Juan Pablo; Poliche, Valeria; Lazarte, Ivanna. (2018). *La UNCA al alcance de tu mano*. XVI Semana Nacional de la Ciencia, la Tecnología y el Arte Científico (Edición 2018). Facultad de Tecnología y Ciencias Aplicadas, UNCA, Catamarca, Argentina. 3 al 9 de Septiembre de 2018.
- Segura, Andrea; Fernández, Gabriel; Lazarte, Ivanna. (2018). *Aplicación móvil de test de orientación vocacional para la Universidad Nacional de Catamarca*. IV Jornadas Estudiantiles de Investigación e Innovación Tecnológica. Facultad de Tecnología y Ciencias Aplicadas, UNCA, Catamarca, Argentina. 05 de noviembre de 2018.
- Lazarte, Ivanna; Doria, Vanesa; Flores, Carola; Moreno, Juan Pablo; Fernández, Natalia; Poliche, Valeria. (2019). *Aplicación móvil de test de orientación vocacional para la Universidad Nacional de Catamarca como herramienta para ayudar a disminuir la deserción universitaria y promocionar las carreras*. 21° Edición del Workshop de Investigadores en Ciencias de la Computación (WICC). Facultad de Ciencias Exactas, Físicas y Naturales, UNSJ, San Juan, Argentina. 25 y 26 de abril de 2019.

1.9. Organización del Trabajo Final

El Capítulo 2 tiene por objetivo establecer el marco teórico de los conceptos usados a lo largo del presente Trabajo Final y presenta también los principales trabajos relacionados.

El Capítulo 3 se enfoca en el análisis y diseño de la aplicación móvil, según indica la Metodología para el desarrollo de aplicaciones móviles.

El Capítulo 4 se enfoca en el despliegue de la aplicación móvil, el cual comprende la codificación de la aplicación, la documentación del código y las pruebas de funcionamiento, según indica la Metodología para el desarrollo de aplicaciones móviles.

El Capítulo 5 presenta las conclusiones, las cuales resumen las acciones llevadas a cabo para el cumplimiento de los objetivos del Trabajo Final, destaca las principales contribuciones y describe los aspectos de las mismas que constituyen el punto de partida para el desarrollo de trabajos futuros.

Capítulo 2:

Marco Teórico

2.1. Introducción

Este capítulo abarcará aquellas temáticas que se hicieron necesarias profundizar para llevar a cabo el presente Trabajo Final.

En primer lugar, se describirá de qué se trata la orientación vocacional y los distintos test utilizados en los procesos de orientación vocacional, dado que no es la única técnica utilizada. Y para finalizar esta temática se mencionará una experiencia realizada en la Universidad de Buenos Aires sobre la utilización de herramientas tecnológicas en el proceso de orientación vocacional.

En segundo lugar, se describirán conceptos relacionados al desarrollo de aplicaciones móviles que son necesarios conocer a la hora de emprender un proyecto de desarrollo de estas características. Los mismos son: dispositivos móviles, sus características, los sistemas operativos utilizados en los mismos, lenguajes de programación que se utilizan para realizar desarrollo de aplicaciones móviles.

Por tercer lugar, se describen las metodologías de desarrollo de software que pueden aplicarse en el desarrollo de aplicaciones móviles. En estas últimas, se enlistan algunas de las metodologías agiles más usuales y la utilizada en el presente trabajo final.

Por último, se describirá la metodología de investigación aplicado para el desarrollo del Trabajo Final.

2.2. Orientación Vocacional

Durante el proceso de orientación vocacional se utilizan múltiples técnicas para poder brindar un acompañamiento al individuo (en este caso, el alumno) en la elección de una carrera. En los siguientes incisos se hace referencia a la utilización de tests, también llamados pruebas, como herramientas de apoyo en dicho proceso. También resulta importante investigar sobre la utilización de herramientas tecnológicas dentro del proceso de orientación vocacional considerando que en la Dirección de Orientación Vocacional no se aplica ninguna herramienta de este tipo. Esta investigación podrá aportar argumentos sólidos propios del área de la Orientación Vocacional a la hora de realizar entrevistas con el cliente, si es que este tuviera alguna duda sobre la implementación de una herramienta tecnológica en el proceso de orientación vocacional.

2.2.1. Universidades y procesos de Orientación Vocacional

En el I Congreso Iberoamericano de Orientación celebrado en Argentina (año 2004), se propuso la participación activa que deben tener las Universidades para que las actividades de Orientación Vocacional sean definitivamente incorporadas a las actividades normales y propias de las mismas. Esta propuesta se basa en el hecho de que hay que pensar en una Universidad distinta a la que hasta ahora hemos tenido. En este sentido se presentaron experiencias en las cuales algunas universidades han incorporado como actividades de Extensión, dentro de los servicios comunitarios, el asesoramiento vocacional-laboral a todos los miembros de la Comunidad, llegando incluso a desarrollar desde Centros de Información y Orientación hasta Oficinas de Empleo.

Asimismo, la Dra. Mirta Gavilán [5], Presidenta del Comité Organizador, destacó en sus palabras de apertura, la necesidad de que la Orientación, como disciplina, considere y tome

para sí otros saberes disciplinarios tales como los de la antropología, la política, la economía, la salud y la educación. Avalando de esta forma la concepción interdisciplinaria de esta disciplina.

Otra consideración importante fue la de que definitivamente se reconoce que la denominación de Orientación Vocacional no significa que sólo se aborden problemas relacionados con la selección de tipo de estudio o la ubicación en un trabajo determinado, sino que la Orientación Vocacional debe ocuparse por aspectos integrales de la vida de los ciudadanos así como contribuir al desarrollo económico y social de los países, y que aun cuando se reconoce la naturaleza global de los problemas, las soluciones no pueden ser las mismas en cada país.

Al igual que en el Congreso realizado en Suiza, se propuso la búsqueda de un nuevo término que sustituya al de Orientación Vocacional y que dé una idea más exacta de lo que se trata. Algunos ponentes sugirieron Orientación para la Transición o para la Elección. Ya la Orientación Vocacional no es para una elección de carrera, es una elección a lo largo de la vida. En algunos casos una elección para la soledad, o para un compartir.

La Orientación debe preparar a las personas para los períodos de transición en la vida de cada cual. Por ejemplo, la transición de estudiante a profesional, de empleado a jubilado, de empleado a desempleado, de empleado a subempleado, e incluso de soltero a casado, o de casado a divorciado.

Paralelamente, en el Congreso aludido, se reconoció el importante papel del docente en el desarrollo de la Orientación Educativa, tal como lo refleja una de las ponencias presentada por representantes de Brasil, donde expresaron que “todo educador es un orientador, pero lamentablemente no tiene la formación requerida y la mayoría de ellos sólo se preocupa por el cumplimiento de los objetivos previstos en sus programas y por la transmisión de conocimientos.” Esta afirmación, que fue compartida por los asistentes, supone la necesidad de una revisión del perfil que el profesional de la Orientación requiere para un mejor desempeño de su actividad.

2.2.2. La UNCA y la Orientación Vocacional

La UNCA viene realizando desde hace años, un proceso institucional de Orientación Mensual, Intensivo o Reorientación Vocacional, cuya modalidad es grupal o individual (dependiendo de las características específicas de cada caso), durante esta instancia de trabajo conjunto se guía al estudiante hacia la carrera que más se identifique con su vocación.

Los procesos de orientación llevados a cabo en el servicio tienen como fin generar un espacio de encuentro donde el estudiante pueda plantear sus dudas, incertidumbres, pensamientos y deseos. La duración estimada del proceso es de ocho encuentros (alrededor de un mes), con una frecuencia de dos reuniones semanales de una hora de duración cada una. Está dirigido a estudiantes del último año de secundaria (tanto de la capital, como del interior provincial) o para adultos que decidan iniciar o modificar un proyecto de vida asociado a la una elección de una carrera. Profesionales psicólogas brindan asesoramiento, por medio de charlas informativas sobre la vida universitaria, acompañando en visitas guiadas a facultades y escuelas de la UNCA, las cuales son convocadas por diferentes instituciones educativas. Además, el equipo de trabajo de la

Dirección de Orientación Vocacional participa en el Programa la Unca+Cerca (promovido por la Secretaria de Extensión Universitaria), así como también se trabaja a partir de la convocatoria de instituciones de la capital para brindar charlas informativas en escuelas que requieren dicha participación; pensando en la matrícula potencial de la universidad y una inserción institucional efectiva.

Se trabaja en simultáneo con las escuelas secundarias del interior provincial, cuya agenda de trabajo incluye diferentes actividades académicas, de difusión de las carreras de la UNCA, deportivas y competencias, artísticas, entre otras. La relevancia de esta política del gobierno universitario reside en que se propicia el acercamiento y trabajo con cada comunidad no sólo para la difusión y acercamiento universidad-comunidad de cada localidad; sino que también permite la iniciación de otras actividades de vinculación de la UNCA con su contexto provincial de inserción.

2.2.3. Utilización de test en el proceso de orientación vocacional

Como primera instancia se ponen en claro conceptos propios de la psicología que atañen a la temática planteada en este inciso, dado que no es el área de estudio para la carrera Ingeniería en Informática.
Las pruebas o test vocacionales se componen por dos términos: prueba/test y Orientación Vocacional; los cuales se definirán a continuación. Galtón [37] define una prueba como:

> *"Tarea o labor que es pedida a uno o más individuos con el fin de averiguar del modo más sencillo y rápido el grado que posee una determinada habilidad reaccional". Galtón, (1869)*

De acuerdo a esta definición se puede decir entonces que un test es una tarea que un individuo realiza con el fin de obtener de forma rápida y sencilla información sobre una o varias aptitudes.
Tyler [36] define el término vocación como:

> *"la disposición particular de cada individuo para elegir la profesión u oficio que desee estudiar y ejercer de acuerdo a sus aptitudes, características socioeconómicas y culturales" Tyler, (1975) (p.215).*

La definición de test vocacional emitida por Mira y López [35] dice que:

> *"prueba que permite registrar las reacciones personales, para estudiar sus modalidades, intensificando y formulando un diagnóstico que tiende a pronosticar un empleo en el futuro" (p.494). Mira y López*

Se pueden observar ciertos elementos en común y complementarios en estas tres definiciones por lo que se concluye que los test son instrumentos capaces de identificar y canalizar aptitudes, destrezas y habilidades cognoscitivas, psicológicas y manuales del individuo que aportan al orientador información importante para orientar al individuo, tomando en cuenta sus aptitudes, labores futuras y gustos intelectuales.

Por lo cual los test vocacionales se consideran importantes en el proceso de orientación vocacional ya que estos permiten conocer y reafirmar aptitudes, habilidades y destrezas de los individuos evaluados permitiendo que el orientador tenga una visión clara de la imagen de los diferentes aspectos de la persona. Además, ayuda a descubrir sus posibilidades de desarrollo y a elegir entre lo que se considera conveniente para él.
Algunas consideraciones importantes que deben tenerse en cuenta a la hora de realizar un test son: el factor tiempo, el contenido, cuándo y cómo se deben aplicar, la disposición en que se contesta, entre otras que de una u otra forma intervienen en la elección de carrera.

Los investigadores utilizan distintos tipos de prueba para analizar clases variadas de habilidades, cada una de ellas con distintos grados de complejidad y orientación [35]. Tyler [36] propone la siguiente clasificación:

- **Pruebas de personalidad:** A través de éstas se registran las reacciones personales, para estudiar modalidades e intensificar y formular un diagnóstico que tiende a pronosticar, dentro de ciertas limitantes de probabilidad, el comportamiento individual.

- **Pruebas de habilidad:** Éstas se subdividen en pruebas de logro y pruebas de aptitud.

  - **Pruebas de logro:** Sirven para distinguir las habilidades desarrolladas.

  - **Pruebas de aptitud:** Sirven para distinguir las capacidades de desarrollo.

- **Pruebas de indicador:** Sirven para indicar el desarrollo del potencial. En este contexto se pueden conocer mediante el registro de casos y entrevista. Ejemplo: la Srta. "X", que califica en el percentil 88 sobre la prueba, nunca ha tenido la experiencia que le ayude a desarrollar esta habilidad y la elevada calificación de su prueba señala una importante capacidad que puede desarrollar, si ella quiere.

- **Pruebas de inteligencia:** Estos tipos de prueba se emplean en centros de orientación; se utilizan distintas pruebas, como por ejemplo, ítems de analogías verbales, las cuales consisten en completar una frase relacionando conceptos. Por ejemplo: “... es a libro como pintor es a ...”

  1. Capítulo 2. Escritor 3. Lectura 4. Palabra 5. Literatura

  A. Pintura B. Rembrandt C. Brocha D. Taller E. Pintar

  La respuesta es Escritor es a libro como pintor es a pintura.

- **Pruebas individuales:** Contribuyen a la evaluación que un individuo debe hacer de sí mismo.

- **Pruebas de eficiencia o rendimiento global**: Estas pruebas miden las aptitudes específicas, los conocimientos y entrenamientos profesionales. Se usan

exclusivamente con la finalidad de selección en los campos de trabajo.

Otro tipo principal de prueba empleado extensamente es el que define los intereses vocacionales, también llamado prueba vocacional. Tyler [36] define puntualmente a estas como:

> *"Ahorro del tiempo, ganancia de objetividad que proporciona anticipación y rapidez, de esa manera podremos evitar que los individuos lleguen al fracaso" Tyler (1985) (p.215).*

Las pruebas vocacionales se deben aplicar al finalizar la educación secundaria, ya que en esta etapa los intereses y aptitudes del individuo están aptas para definir alternativas de carreras a las que podrá tener acceso, es decir el joven cuenta con la madurez e interés suficiente para realizar una prueba de este tipo. El individuo probablemente no escogerá definitivamente una carrera, pero el resultado le servirá de guía y reflexión en el proceso de selección de una carrera.
Un estudiante satisfecho que desea terminar su carrera para ejercerla, manifiesta interés y destrezas para esto, así como entusiasmo y confianza; en contraste, un estudiante insatisfecho mostrará indiferencia, agresividad y descontento por las actividades que realizará durante la realización del test. [34]

2.2.4. Utilización de herramientas tecnológicas en el proceso de orientación vocacional

En la Universidad Nacional de Buenos Aires se pusieron en marcha diversas propuestas de trabajo por parte de las investigadoras Quattrocci, Paula Raquel; García, Alejandra Elisa; Schittner, J. Vanesa de la Dirección Técnica de Programa de Orientación. Estas propuestas se presentaron en el Congreso Iberoamericano de Ciencia, Tecnología, Innovación y Educación, en Noviembre de 2014 [38] y surgieron a partir del siguiente interrogante: ¿Por qué la Orientación Vocacional debería reformular su campo disciplinar incorporando TIC en sus prácticas de formación a formadores?
Este interrogante se utilizó como disparador de esta experiencia aplicada a docentes orientadores. Las TIC adquirieron un rol principal ya que las actividades estaban pensadas para la construcción a lo largo del curso de un proyecto de intervención "real" a implementar en la institución escolar en la cual trabajan.
Esta tarea fue llevada a cabo con la dinámica "Work In Progress", donde las TIC tomaron un protagonismo brindando todo su potencial a esta experiencia que propone, entre sus objetivos, transferir competencias cognitivas mediatizadas por TIC, optimizando la formación en una doble vertiente, "aprender a hacer" orientación usando TIC, aprender a utilizar TIC para orientación.
Es aquí donde se ubican las coordenadas desde las cuales los orientadores ponen en marcha diversas propuestas de trabajo tanto con los jóvenes en formación, como en formación de formadores, docentes y orientadores.

Pensar la formación docente contemporánea sin vincularla con los procesos de construcción de conocimiento y construcción de elecciones de vida, son cuestiones que no pueden separarse de los dos bastiones emergentes de la sociedad actual. Uno es el desarrollo con las TIC o TIC para el desarrollo [41] y el otro la necesidad de adquirir competencias cognitivas que viabilicen la inclusión de todos los sectores sociales.

A partir de las premisas enunciadas, la experiencia vierte varios ítems para abordar, a saber:

- Minimizar la brecha digital pedagógica de los docentes.
- Crear o incrementar competencias cognitivas mediatizadas por TIC optimizando la formación en una doble vertiente, “aprender a hacer” orientación usando TIC, aprender a utilizar TIC para orientación.
- Mitigar la actitud refractaria de los profesionales del campo educativo a incorporar y utilizar TIC en la práctica cotidiana.
- Optimizar el uso de las herramientas tecnológicas de manera transversal en pos de mejorar la práctica como profesionales.
- Actualizar las prácticas a los contextos y necesidades contemporáneas.

Otra cuestión a tener en cuenta en por qué insertar las TIC en el área de la Orientación Vocacional es el incremento de la calidad de los servicios: facilitar nuevas perspectivas y oportunidades dirigidas a responder a necesidades cada vez más individualizadas y diversificadas, así como fortalecer el papel de los técnicos en Orientación. [39]
Las posibilidades de los recursos tecnológicos en orientación son múltiples, y plausibles de adecuarse a los objetivos que se pretendan alcanzar con cada acción a desarrollar. [40]
Para ello, es necesario adaptar y actualizar los contenidos de formación de orientadores con TIC, acordes a las demandas contextuales. Algunos de los objetivos básicos que los orientadores requieren alcanzar son:

- Revertir las resistencias y percepciones negativas hacia las TIC y transformarlas en positivas.
- Promover la incorporación de TIC en sus prácticas cotidianas.
- Desarrollar las competencias instrumentales y conceptuales necesarias para utilizar las mismas.

2.3. Aplicaciones Móviles

En esta sección se definirán conceptos relacionados a las aplicaciones móviles, su desarrollo, los dispositivos donde se utilizan, qué sistemas operativos utilizan dichos dispositivos y, por último, se realizará una descripción del sistema operativo para el que se va a desarrollar la aplicación móvil de test para orientación vocacional.
Una aplicación móvil o app (acortamiento del inglés application), es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Las aplicaciones móviles permiten al usuario efectuar un conjunto de tareas de cualquier tipo, según la función para la que fue diseñada; puede ser utilizada en el

ámbito profesional, de entretenimiento, educativas, de acceso a servicios, etc. Su tarea es facilitar las gestiones o actividades a desarrollar.
Se puede acceder a las aplicaciones móviles por medio de plataformas de distribución de las distintas compañías propietarias de los sistemas operativos móviles. Por ejemplo, las aplicaciones para el Sistemas Operativo Móvil Android se pueden encontrar en Play Store. Dentro de estas tiendas las aplicaciones móviles pueden ser gratuitas o pagas, donde en promedio el 20 a 30 % del coste de la aplicación se destina al distribuidor y el resto es para el desarrollador.
El desarrollo de aplicaciones móviles es el proceso por el cual se desarrolla un software para dispositivos móviles. La forma de distribución de estas aplicaciones puede variar: las aplicaciones pueden venir preinstaladas en los teléfonos o pueden ser descargadas por los usuarios desde app stores (tiendas de aplicaciones) como se mencionó en el párrafo anterior.

2.3.1. Características de las Aplicaciones Móviles

Un teléfono inteligente o SmartPhone es un teléfono móvil con mayor capacidad de cómputo y almacenamiento que un teléfono convencional [2]. En la gran mayoría de los teléfonos inteligentes, sus funciones no sólo se limitan a llamar o enviar y recibir mensajes de texto, sino también permite navegar por internet, sacar fotos y almacenarlas, escuchar música y ver videos. Dado que estos dispositivos móviles permiten hacer tareas que antes solo eran ejecutadas por computadoras de escritorio, surge la necesidad de desarrollar aplicaciones que permitan realizar tareas específicas, como compartir material en redes sociales, sincronizar datos entre diferentes dispositivos, enviar emails, etc.

Una aplicación móvil es una pieza de software diseñada para ser ejecutada por dispositivos móviles, como teléfonos o tablets [2]. Por lo general, estas aplicaciones, se encuentran disponibles a través de plataformas de distribución, operadas por las compañías propietarias de los sistemas operativos móviles, como Android, iOS, BlackBerry OS, Windows Phone, entre otros.

Los desarrolladores de aplicaciones móviles se enfrentan con problemas que tienen que ver con memoria insuficiente y capacidades de almacenamiento de datos, dispositivos lentos, capacidades online/offline y en los últimos años, los siempre evolutivos dispositivos con sistemas operativos y ambientes de desarrollo asociados. Todos estos problemas y complicaciones requieren de una rápida, efectiva y eficiente respuesta a los cambios de requerimientos, testeo, capacidades del dispositivo, entre otras cosas. Es por esto que un enfoque ágil es perfecto para el desarrollo de este tipo de aplicaciones, que requieren de una entrega rápida.

Todos los dispositivos que cuentan con la plataforma Android tienen embebido como Sistema Gestor de Base de Datos de código abierto SQLite. Éste tiene soporte a Bases de Datos relacional a través de sintaxis similares a SQL [1].

Para el desarrollo de la aplicación móvil se hará uso de la herramienta Android Studio [3], el cual es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones

para Android y se basa en IntelliJ IDEA[2]. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan la productividad durante la compilación de la aplicación para Android, como las siguientes:

- Un sistema de compilación basado en Gradle[3] flexible
- Un emulador rápido con varias funciones
- Un entorno unificado en el que se puede realizar desarrollos para todos los dispositivos que utilicen Android
- Gran cantidad de herramientas y frameworks de prueba
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.

Los lenguajes de programación que se utilizarán serán Kotlin y el Lenguaje de Marcado Extensible (XML).
Kotlin es un lenguaje de programación de tipo estático OSS (Sistemas de soporte a las operaciones) que se dirige a la JVM, Android, JavaScript y Native. Es desarrollado por JetBrains. El proyecto comenzó en 2010 y fue de código abierto desde muy temprano. El primer lanzamiento oficial 1.0 fue en febrero 2016. Kotlin es gratuito, se desarrolla bajo la licencia Apache 2.0 y el código fuente está disponible en GitHub. Este lenguaje tiene construcciones tanto orientadas a objetos como funcionales. Puede ser utilizado en los estilos Orientado a Objetos y en Programación Funcional. Kotlin es más conciso. Las estimaciones aproximadas indican un recorte del 40% en la cantidad de líneas de código en comparación a otros lenguajes de programación. También es más seguro, si hablamos de la compatibilidad con tipos de datos que no admiten nulos, hace que las aplicaciones sean menos propensas a los Null Pointer Exeption, excepción que surge cuando un objeto no está inicializado y no puede utilizar ningún método.
Es totalmente interoperable con el lenguaje de programación Java y se ha puesto gran énfasis en asegurar que su base de código existente puede interactuar correctamente con Kotlin. Se puede llamar fácilmente al código de Kotlin desde el código de Java y viceversa.
Es compatible con todos los principales IDE de Java, incluidos IntelliJ IDEA, Android Studio, Eclipse y NetBeans. Además, existe un compilador en línea que proporciona soporte directo para compilar y ejecutar aplicaciones [6].
La elección de este lenguaje se hace desde la perspectiva de innovar en la utilización de un lenguaje no convencional en el desarrollo de aplicaciones móviles como lo viene siendo Java.
XML [4] es un lenguaje que utiliza etiquetas auto descriptivas para agregar una capa de inteligencia a la información, es decir mediante estas es posible describir la naturaleza de la

[2] Ambiente de desarrollo integrado (IDE) para el desarrollo de programas informáticos. Es desarrollado por JetBrains (anteriormente conocido como IntelliJ), y está disponible en dos ediciones: community edition y edición comercial.

3 Sistema de compilación que se ejecuta en una herramienta integrada desde el menú de Android Studio, y lo hace independientemente de la línea de comandos. Puede personalizar, configurar y extender el proceso de compilación; crear múltiples APK para tu app, con diferentes funciones utilizando el mismo proyecto y los mismos módulos.

misma. Sin esta descripción la información tiene poco significado fuera del entorno donde se la utiliza. Es un metalenguaje que permite crear un lenguaje propio de etiquetas para múltiples tipos de documentos. Su función primordial es permitir que dos aplicaciones puedan comunicarse sin la necesidad de intervención humana, es decir que cualquier programa pueda interpretar correctamente los tipos de datos recibidos desde otro programa.

2.3.2. Dispositivos Móviles

Si se piensa en dispositivos móviles, lo primero que viene a la cabeza es un teléfono móvil, pero en realidad existe una gran cantidad de dispositivos electrónicos que se clasifican actualmente como tales, desde teléfonos hasta tabletas.
En la mayoría de los casos, un dispositivo móvil puede definirse con cuatro características que lo diferencian de otros dispositivos que, aunque pudieran parecer similares, carecen de algunas de las características de los verdaderos dispositivos móviles. Estas cuatro características son:

- **Movilidad:** es la cualidad de un dispositivo para ser transportado o movido con frecuencia y facilidad. Por tanto, el concepto de movilidad es una característica básica. Los dispositivos móviles son aquellos que son lo suficientemente pequeños como para ser transportados y utilizados durante su transporte.

- **Tamaño reducido:** es la cualidad de un dispositivo móvil de ser fácilmente usado con una o dos manos sin necesidad de ninguna ayuda o soporte externo. El tamaño reducido también permite transportar el dispositivo cómodamente por parte de una persona.

- **Comunicación inalámbrica:** es la capacidad que tiene un dispositivo de enviar o recibir datos sin la necesidad de un enlace cableado.

- **Interacción con las personas:** Se entiende por interacción al proceso de uso que establece un usuario con un dispositivo. Entre otros factores, en el diseño de la interacción intervienen disciplinas como la usabilidad y la ergonomía [2].

2.3.3. Sistemas Operativos para Dispositivos Móviles

La industria de los dispositivos y las aplicaciones móviles es un entorno en constante cambio. En referencia a los sistemas operativos (SO) desarrollados para éstos, se puede observar a lo largo de su historia grandes migraciones desde un SO a otro, protagonizadas por grandes empresas.

Symbian

Symbian es el SO que en sus inicios utilizaba Nokia, luego a partir del año 2010, comienza a utilizar MeeGo. Finalmente, en el año 2016, Symbian dejó de

Ilustración 1. Logo Sistema Operativo Symbian

tener soporte al no poder seguir compitiendo en el mercado de los nuevos smartphones con SO de última generación como Android, iOS o Windows Phone.

Android

Otro de los grandes movimientos es el gran crecimiento exponencial de Android, que ha pasado por delante del iOS de Apple y de BlackBerry. Android es el SO móvil más utilizado del mundo, con una cuota de mercado superior al 80% al año 2017, muy por encima de IOS [22]. Según el ingeniero de software Hiroshi Lockheimer: "La plataforma Android permite a los fabricantes de dispositivos competir e innovar. Los desarrolladores de aplicaciones pueden llegar a grandes audiencias y crear negocios sólidos. Los consumidores ahora tienen opciones de dispositivos sin precedentes, a precios cada vez más bajos" [24].

Ilustración 2.Logo Sistema Operativo Android.

El SO Android, es el conjunto de programas básicos que se utilizan los dispositivos móviles con pantalla táctil, entendiendo como dispositivos a smartphones, relojes inteligentes, tablet, televisores y automóviles. Es de código abierto, gratuito y no requiere del pago de licencias. Incluye además el middleware y aplicaciones básicas.
Pertenece a Google Inc. ya que este compró Android Inc. en el 2005.
Se basa en una versión modificada del kernel de Linux [4].
El Proyecto de Código Abierto de Android por Google, AOSP por sus siglas en inglés es quien se encarga del mantenimiento y desarrollo de Android. Nace con la idea de mantener el control del desarrollo de las diferentes adaptaciones que se realizan de este sistema operativo en los dispositivos móviles. Cada fabricante añade sus propias capas de personalización a los archivos AOSP, así como también los propios usuarios crean ROMs a partir de estos mismos archivos [25].
El Android Market es la tienda en línea gestionada por Google en la cual también se pueden descargar apps de sitios de terceras partes. Los desarrolladores programan principalmente en el lenguaje Java y controlan el dispositivo mediante librerías Java desarrolladas por Google.
Utiliza SQLite para el almacenamiento de datos, un motor de bases de datos relacionales de gran alcance y ligero disponible para todas las aplicaciones.
La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución.
Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic [4].
Uno de los elementos clave de Android es la máquina virtual de Dalvik. Todo el hardware de Android y acceso a los servicios del sistema se gestiona mediante Dalvik como un nivel intermedio. Mediante el uso de una máquina virtual para organizar la ejecución de aplicaciones, los desarrolladores tienen una capa de abstracción que asegura que nunca tendrán que preocuparse de una aplicación de hardware en particular [26].

En la Ilustración 3 se puede observar la arquitectura del SO Android.

Debido a que se va a utilizar este sistema operativo para la aplicación móvil propuesta, se va a describir su arquitectura y las últimas versiones del mismo.

A continuación se describen brevemente los componentes de la arquitectura:

- Bibliotecas: incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema.
- Runtime de Android: incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik.
- Núcleo Linux: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. También actúa como capa de abstracción entre el hardware y el resto de la pila de software.

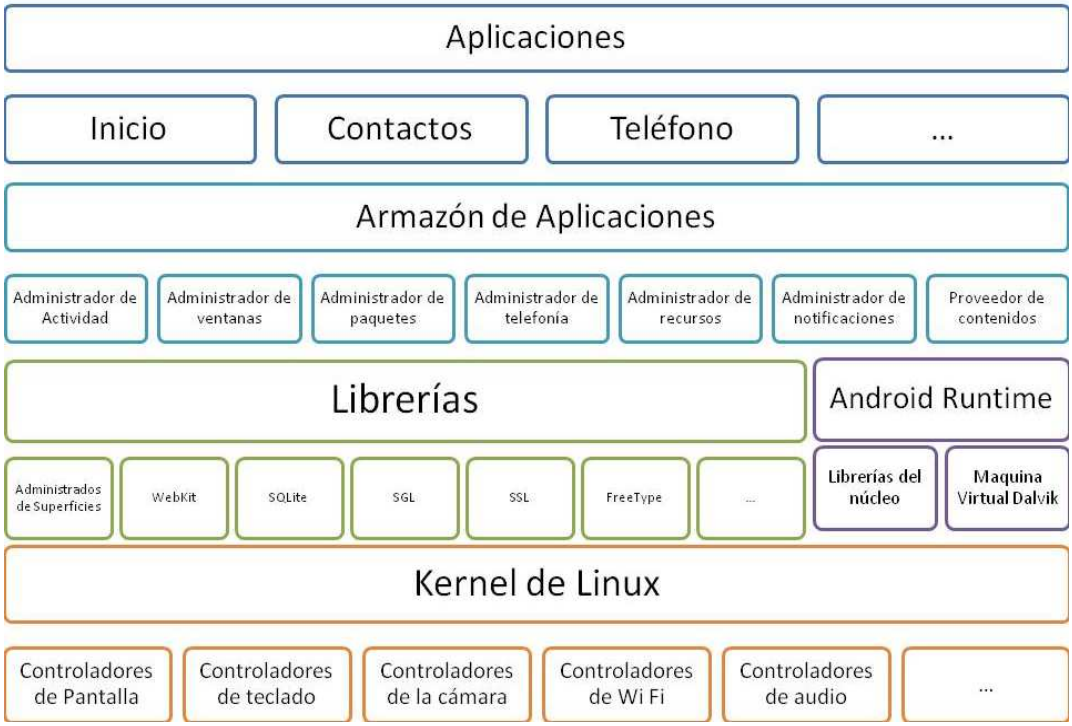


Ilustración 3. Arquitectura Sistema Operativo Android.

- Aplicaciones: incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas ellas escritas en Java. Marco de trabajo de aplicaciones: los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades [27].

En cuanto a las versiones de este SO, cada una de ellas introduce mejoras y nuevas funcionalidades. Se corrigen bugs y errores detectados. Se identifican por una numeración y sus nombres están basados en deliciosos postres. A cada versión generalmente se le agregan modificaciones, surgiendo pequeñas actualizaciones menores.

En la Tabla 1. Versiones de Sistema Operativo Android se describen los nombres, versiones y fecha de lanzamiento de cada versión.

Nombre	Versión	Lanzamiento
Apple Pie	1.0	23/09/08
Banana Bread	1.1	09/02/09

Cupcake	1.5	25/04/09
Donut	1.6	15/09/09
Eclair	2.1	26/10/09
Froyo	2.2–2.2.3	20/05/10
Gingerbread	2.3–2.3.7	06/12/10
Honeycomb	3.0–3.2.6	22/02/11
Ice Cream Sandwich	4.0–4.0.5	18/10/11
Jelly Bean	4.1–4.3.1	09/07/12
KitKat	4.4 4.4.4, 4.4W 4.4W.2	31/10/13
Lollipop	5.0–5.1.1	12/11/13
Marshmallow	6.0–6.0.1	5/10/15
Nougat	7.0 - 7.1 - 7.1.1 - 7.1.2	15/06/16
Oreo	8.0 - 8.1	21/08/17
Pie	9.0	27/01/18

Tabla 1. Versiones de Sistema Operativo Android

A continuación, se describen las últimas versiones.

Android Lollipop, en esta versión llega la pantalla de bloqueo, donde se podían mostrar las notificaciones. Otro cambio de parecer de Google es con respecto al acceso al almacenamiento externo (como tarjetas microSD), que vuelve a ser posible.

Ilustración 4. Logo Android Versión Lollipop.

Se reemplaza oficialmente a Dalvik por una compilación de aplicaciones AOT, de compilación anticipada. Se realiza una serie de cambios como el modo de ahorro de energía y programación de tareas para que se ejecuten solo con WiFi para ahorrar batería al reducir el uso de los datos móviles.

Ilustración 5. Logo Android versión Marshmallow.

Android Marshallow, modifica el acceso a los permisos. Quita el absolutismo de permisos de todo o nada con los permisos en tiempo de ejecución, las aplicaciones pueden pedir permiso para usar cierta función (cámara, micrófono, etc.) solo cuando lo necesitan, y no cuando se instalan. Utiliza in Modo Doze el cual se ocupa de obligar a las aplicaciones a “dormir” y reduce la velocidad de la CPU cuando la pantalla está apagada, de esta manera se logra una mayor duración de la batería.

Android Nougat, mejora el Doze de Marshmallow, haciéndolo efectivo incluso cuando el teléfono está en movimiento. Permite que aplicaciones de terceros añadan botones a los ajustes rápidos. Llega también Unicode 9.0 y los emojis con distintos tonos de piel, la calibración de color para la pantalla, las actualizaciones del sistema seamless, el nuevo modo de ahorro de datos y la posibilidad de elegir varios idiomas conocidos.

Ilustración 6. Logo Android versión Nougat.

Android Oreo, a partir de esta versión, el sistema limita más los procesos en segundo plano, ahorrando por consiguiente mucha batería. Las notificaciones se pueden ordenar según categorías (noticias, tecnología, etc.) y se pueden definir la frecuencia con se reciben. También al deslizar la notificación hacia la izquierda aparece en la parte de los ajustes para controlar las notificaciones de la aplicación específica, un reloj con el que se puede definir un periodo de tiempo en el que la notificación volverá a aparecer [27].

que

Ilustración 7. Logo Android versión Oreo.

Ilustración 8. Logo Android versión P.

Android Pie, es la novena y última versión principal y la decimosexta versión del sistema operativo Android. Primero fue anunciado por Google el 7 de marzo de 2018, y la primera vista previa de desarrollador fue lanzada el mismo día. La versión beta final de Android P se lanzó el 25 de julio de 2018. Incluye tecnología de inteligencia artificial para adaptarse al usuario. El SO aprende de los hábitos del usuario y es capaz de predecir las acciones que va a realizar. Funciones como Batería adaptativa, Brillo adaptable y App actions (Acciones de aplicación), son posibles gracias a esta Inteligencia artificial. Las mismas se describen a continuación:

- Batería adaptativa: esta función utiliza el aprendizaje automático para predecir qué aplicaciones usarás en las próximas horas y las que probablemente no usarás, por lo que tu teléfono solo gasta la batería en las aplicaciones que te interesan.
- Brillo adaptativo: con Brillo adaptativo, su teléfono aprende cómo configurar el brillo de su pantalla en diferentes entornos de iluminación y lo hace automáticamente con el tiempo.
- App Actions: con esta función el SO predice lo que el usuario querrá hacer a continuación según su contexto y muestra esa acción directamente en el smarthphone. Es decir predice "Acciones", que el usuario está a punto de realizar según el uso de su teléfono inteligente y la hora del día. Por ejemplo, en la mañana a su hora habitual de viaje al trabajo, las App actions pueden sugerir la navegación en Google Maps, mientras que al conectar los auriculares puede sugerir una lista de reproducción de Spotify o llamar a uno de los contactos.

Otras características para resaltar son:

- Compatibilidad con cámaras externas: admite cámaras USB / UVC externas en ciertos dispositivos.
- Múltiples conexiones Bluetooth: puede conectar hasta cinco dispositivos Bluetooth y cambiar entre estos dispositivos sin problemas.
- Confirmación protegida de Android: en el hardware compatible, las aplicaciones ahora pueden utilizar la IU controlada por el hardware de seguridad para obtener la confirmación de una transacción confidencial, como realizar un pago.
- Mejoras en la privacidad: el acceso al micrófono, la cámara u otros sensores del SmartPhone del usuario están deshabilitados cuando una aplicación esté inactiva o se esté ejecutando en segundo plano [62].

iOS

iOS es un sistema operativo móvil de la multinacional Apple Inc. Originalmente desarrollado para el iPhone (iPhone OS), después se ha usado en dispositivos como el iPod touch y el iPad. No permite la instalación de iOS en hardware de terceros.

Actualmente es el segundo SO móvil más utilizado del mundo, detrás de Android, con una cuota de mercado de entre 10-15% al año 2017. La última versión del sistema operativo es el iOS 11, aparecida en el mes de septiembre del 2017, disponible en dispositivos con

Ilustración 9. Logo Sistema Operativo iOS.

procesadores 64-bits [23].

En los últimos tiempos se ha producido una migración en las preferencias de los desarrolladores, que los ha movido desde Symbian, BlackBerry y Java hacia otras propuestas: iOS y Android. Según algunos estudios, cerca del 60% de los desarrolladores han desarrollado aplicaciones para Android. El iOS de Apple ocupa el segundo lugar (con más del 50%), seguido por Java ME, que se encuentra en tercera posición. En la Ilustración 10. Plataformas más utilizadas por los desarrolladores de aplicaciones móviles. Se puede ver cómo los desarrolladores cambian el foco de su atención hacia unas plataformas y abandonan otras. [4]

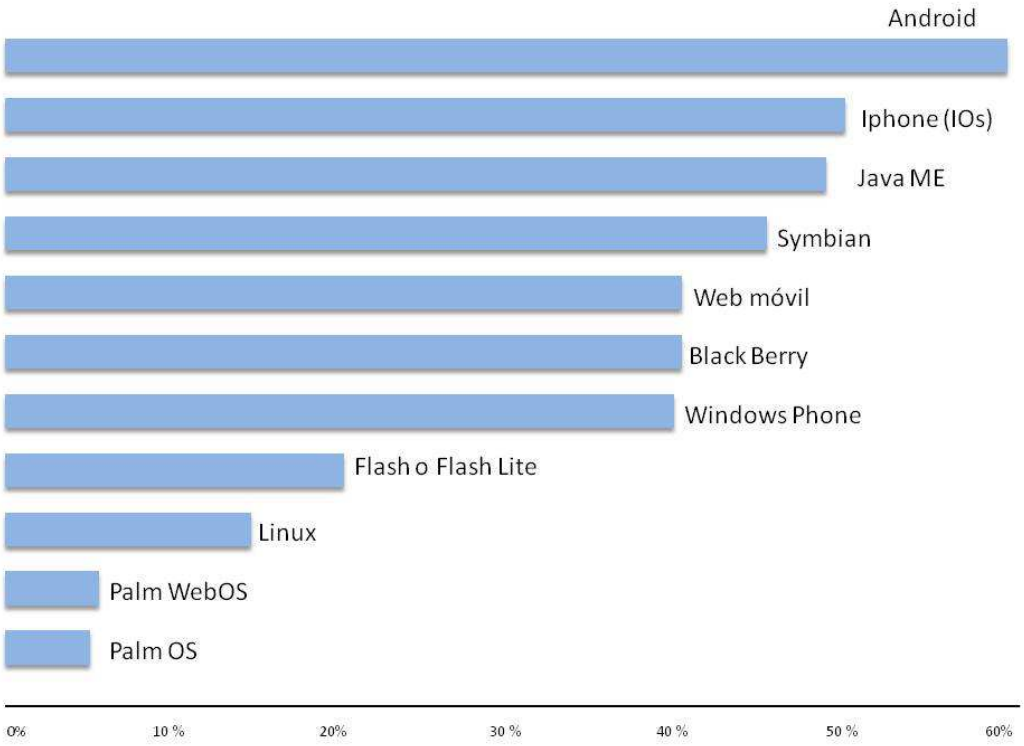


Ilustración 10. Plataformas más utilizadas por los desarrolladores de aplicaciones móviles[4].

Sin lugar a dudas, el cambio más significativo en los últimos tiempos es que la distancia entre Android y iOS, por un lado, y el resto de plataformas, por otro, se está haciendo cada vez mayor. La app store de Apple contiene más de trescientas mil aplicaciones, mientras que estimaciones recientes sitúan el número de aplicaciones del Market de Android en ciento treinta mil. Los motivos por los que los desarrolladores se mueven hacia iOS y Android pueden ser [4]:

- Apple ofrece una plataforma que es relativamente fácil de aprender y de usar, con la que el desarrollador puede diseñar UI muy buenas. Además, tiene la tienda de aplicaciones más grande y, aunque el problema de la certificación es un inconveniente para algunos, no existen los problemas de portabilidad y fragmentación.

[4] Fuente: Mobile Developer Economics 2010 and Beyond. Producido por VisionMobile. Patrocinado por Telefónica Devloper Communities. Licenciado bajo licencia Creative Commos Attribution 3.0.

- Android, por otro lado, ha ido ganando ímpetu en todos los campos asaltando los mercados clave de sus competidores. Por supuesto, tiene muchos inconvenientes derivados de la fragmentación, pero estos se pasan por alto muchas veces debido a la dependencia de muchos fabricantes de esta plataforma.

Otro aspecto importante a comentar es la disparidad que ha habido entre las ventas de dispositivos para cada plataforma y el número de aplicaciones disponibles.

2.3.4. Lenguajes de programación para aplicaciones móviles

El lenguaje de programación que se utilice en el desarrollo de una aplicación móvil depende fundamentalmente de la plataforma para la que está destinada dicha aplicación [4]. Por ejemplo, si se desea desarrollar para Symbian OS se pueden utilizar los lenguajes C++, Java o .NET. En el siguiente cuadro se explicitan las diversas plataformas y los lenguajes de programación que se pueden utilizar para el desarrollo de aplicaciones móviles para las mismas:

Plataforma	Lenguajes de programación
IPhone - iOS	Objective-C. Swift
Android	Java Kotlin Java Script Etc.
BlackBerry	Java MicroEdition.
Symbian OS	C++, Java .NET
Windows Mobile	VisualC++, VisualC#, VisualBasic, JScript ASP.NET

Tabla 2. Sistemas Operativos Móviles y sus lenguajes de programación.

En los siguientes incisos se realiza la descripción del Lenguaje de marcado extensible (XML) utilizado en este proyecto como maquetador de las interfaces y del lenguaje de programación Kotlin utilizado como motor controlador de las interfaces.

Kotlin

Kotlin [6] es un lenguaje de programación de tipado estático[5], tiene construcciones tanto orientadas a objetos como funcionales. Se puede utilizar en los estilos OO[6] y FP[7], o mezclar elementos de los dos. Está dirigido a Android, Native, corre sobre la Máquina Virtual de

[5]Se dice de un lenguaje de programación que usa un tipado estático cuando la comprobación de tipificación se realiza durante la compilación, y no durante la ejecución.
[6] Programación Orientada a Objetos.
7 Programación Funcional.

Java [8] y también puede ser compilado a código fuente de JavaScript[9]. Su desarrollo primario es de un equipo de programadores de JetBrains[10]. Kotlin es gratuito, se desarrolla bajo la licencia Apache 2.0 y el código fuente está disponible en GitHub[11]. El proyecto comenzó en 2010 y fue de código abierto desde muy temprano. El primer lanzamiento oficial 1.0 fue en febrero 2016. La versión actualmente publicada es 1.2.31, publicada el 23 de marzo de 2018. Desde el equipo se argumentó su creación debido a que la mayoría de lenguajes no tienen las características que buscaban, con la excepción de Scala. Sin embargo, el lento tiempo de compilación de este es una deficiencia obvia. Uno de los objetivos establecidos de Kotlin es el de conservar la velocidad de compilación [6].

En enero de 2012 la revista Dr. Dobb lo nombró como lenguaje del mes, este lo describe como un lenguaje de uso general destinado para uso industrial, donde la principal motivación de su creación es mantener las características positivas de Java, su rápida compilación, y buscando superarlo en cuanto a seguridad, flexibilidad, precisión, pero buscando no ser excesivamente complejo. Y por supuesto totalmente compatible con Java permitiendo a las compañías hacer una migración gradual de este a Kotlin [17].

Ilustración 11. Logo Lenguaje de Programación Kotlin.

En una entrevista brindada por el equipo de desarrollo del lenguaje pertenecientes a JetBrains a la revista Oracle Technology Network, estos hacen énfasis en resaltar que se apuesta fuertemente en la utilización de las bibliotecas ya existentes de Java, a la integración con sus Frameworks, a diferencia de otros lenguajes cuya apuesta es la de abandonar el JDK, construir su propia plataforma y biblioteca de colecciones, lo cual ocasiona inconvenientes a la hora de interoperar con el código Java [21].
En cuanto a la consulta sobre por qué Kotlin es más conciso que Java, ellos afirmaron que Kotlin generalmente requiere menos ceremonia, como por ejemplo:

- La inferencia de tipo es mucho más fuerte, por lo que no tiene que repetirse, especificando los mismos tipos una y otra vez.
- Las declaraciones de clase son más concisas, gracias a los conceptos de propiedades y constructores primarios.
- La mayoría de las veces no se necesitan delegar sobrecargas, gracias a los valores predeterminados para los parámetros de función.
- Las clases de utilidad estática no son necesarias debido a las funciones de nivel superior.

Pero lo más importante es que Kotlin ofrece abstracciones más flexibles que Java (incluido Java 8), como por ejemplo:

[8] Una máquina virtual Java (JVM) es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.
[9] JavaScript (JS) es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
[10] https://www.jetbrains.com
[11] GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora.

- Las funciones y propiedades de extensión en Kotlin se pueden agregar a cualquier clase / tipo sin alterar la definición de la clase.
- Las funciones de orden superior (pasar el código como valores) son mucho más prácticas, porque Kotlin admite tipos de función adecuados (a diferencia de las conversiones SAM de Java 8 que le hacen crear una nueva interfaz cada vez que necesita una nueva firma de función para pasarla)
- La variación del sitio de la declaración, y las colecciones de variantes en particular, hacen que el procesamiento de datos común sea mucho más natural al eliminar la necesidad de comodines ubicuos en tipos genéricos [21].

Kotlin es compatible con todos los principales IDE de Java, incluidos IntelliJ IDEA, Android Studio, Eclipse y NetBeans. Además, cuenta desde la página oficial del lenguaje con un compilador en línea[12], donde también se encuentran distintos ejemplos.
A continuación se describe la sintaxis básica del lenguaje Kotlin.

Tipos básicos [32]

En esta sección se describen los tipos básicos utilizados en Kotlin: números, caracteres, valores booleanos, arreglos y cadenas.
Una de las diferencias más marcadas con respecto al lenguaje Java es que en Kotlin todos los tipos son objetos. Es decir no existe la diferencia entre los tipos primitivos: int, long, boolean, byte, char y tipos de referencia (ej. Array, String). No hace falta utilizar envoltorios como ser java.lang.Interger para que los de tipo primitivo se comporten como objetos.

Números

Kotlin maneja los números de una manera cercana a Java, pero no exactamente igual. Por ejemplo, no hay conversiones de ampliación implícitas para números, y los literales son ligeramente diferentes en algunos casos.

Kotlin proporciona los siguientes tipos incorporados que representan números:

Tipo	Bit
Double	64
Float	32
Long	64
Int	32
Short	16

[12] https://play.kotlinlang.org

Byte	8

Tabla 3. Tipos de datos numéricos en Kotlin.

Se declaran colocando en primer lugar el tipo de variable, es decir si es solo lectura o si es mutable, conceptos que se explicitan más adelante en este inciso. Luego se indica el nombre de la variable y se le asigna un valor.

```
//Variable de tipo entera
val myInt = 55

//Variable de tipo Long
val myLong = 40L

//Variable de tipo Float
val myFloat = 34.43F

//Variable de tipo Double
val myDouble = 45.78

//Variable de tipo Hexadecimal
val myHexadecimal = 0x0F
//Variable de tipo Binaria
val myBinary = 0b010101
```

En el caso del tipo de dato numérico Long se crea un literal Long agregando el sufijo L, y para Float se agrega el sufijo F o f. Los números también pueden ser escritos en notación hexadecimal usando el prefijo 0x o 0X y en binario usando el prefijo 0b o 0B. En caso no de utilizar un literal se declara explicitando el tipo de dato:

```
//Declarando el tipo de dato.
val myLongAgain: Long = 40
```

En cuanto a la conversión de tipos, se debe realizar de manera explícita. Es decir debe utilizarse una función para indicar a qué tipo de dato se desea convertir. Las funciones son: `toByte()`, `toInt()`, `toLong()`, `toFloat()`, `toDouble()`, `toChar()`, `toShort()`.Por ejemplo:

```
//Se declara la variable como de tipo entero
val myInt = 987
//Se convierte desde un entero a un long
val myLong = myInt.toLong()
```

Booleano

El tipo Boolean en Kotlin es el mismo que en Java. Su valor puede ser verdadero o falso. Los operadores disyunción (||), conjunción (&&), y negación (!) pueden ser ejecutados sobre tipos booleanos. Por ejemplo:

```
//Se declaran dos variables de tipo booleano.
val myTrueBoolean = true
val myFalseBoolean = false
```

```
//Se declaran variables de tipo enteras y se les asigna un valor.
val x = 1
val y = 3
val w = 4
val z = 6

/*Se declara una variable n que será de tipo booleana y su valor dependerá
de si se cumple o no la expresión: x < z && z > w de acuerdo a los valores
asignados anteriormente.*/
val n = x < z && z > w // n is true
```

Caracteres - Char

Las cadenas pueden ser creadas ya sea con comillas dobles o comillas triples. Además, los caracteres de escape pueden ser usados con comillas dobles. Por ejemplo:

```
val myString = "This is a String"
val escapeString = "This is a string with new line \n"
```

Las comillas triples se utilizan para crear una cadena que se extienda múltiples líneas en el archivo fuente. Por ejemplo:

```
val multipleStringLines = """
        This is first line
        This is second line
        This is third line """
```

Se puede realizar interpolación de cadena o plantillas de cadena. Es una manera más sencilla de realizar cadenas dinámicas. Usando plantillas de cadena, se pueden insertar variables y expresiones en una cadena. Por ejemplo:

```
//Declaramos la variable
val accountBalance = 200

//Se crea una cadena literal y referimos a la variable accountBalance
anteponiendo el carácter $ al nombre de esta.
val bankMessage = "El saldo de su cuenta es $accountBalance"
// El saldo de su cuenta es 200
```

También pueden llamarse métodos desde un String utilizando la interpolación por medio de ${}. Por ejemplo:

```
val name = "Dami"
val message = "La primera letra en mi nombre es ${name.first()}"
// La primera letra en mi nombre es D
```

Arreglos

En Kotlin, hay dos maneras principales de crear un arreglo: usando la función de ayuda arrayOf() o el constructor Array().

Función arrayOf()

Por ejemplo, creación de un arreglo con algunos elementos usando arrayOf().

```
val myArray = arrayOf(4, 5, 7, 3)
```

Ahora, para acceder a cualquier elemento, se puede usar su índice: myArray[2]. Se pueden ingresar distintos tipos en el arrayOf() como argumentos el cual será un arreglo de tipo mixto.

```
val myArray = arrayOf(4, 5, 7, 3, "Dami", false)
```

Para hacer cumplir que todos los valores del arreglo tienen el mismo tipo, ej. Int, se declara un tipo llamando arrayOf<Int>() o intArrayOf().

```
val myArray3 = arrayOf<Int>(4, 5, 7, 3) //Compilará
val myArray4 = intArrayOf(4, 5, 7, 3, "Dami", false)  /* No compilará
porque se intenta ingresar valores distintos de Int y se utiliza una
función intArrayOf*/
```

Es decir, se puede indicar el tipo de datos que llevará el array de dos maneras. Otras funciones para crear arreglos de otros tipos son:

Función	Tipo
charArrayOf()	Char
booleanArrayOf()	Boolean
longArrayOf()	Long
shortArrayOf()	Short
byteArrayOf()	Byte

Tabla 4. Función ArrayOf() tipificada.

El Constructor Array()

Ahora se muestra cómo crear un arreglo con Array(). El constructor de esta clase requiere un tamaño y una función lambda[13]. En este caso, el trabajo de la función lambda es inicializar el arreglo con elementos.

```
val numbersArray = Array(5, { i -> i * 2 })
```

En el código de arriba, se establece 5 como el tamaño del arreglo en el primer argumento. El segundo argumento toma una función lambda, la cual toma el índice del elemento del arreglo y después devuelve el valor para ser insertado en ese índice en el arreglo. Así es que se crea un arreglo con elementos 0, 2, 4, 6, y 8.

[13]En el ámbito de la programación, una función anónima (función literal, expresión lambda) es una subrutina definida que no está enlazada a un identificador.

Paquetes

La especificación del paquete debe estar en la parte superior del archivo fuente:

```
package my.demo
import java.util.*
// ...
```

Variables

Las variables locales de solo lectura se definen utilizando la palabra reservada **val**. Se les puede asignar un valor una sola vez. Por ejemplo:

```
// Asignación inmediata
val a: Int = 1
// `Int` tipo inferido
val b = 2
// Tipo requerido cuando no se proporciona uno inicial
val c: Int
//Asignación diferida
c = 3
```

Las variables que se pueden reasignar utilizan la palabra reservada **var**:

```
// `Int` tipo inferido
var x = 5
x += 1
println("x = $x")
```

La diferencia entre las palabras clave **val** y **var** es que el primero es inmutable o de solo lectura (su valor no puede ser cambiado), mientras que el último es mutable (su valor puede ser cambiado).

Funciones

Las funciones se definen utilizando la palabra reservada fun, luego el nombre de la función, en caso de utilizar parámetros estos irán a continuación del nombre en paréntesis: nombre: tipo de dato y si requiere más de un parámetro se los separa con una coma. Luego de los parámetros en caso de tenerlos se indica el tipo de dato que devuelve la función. Las líneas de código de la función se encierran entre corchetes. Algunos ejemplos:

```
// Función que tiene dos parámetros Int con Int  de tipo de retorno
fun sum(a: Int, b: Int): Int {
    return a + b
}

// Función con una línea de expresión y tipo de retorno inferido
fun sum(a: Int, b: Int) = a + b
```

```
//Función que no devuelve ningún valor
fun printSum(a: Int, b: Int): Unit {
    println("sum of $a and $b is ${a + b}")
}
```

Nulabilidad [32]

En Kotlin se puede evitar los NullPointerExeption ya que es un lenguaje NullSafety. Esto quiere decir que por defecto los tipos no pueden ser nulos a menos que se indique de manera explícita. Por ejemplo, en el código siguiente se declara la variable de tipo String de nombre name y se le asigna el valor null. Al asignarle el valor nulo y no indicar de manera explícita que este tipo puede aceptar este tipo de valor la siguiente línea no compilará.

```
// No compilará
val name: String = null
```

Para que pueda compilar y se acepte la asignación, se debe declarar el nombre del tipo como anulable agregando "?" después de este.

```
// Compilará
val name: String? = null

// El nuevo valor será Dami
name = "Dami"
```

De esta manera se indica al compilador que el valor del tipo puede almacenar una referencia de objeto o puede ser anulable. También puede ser utilizado con Int?, Byte?, Long?, MiClase?, etc.

El operador de llamada segura ?.

El siguiente código no se compilará, ya que Kotlin es un lenguaje seguro para nulos:

```
var name: String? = null
print(name.length) // no compilará
```

A la variable name se le asigna el valor null. Ahora, al invocar la propiedad length en esa variable provocaría un error NullPointerException en Java. En Kotlin, su compilador no permitirá la invocación de esta propiedad dado que la variable podría ser null y derivar en un NullPointerException.
Este problema se soluciona utilizando el operador de llamada segura "?." que se coloca luego del nombre de la variable antes de invocar a la propiedad length. De esta manera, se indica de forma explícita al compilador que invoque la propiedad solo si el valor no es null. Y que en caso de serlo, el compilador utilizará la cadena "null" como valor para nosotros. Esto funciona también para los métodos y no solo para las propiedades. Por ejemplo:

```
val v: String? = null
// Compilará e imprimirá la cadena "null"
print(v?.length)
```

Cuando se llama a un método de un nullable, el tipo de retorno también será nullable. Así es que, siguiendo con el ejemplo anterior, el tipo de retorno de la expresión v?.length cuando v es anulable será Int?.

```
/*Se declara la variable v de tipo String nullable y se le asigna el valor null*/
val v: String? = null
/* Se declara una variable len de tipo Int nullable y se le asigna el valor devuelto por la propiedad length*/
val len: Int? = v?.length
// Compilará e imprimirá la cadena "null"
print(len)
```

Se puede omitir la comprobación de nulidad si reemplazamos el operador “?.” por ”!!”. Aunque no se recomienda dada la alta probabilidad de errores NullPointerException al utilizar esta opción.

El operador de Elvis ?:

Este operador ?: se llama el operador de Elvis (porque su forma se parece a la cabeza de Elvis). Se utiliza para proporcionar un valor alternativo para la variable si es null. Por ejemplo:

```
//Se declara una variable y se le asigna el valor null
val username = null
val name: String = username ?: "No name"
// Compilará e imprimirá la cadena "No name"
print(name)
```

Lo que hace en el ejemplo anterior el compilador es asignar la cadena “No name” a la variable name, porque el primer nombre de usuario es null. Si el primer valor no era null, ese valor se asignaría a la variable.

Bucles [32]

Kotlin utiliza while, do-while, y for para bucles.

while

Una declaración de repetición permite especificar que un bloque de código debe repetirse mientras alguna condición sigue siendo verdadera. Así que en Kotlin, usar el bucle while es igual que en otros lenguajes como Java.

```
while (condición) {
    // execute code here
}
```

Mientras la condición sea verdadera, se ejecutará el código dentro de las llaves o el cuerpo del bucle. Si la condición es falsa, entonces el cuerpo del bucle no se ejecutará.

do…while

Kotlin también utiliza do…while para la construcción de un bucle.

```
do {
    // execute code here
} while (condición)
```

El bucle do-while prueba la condición después de ejecutar el cuerpo del bucle. Esto significa que el cuerpo se ejecuta al menos una vez.

for

Un bucle for es una declaración de repetición que permite iterar sobre objetos mientras una condición dada es verdadera.

En Kotlin, la construcción del bucle for funciona con iteración sobre rangos, colecciones u otros iterables. Utiliza el operador “in” que se utiliza para determinar si un valor está presente en un rango determinado. Por ejemplo:

```
for (a in 1..5) {
     print("$a ") // imprimirá  1 2 3 4 5
}
```

En el código anterior, se está iterando a través de un rango cerrado del 1 al 5 e imprimiendo cada valor en el rango.

Condiciones [32]

Kotlin tiene tres tipos de sentencias de declaraciones de condiciones: if, if…else y when.

if

Una declaración if ejecuta un bloque de código si una condición es verdadera, o simplemente la omite si la condición es falsa.

```
val number = 20
if (number % 2 == 0) {
    print("$number es divisible por 2") // 20 es divisible por 2
}
```

If…else

El if..else realiza una acción si la condición es verdadera y realiza una acción diferente si la condición es falsa.

```
val number = 13
if (number % 2 == 0) {
    print("$number es divisible por 2")
} else {
    print("$number no es divisible por 2") // 13 no es divisible por 2
}
```

Una característica importante que distingue la declaración if..else en Kotlin de otros lenguajes de programación es la capacidad de asignar el valor devuelto por la declaración a una variable. Esto es posible ya que en Kotlin una declaración de este tipo puede utilizarse también como una expresión. Por ejemplo:

```
val number = 13
//Se asigna a la variable result el valor obtenido de la expresión if..else
val result = if (number % 2 == 0) {
    "$number es divisible por 2"
} else {
    "$number no es divisible por 2"
}
print(result) // 13 no es divisible por 2
```

when

Kotlin introduce el condicional when como reemplazo de switch, el cual es comúnmente utilizado en varios lenguajes de programación como C++, java, etc. When es más conciso en comparación de switch. La declaración de when realiza diferentes acciones en función de los posibles valores de una variable de tipo, String, Byte, Short o cualquier otro objeto.

```
fun guessTheNumber(number: Int) {
    when (number) {
        1 -> println("number is 1")
        2 -> println("number is 2")
        3 -> println("number is 3")
        else -> println("number is neither 1, 2 or 3")
    }
}
```

Si se quiere ejecutar más de una acción en una rama, se necesita envolver las acciones entre llaves { }.

```
val number = 2
when (number) {
            1, 2 -> println("number is either 1 or 2")
                  /* number is either 1 or 2*/
            3 -> {// block of code executed
                 println("number is 3")
                 println("it is an odd number")
                 }
              }
```

Además, se pueden combinar valores de prueba en una sola rama como en la primera rama.

Comentarios [32]

Al igual que Java y JavaScript, Kotlin admite comentarios de final de línea y de bloque. Por ejemplo:

```
//Comentario de fin de línea
```

```
/* Este es un comentario de bloque.
   de múltiples líneas. */
```

Existen otros lenguajes, como Kotlin, diseñados para la JVM:

- **Ceylon**, diseñado por Red Hat, sigue el paradigma orientado a objetos, su tipado es fuerte y estático. Multiplataforma [18].
- **Clojure**, diseñado por Rich Hickey, sigue el paradigma funcional, lenguaje de programación multiparadigma. Su sistema de tipos es dinámico y fuerte. Es multiplataforma [19].
- **Scala**, es un lenguaje de programación multi-paradigma diseñado para expresar patrones comunes de programación en forma concisa, elegante y con tipos seguros. Integra sutilmente características de lenguajes funcionales y orientados a objetos. La implementación actual corre en la máquina virtual de Java y es compatible con las aplicaciones Java existentes [20].
- Otros:
- **Tipado sensitivo al flujo**
- **Fantom**
- **Gosu**

XML

Extensible Markup Language (XML) [29] es un formato de texto simple y muy flexible derivado de SGML (ISO 8879). Originalmente diseñado para enfrentar los desafíos de la publicación electrónica a gran escala, XML también está desempeñando un papel cada vez más importante en el intercambio de una amplia variedad de datos en la Web y en otros lugares.

Se traduce como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

XML no ha nacido únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande, con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Estructura de un documento XML

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de porciones de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Una porción de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
     <Mensaje>
          <Remitente>
               <Nombre>Nombre del remitente</Nombre>
               <Mail> Correo del remitente </Mail>
          </Remitente>
          <Destinatario>
               <Nombre>Nombre del destinatario</Nombre>
               <Mail>Correo del destinatario</Mail>
          </Destinatario>
          <Texto>
               <Asunto>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Asunto>
               <Parrafo>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Parrafo>
          </Texto>
     </Mensaje>
</Edit_Mensaje>
```

Partes de un documento XML

Un documento XML está formado por el prólogo y por el cuerpo del documento, los cuales se describen a continuación:

Prólogo

Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.

El prólogo de un documento XML contiene:

- Una declaración XML. Es la sentencia que declara al documento como un documento XML.

- Una declaración de tipo de documento. Enlaza el documento con su DTD (definición de tipo de documento), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
- Uno o más comentarios e instrucciones de procesamiento.

Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Cuerpo

A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener solo un elemento raíz, característica indispensable también para que el documento esté bien formado. Sin embargo, es necesaria la adquisición de datos para su buen funcionamiento.

Por ejemplo:

```
<Edit_Mensaje>
    (...)
</Edit_Mensaje>
```

Elementos

Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.

Atributos

Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.

Por ejemplo, los siguientes datos de un producto:

- Código: G45
- Nombre: Gorro de lana
- Color: negro
- Precio: 12.56

Su representación en un documento XML podría ser, por ejemplo:

```
<producto codigo="G45">
   <nombre color="negro" precio="12.56">Gorro de lana</nombre>
</producto>
```

En este ejemplo se han escrito tres atributos: código, color y precio.

Comentarios

Comentarios a modo informativo para el programador que han de ser ignorados por el procesador. Los comentarios en XML tienen el siguiente formato:

```
<!--- Esto es un comentario --->
<!-- Otro comentario -->
```

2.3.5. Aplicaciones móviles para Test de Orientación Vocacional

En esta sección se describen las soluciones existentes que se encuentran en el mercado nacional similares a la propuesta en este Trabajo Final, desarrolladas por instituciones públicas o desarrollos de empresas privadas.
El estudio del estado del arte se realiza conforme a lo descripto por la metodología de investigación para la Ingeniería de Software utilizada para el desarrollo de este Trabajo Final, la cual se describe en la sección 2.5.1.

2.3.5.1. Orienta tu futuro

Aplicación móvil que cuenta también con su versión web desarrollada e implementada por el Gobierno de la Provincia de Buenos Aires. Dirigida a los jóvenes que quieran iniciar sus estudios universitarios, los cuales deberán registrarse para poder acceder. Los datos que deben ingresar son: Nombre, email, fecha de nacimiento, distrito y género. Luego de esto podrán acceder a una serie de ejercicios y test. Se compone de dos etapas, la primera llamada: ¿En qué sos bueno? Que consta de 22 preguntas y ejercicios; luego de completar esta instancia se accede a la segunda llamada ¿Qué te gusta? Donde encontramos 20 preguntas y ejercicios. Luego de completar estas dos etapas es que se accede al resultado.

Orienta tu futuro, acompañando los resultados, se brinda información de las carreras afines que se dicten en universidades, terciarios y cursos de formación profesional próximos a su ubicación geográfica. Necesita conexión para poder realizar los test [30].

La aplicación móvil propuesta en el presente trabajo se diferencia de esta en que no se requiere conexión a internet para realizar el test de orientación vocacional, como así tampoco se requiere conexión para acceder a la información sobre carreras, servicios, becas, etc. Además, se brinda información sobre la oferta académica de la UNCA, sirviendo de esta manera como una herramienta para promoción de carreras.

Ilustración 12. Pantalla de ingreso a aplicación móvil "Orienta tu futuro"

2.3.5.2. Test Vocacional de DominGame – Educación

Aplicación móvil que no cuenta con versión web. De acuerdo a las respuestas, se obtiene un informe con un perfil aproximado de los intereses y habilidades destacadas y las dos áreas ocupacionales que más se relacionan con esas características.

En su inicio contaba con un solo test. En su actualización realizada el 2 de agosto de 2018, se aumentó un segundo test que como resultado ofrece también un listado de carreras asociadas al resultado. No se requiere conectividad para realizar los test [31].

Esta aplicación se diferencia de la propuesta en el presente trabajo en que no está asociada a ninguna institución educativa, por lo cual las carreras que sugiere al finalizar el test pueden no ser alcanzables por el usuario, es decir, que no se dicten en su cercanía o quizás se dicte en una universidad privada.

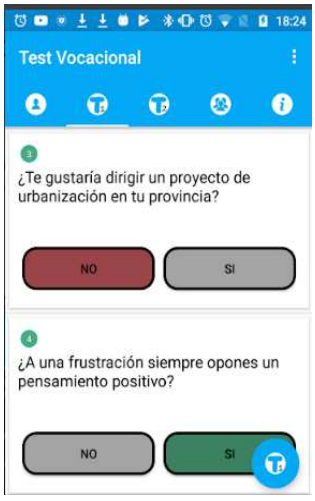


Ilustración 13. Pantalla de preguntas de Test de Orientación Vocacional.

2.4. Metodologías de desarrollo de aplicaciones móviles

En la actualidad existen distintas metodologías para el proceso de desarrollo de software. Algunas de ellas son:

2.4.1. Modelo Waterfall

El modelo Waterfall [28] es el modelo más estático y predictivo. Por lo cual es aplicable en proyectos en los que los requisitos están fijados y no van a cambiar durante el ciclo de vida del desarrollo. Este modelo es una secuencia de actividades (o etapas) que consisten en el análisis de requerimientos, el diseño, la implementación, la integración y las pruebas. Se interpreta como el agua que va cayendo de un estanque al siguiente. Se le da mucho énfasis a la planificación, a los tiempos, a las fechas límite y al presupuesto. Se caracteriza por ordenar de manera rigurosa las etapas del ciclo de vida de software, dado que el comienzo de cada etapa debe esperar a la finalización de la inmediata anterior.

En el contexto del desarrollo de aplicaciones móviles, el modelo Waterfall puede ser aplicable a proyectos realmente controlados y previsibles, en los que no hay mucha incertidumbre por lo que se desea hacer y para los que no son importantes los cambios constantes en la industria.

2.4.2. Desarrollo Rápido de Aplicaciones

El desarrollo rápido de aplicaciones [28] o RAD por sus siglas en inglés, es un método de desarrollo iterativo cuyo objetivo es conseguir prototipos lo antes posible para mejorarlos después. Se suele priorizar la implementación sobre la planificación, y se utilizan muchos patrones de diseño conocidos para poder adaptarse de la mejor manera a cambios en los requerimientos.
El método comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades CASE (Computer Aided Software Engineering). Tradicionalmente, el desarrollo rápido de aplicaciones tiende a englobar también la usabilidad, utilidad y la rapidez de ejecución.
El desarrollo rápido de aplicaciones es un método muy útil para el desarrollo de proyectos realmente urgentes con tiempos de entrega muy cortos, pero cuenta con algunas desventajas como por ejemplo el progreso en el desarrollo del proyecto es más difícil de medir, al ser apresurados los tiempos existen más posibilidades de fallas por la necesidad de codificar sin contar con el tiempo suficiente para hacerlo de manera tranquila. Al conseguir un prototipo puede ser que este no escale.

2.4.3. Desarrollo Ágil

El desarrollo ágil [28] se basa en los principios del manifiesto ágil y sus valores éticos, los cuales son:

- Dar más valor a los individuos y a sus interacciones que a los procesos y herramientas.
- Dar más valor al software que funciona que a la documentación exhaustiva.
- Dar más valor a la colaboración con el cliente que a la negociación contractual.
- Dar más valor a la respuesta al cambio que al seguimiento de un plan.

El desarrollo ágil es apropiado para proyectos cambiantes, ya sean grandes o pequeños, ya que mediante estos valores se pueden mitigar los riesgos. Se basa en seis pasos comunes dentro del ciclo de vida del software: planificación, análisis de requisitos, diseño, codificación, test y documentación. En cada interacción con el cliente, el equipo de desarrollo no entrega todo el programa, sino que se van añadiendo pequeños elementos totalmente probados, sin errores, con el fin de que la solución final esté completamente operativa desde el minuto uno.
Los métodos ágiles suelen ser muy adecuados para el desarrollo de aplicaciones móviles por las siguientes razones [28]:

- **Alta volatilidad del entorno:** Con cambios en entornos de desarrollo, nuevos terminales y nuevas tecnologías a un ritmo mucho más elevado que en otros entornos de desarrollo.
- **Equipos de desarrollo pequeños.**
- **Software no crítico:** No suelen ser aplicaciones de alto nivel de criticidad, dado que suelen ser aplicaciones para entretenimiento o gestión empresarial no crítica.
- **Ciclos de desarrollo cortos:** Dada la evolución constante de la industria, se requieren ciclos de vida realmente cortos para poder dar salida a las aplicaciones a tiempo.

El término Ágil, aplicado a software nace en 2001, en una reunión celebrada en Utah, donde un grupo de 17 expertos de la industria de software tenían como objetivo la elaboración de una metodología que sirviera como alternativa ante los procesos tradicionales, los cuales se caracterizan por ser fuertemente orientados a la documentación. Como resultado de esta reunión nace la Alianza Agile [11], la cual es una comunidad donde los miembros comparten sus experiencias y conocimientos con la idea de crear productos innovadores utilizando metodologías agiles en su desarrollo. Esta comunidad se extiende en varios países, Argentina es uno de ellos. Los miembros de la comunidad argentina se definen como: “Se trata de un grupo de gente con ganas de desarrollar productos y dar valor a nuestros usuarios y clientes. Creemos que las ideas de Desarrollo de Software Ágil pueden ayudar a esto, y por eso que tratamos de aprender y crecer juntos, intercambiando experiencias y conocimiento” [14].

Agile Software Development (Desarrollo Ágil de Software) es un término genérico que enmarca un conjunto de métodos y prácticas basados principalmente en los valores y principios que se expresan en el Manifiesto Ágil, los cuales se abordarán más adelante en el documento. Estas técnicas permiten la incorporación de cambios con mayor rapidez y fluidez en el proceso de desarrollo de software. Las soluciones evolucionan por medio de la colaboración entre equipos auto organizados y multifuncionales que utilizan las prácticas apropiadas para su contexto. Algunas de las técnicas de desarrollo ágil más utilizadas son: XP, Scrum, Kanban y Open Up [11].

- **Programación extrema (XP) por sus siglas en inglés**: su creador es Kent Beck y se denomina extrema porque lleva a límites extremos algunos elementos y actividades comunes de la forma tradicional de programar.
- **Scrum**: este concepto nace a principio de los años 90, está basado en el estudio de gestión de equipos desarrollado por Hirotaka Takeuchi e Ikujiro Nonaka en 1986. Este es uno de los métodos ágiles que más se utiliza y es aplicable a otros tipos de proyectos. Cuenta con una organización (Scrum Alliance) sin fines de lucros que se encarga de difundirlo.
- **Kanban**: este es considerado como el Enfoque Lean. Kanban, significa tablero visual, es un sistema de información que controla de modo armónico la fabricación de los productos necesarios en la cantidad y tiempo necesarios en cada uno de los procesos que tienen lugar tanto en el interior de la fábrica, como entre distintas empresas.
- **Open Up**: es un proceso unificado ágil y liviano, que aplica un enfoque iterativo e incremental dentro de un ciclo de vida estructurado y contiene un conjunto mínimo de prácticas que ayuda al equipo a ser más efectivo desarrollando software. Es un modelo de desarrollo de software, es parte del Framework de modelo de proceso de Eclipse (Eclipse Process Framework), desarrollado por la fundación Eclipse.

2.4.4. Mobile-D

El método Mobile-D [28] se desarrolló junto con un proyecto finlandés en el 2004. Fue realizado, principalmente, por investigadores de la VTT (Instituto de Investigación Finlandés) y, a pesar de que es un método antiguo, sigue en vigor.
El objetivo es conseguir ciclos de desarrollos muy rápidos en equipos muy pequeños (de no más de diez desarrolladores) trabajando en un mismo espacio físico. Según este método, trabajando de esa manera se deben conseguir productos totalmente funcionales en menos de diez semanas. Se trata de método basado en soluciones conocidas y consolidadas: Extreme Programming (XP), Crystal Methodologies y Rational Unified Process (RUP), XP para las prácticas de desarrollo, Crystal para escalar los métodos y RUP como base en el diseño del ciclo de vida.

2.4.5. Metodología específica para el desarrollo de Aplicaciones Móviles

En esta sección se describe la metodología que se utilizará en el desarrollo de la aplicación móvil de Test Orientación Vocacional. Esta metodología fue propuesta como tema de investigación en la Universidad Distrital Francisco José de Caldas, Colombia, presentada por Gasca Mantilla, Maira Cecilia; Camargo Ariza, Luis Leonardo; Medina Delgado, Byron con el título “Metodología para el desarrollo de aplicaciones móviles” [9]. Esta metodología emplea como principales ejes la conceptualización de las tecnologías y de las metodologías ágiles para el desarrollo de software.

La metodología propuesta para el desarrollo de aplicaciones móviles se fundamenta en la experiencia de investigaciones previas en aplicaciones móviles [9]:

- **Evaluación del potencial de éxito para servicios de tercera generación denominada 6 M’s**, se extrae la concepción de que las aplicaciones móviles deben garantizar el cumplimiento de las necesidades de los usuarios y al mismo tiempo generen ingresos. La 6 M’s debe su nombre a los seis atributos que se miden para

evaluar el éxito del servicio propuesto: *Movement* (Movimiento), *Method* (Metodo), *Me* (Yo), *Moment* (Momento), *Money* (Dinero) y *Machines* (Máquinas) [12]. Los cuales se describen en la Tabla 5. Atributos de la Evaluación de las 6M's.

- **Ingeniería de software educativo con modelado orientado por objetos (ISE-OO)**, La ingeniería de software está compuesta por una serie de modelos que abarcan los métodos, las herramientas y los procedimientos. Estos modelos se denominan frecuentemente paradigmas de la ingeniería del software y la elección de un paradigma se realiza básicamente de acuerdo al tipo del proyecto y de la aplicación, los controles y las entregas a realizar. Debido a las características particulares de los desarrollos educativos, ya que se deben tener en cuenta los aspectos pedagógicos y de la comunicación con el usuario, en cada caso en particular, la respuesta a la problemática debe basarse en una adaptación de los actuales paradigmas de desarrollo a las teorías educativas que permitan satisfacer una demanda en especial [15].
  La metodología propuesta para el desarrollo de la aplicación móvil hereda de la ISE-OO el enfoque de los micromundos interactivos y la orientación por objetos [13].
  Un micromundo puede ser desde un conjunto de textos e imágenes articuladas entre sí, hasta un sofisticado sistema multimedia de simulación de fenómenos de la realidad y de conceptos abstractos que interactúan con las personas, las cuales, participan activamente en una experiencia que les permite crear, destruir y reacomodar el conocimiento que estas previamente poseen [16].
  Los elementos de los micromundos más utilizados en los servicios móviles interactivos son: Mundo, Escenarios, Personajes y Roles, Argumento e Historia, etc. [13].

- **Valores de las metodologías ágiles**. De las metodologías ágiles incluye los conceptos de los cuatro postulados o Manifiesto Ágil [11].

  El manifiesto hace énfasis en cuatro valores principales que deben soportar el desarrollo de software:

  - **Los individuos e interacciones por encima de los procesos y las herramientas:** para garantizar una mayor productividad, las metodologías ágiles valoran el recurso humano como el principal factor de éxito. Reconocen que contar con recurso humano calificado con capacidades técnicas adecuadas, facilidades para adaptarse al entorno, trabajar en equipo e interactuar convenientemente con el usuario, da mayor garantía de éxito que contar con herramientas y procesos rigurosos.

  - **Software funcionando por encima de la documentació**n: las metodologías ágiles respetan la importancia de la documentación como parte del proceso y del resultado de un proyecto de desarrollo de software, sin embargo, con la misma claridad hacen énfasis en que se deben producir los documentos estrictamente necesarios; los documentos deben ser cortos y limitarse a lo fundamental, dando prioridad al contenido sobre la forma de presentación.

- **La colaboración del cliente por encima de la negociación del contrato:** clásicamente el usuario o cliente es quien solicita e indica qué debe hacer el software, y espera los resultados de acuerdo con sus exigencias o expectativas, en los plazos establecidos. Con frecuencia las dos partes, cliente y equipo de desarrollo, asumen posiciones distantes, con ingredientes de rivalidad y prevención al punto de tener que dedicar tiempo valioso a la tarea de redactar, depurar y firmar el contrato. En este sentido, y complementando el valor que se da al trabajo en equipo, las metodologías ágiles incluyen de manera directa y comprometida al cliente o usuario en el equipo de trabajo. Es un ingrediente más en el camino al éxito en un proyecto de desarrollo de software. Más que un ambiente de enfrentamiento en el cual las partes buscan su beneficio propio, evadiendo responsabilidades y procurando minimizar sus riesgos, bajo la filosofía de las metodologías ágiles se busca el beneficio común, el del equipo de desarrollo y el del cliente. La participación del cliente debe ser constante, desde el comienzo hasta la culminación del proyecto, y su interacción con el equipo de desarrollo, de excelente calidad. Es el cliente quien sabe qué es lo que necesita o desea, el más indicado para corregir o hacer recomendaciones en cualquier momento del proyecto.

- **La respuesta al cambio por encima del seguimiento de un plan:** dada la naturaleza cambiante de la tecnología y la dinámica de la sociedad moderna, un proyecto de desarrollo de software se enfrenta con frecuencia a cambios durante su ejecución. Van desde ajustes sencillos en la personalización del software hasta cambios en las leyes, pasando por la aparición de nuevos productos en el mercado, comportamiento de la competencia, nuevas tendencias tecnológicas, etc. En este sentido, las metodologías pesadas con frecuencia caen en la idea de tener todo completo y correctamente definido desde el comienzo. No se cuenta entre sus fortalezas la habilidad para responder a los cambios. Por el contrario, en las metodologías ágiles la planificación no debe ser estricta, puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir. Una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para los siguientes meses [10].

Atributo	Descripción
Me / Yo	Se consideran todos los aspectos asociados al personal, a la mano de obra. Su capacitación, motivación, habilidad en su trabajo, etc.
Movement / Movimiento	Analiza la movilidad de la aplicación.
Money / Dinero	Analiza si existe algún costo en la utilización, licencias, utilización de la red.
Moment / Momento	Rapidez con la que realiza los procesos requeridos por el usuario.
Method / Método	Se evalúa la forma en la que se realizan las acciones. Al evaluar los métodos, se está evaluando la forma en cómo se produce

	independiente de la mano de obra.
Machines / Máquinas	Herramientas con las que se cuenta para dar salida al producto final. Software, hardware, etc. Se evalúa capacidad suficiente para cumplir su función, eficiencia, modernidad, etc.

Tabla 5. Atributos de la Evaluación de las 6M's

La metodología se encuentra enmarcada en cinco fases como se muestra en la Ilustración 14. Etapas de la Metodología de Desarrollo., denominadas: análisis, diseño, desarrollo, pruebas de funcionamiento y entrega. A continuación, se describe cada una de las actividades que intervienen en el desarrollo de la propuesta.

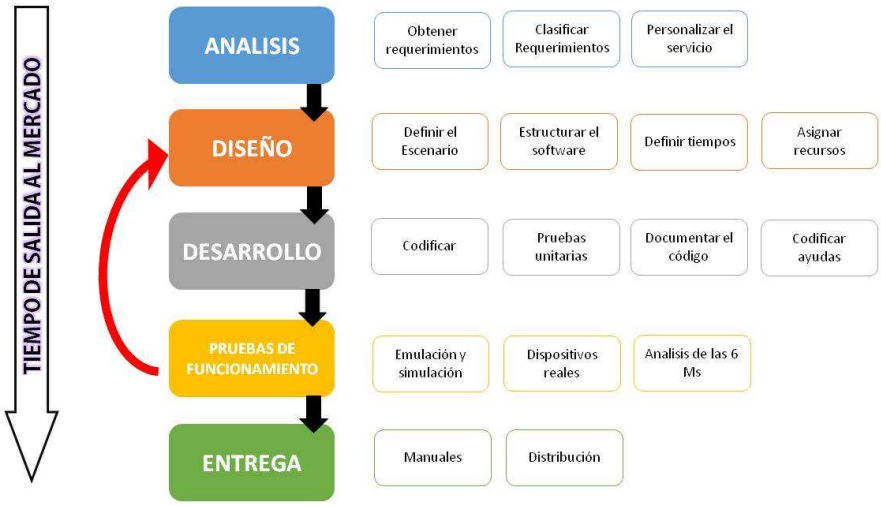


Ilustración 14. Etapas de la Metodología de Desarrollo.

Análisis

La Ingeniería de Requerimiento (IR) es el proceso de recopilar y analizar las necesidades del cliente o usuario para un sistema [4]. La IR cumple un papel primordial en el proceso de producción de software, ya que se enfoca en un área sensible: la definición de lo que se desea producir [3]. Su principal tarea consiste en la generación de especificaciones concretas que describan con claridad, sin ambigüedades, en forma consistente, verificable y compacta, las necesidades de los usuarios o clientes.

En esta fase inicial se realiza el análisis de requerimientos en función a lo expresado por las personas o entidad para la cual se está desarrollando el software. Plantea como principal objetivo la definición del entorno de la aplicación en desarrollo. Las tareas que se realizan:

- **Obtención de requerimientos:** se sugiere hacer una serie de entrevistas al cliente, para que manifieste los síntomas del problema o necesidades que se pretenden solucionar con las tecnologías móviles, o simplemente, para que señale las características que debe tener la aplicación.

- **Clasificar los requerimientos:** una vez identificados los requerimientos que debe tener el software, se procede a clasificarlos. Dichos requerimientos se pueden clasificar en entorno, mundo, funcionales y no funcionales.

  Se define *Entorno* como todo lo que rodea al servicio. Por ejemplo, las características técnicas del dispositivo móvil del cliente, el sistema operativo subyacente (móvil y servidores), la tecnología utilizada para la transferencia de información, el Sistema Manejador de Base de Datos, si se requiere, el formato de archivos y, otros módulos tecnológicos utilizados para el servicio.

  El *mundo* es la forma de cómo interactúan el usuario y la aplicación. Aquí se encuentran los requerimientos de la Interfaz Gráfica de Usuario, Graphical User Interface (IGU), la forma en que el software va a generar los datos de salida, el formato de los datos y los demás requerimientos que involucren la comunicación hombre-máquina, considerando la gama tecnológica de los teléfonos móviles de los usuarios a la que va dirigida el servicio.

  Los *requerimientos funcionales* son todos aquellos que demandan una función dentro del sistema. Se deben definir claramente cada una de las tareas que debe realizar la aplicación.

  Los *requerimientos no funcionales* son la estabilidad, la portabilidad, el rendimiento, el tiempo de salida al mercado y, el costo, entre otros.

- **Personalizar el servicio:** adicionalmente se deben analizar aspectos de la cotidianidad del cliente como preferencias, costumbres y particularidades del usuario, con el propósito de garantizar la aceptación del servicio.

Diseño

El objetivo de esta etapa es plasmar el pensamiento de la solución mediante diagramas o esquemas, considerando la mejor alternativa al integrar aspectos técnicos, funcionales, sociales y económicos. A esta fase se retorna si no se obtiene lo deseado en la etapa prueba de funcionamiento.

Las actividades que se realizan en esta etapa son:

- **Definir el escenario:** el diseño de aplicaciones móviles puede realizarse para la ejecución en distintos escenarios[14]. Entonces en función del sistema de conexión y sincronización con este; la sincronización se realiza para insertar, modificar o borrar información. Los distintos escenarios que se pueden definir son:

  Desconectado: los procesos se realizan en el dispositivo móvil desconectado. Después de terminar el proceso, si se requiere, puede conectarse con una aplicación central mediante el proceso de sincronización.

[14] El término escenario hace referencia al estado de conexión con el servidor.

Semiconectado: los procesos pueden ejecutarse en el dispositivo móvil desconectado, pero se requiere establecer conexión en algún momento para terminar el proceso, al sincronizar la información con el servidor o aplicación central. En los escenarios desconectado y semiconectado se recomienda utilizar los protocolos y tecnologías que se ajusten al servicio y capacidades tecnológicas del dispositivo. Algunos son: Media Transfer Protocol (MTP), Near Field Communication (NFC), SlowSync, FastSync, SyncML, entre otros.

Conectado: el dispositivo debe estar siempre conectado con la aplicación central o servidor para su correcto funcionamiento, no se almacenan datos o archivos en el móvil, la sincronización se realiza mediante la validación de formularios, usualmente se utiliza el Protocolo de Transferencia de Hipertexto (*Hypertext Transfer Protocol*, HTTP).

- **Estructurar el software:** se deben utilizar algunos diagramas de Lenguaje Unificado de Modelado, *Unified Modeling Language* (UML), según las necesidades del proyecto, modelando el sistema desde varias perspectivas, ver la siguiente Ilustración de Diagramas sugeridos para la estructuración del software.
  Se sugiere traducir los requerimientos obtenidos de la etapa anterior en un diagrama que describa en forma objetiva el servicio a implementar. Además, definir un patrón de diseño para flexibilizar, modular y reutilizar lo desarrollado; la selección del patrón de diseño debe estar acorde con el escenario del servicio. Algunos patrones que se ajustan a los escenarios de las aplicaciones móviles son: modelo-vista–controlador (MVC), diseño de capas, entre otros.

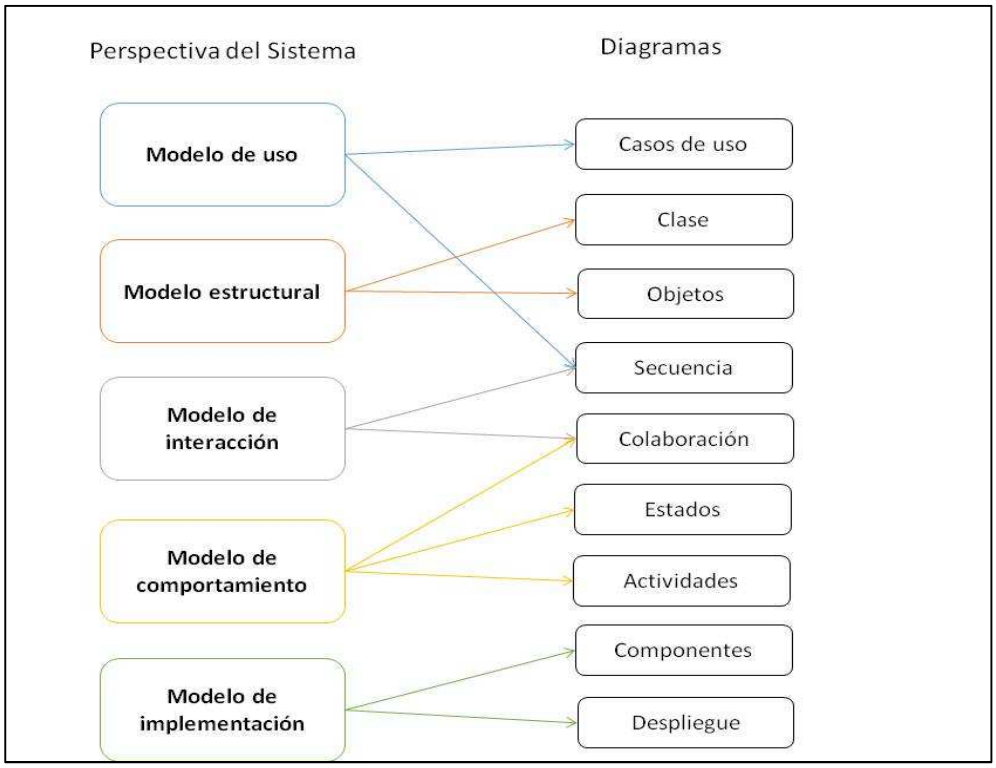


Ilustración 15. Diagramas sugeridos para el proceso de estructuración del software.

- **Definir tiempos:** se establecen los plazos para cada una de las actividades restantes, con el objetivo de terminar la aplicación a tiempo para su salida al mercado. Se debe tener en cuenta el diseño computacional del software realizado en la tarea anterior y, las características volátiles y dinámicas de los servicios móviles.

- **Asignar recursos:** se asignan los recursos para realizar cada actividad y alcanzar los objetivos propuestos, se deben considerar recursos humanos, financieros y tecnológicos. Además, se deben seleccionar las herramientas para el desarrollo de la aplicación móvil.

Desarrollo

El objetivo de esta fase es implementar el diseño en un producto de software. En esta etapa se realizan las siguientes actividades:

- **Codificar:** se escribe en el lenguaje de programación seleccionado, cada una de las partes definidas en los diagramas realizados en la etapa de diseño.

- **Pruebas unitarias:** se verifica el funcionamiento de la aplicación. En primer lugar, se comprueba la correcta operación de cada elemento desarrollado —objeto, clase, actividad, documento, entre otros— en forma individual; posteriormente, se pone en funcionamiento el conjunto de elementos, comprobando la interrelación entre ellos. Se ejecuta y se observan los resultados obtenidos, para compararlos con los esperados.

- **Documentar el código:** a medida que se codifica y se prueba cada elemento, se redacta una pequeña documentación sobre lo desarrollado.

- **Codificar ayudas:** además del manual de instalación y de usuario, deben existir una serie de ayudas que informen de manera didáctica lo que puede hacer el usuario con la aplicación. Estas ayudas deben ser codificadas en el mismo lenguaje de programación e integrada en la interfaz de aplicación para visualizarlas en el móvil.

Pruebas de funcionamiento

El objetivo de esta fase es verificar el funcionamiento de la aplicación en diferentes escenarios y condiciones; para esto se realizan las siguientes tareas:

- **Emulación y simulación:** se realizan pruebas simulando el escenario y emulando el dispositivo móvil, explorando todas las utilidades y funciones de la aplicación, introduciendo diferentes datos, inclusive erróneos, para medir la funcionalidad y el nivel de robustez del software. Si se encuentran algunas fallas, se debe regresar a la etapa de codificación en la fase de desarrollo para solucionar los problemas. Si las pruebas son satisfactorias se procede a la etapa de pruebas con dispositivos reales.

- **Dispositivos reales:** deben hacerse pruebas de campo en equipos reales para medir el desempeño y el rendimiento de la aplicación. Si se encuentran fallas en el tiempo de ejecución, si el software no cumple con los requerimientos especificados, o si el cliente solicita un cambio de última hora, hay que regresar a la fase de diseño para reestructurar y solucionar el inconveniente presentado.

- **Análisis de las 6 M's:** para valorar el potencial de éxito del servicio, se sugiere buscar un grupo de expertos en el campo del desarrollo móvil para que utilicen el método de evaluación de las 6 M's, y califiquen la presencia de los seis atributos en la aplicación desarrollada.

Entrega

Terminada la depuración de la aplicación y atendidos todos los requerimientos de última hora del cliente se da por finalizada la aplicación y se procede a la entrega del ejecutable, el código fuente, la documentación y el manual del sistema.

- **Manuales:** el objetivo es el entrenamiento; una aplicación móvil debe constar de un manual del sistema donde se indique el proceso de instalación, la atención a posibles fallas en tiempo de ejecución y, las especificaciones técnicas mínimas de hardware y software que requiere el equipo, para el funcionamiento adecuado del aplicativo desarrollado.

- **Distribución:** se define el canal de distribución de la aplicación, con el propósito de adecuar la aplicación a este medio. A continuación, se mencionan algunos de los canales de distribución existentes. Las tiendas físicas u outlets, especializadas o no, corresponden a las tiendas que venden dispositivos y servicios de telecomunicaciones, normalmente operadores o marcas como Apple. Los portales de operadores o desarrolladores de servicios ofrecen un catálogo amplio de aplicaciones y ventas vía Web Site desde el PC, que luego son instaladas en el

móvil. Las Applications Stores, son las tiendas *online* de los fabricantes de dispositivos o de sistemas operativos. OTA, Over the Air, es la comercialización de aplicaciones a través de la interfaz de radio. Los contenidos móviles son distribuidos a los terminales usando SMS, WAP y Streaming, entre otros. Los servicios basados en el Subscriber Identity Module (SIM), o la Universal Integrated Circuit Card (UICC), son aplicaciones instaladas previamente en el chip de estas tarjetas.

2.5. Metodología de Investigación para la Ingeniería de Software

A continuación, se plantea la necesidad de utilizar una metodología de investigación desarrollada específicamente para la Ingeniería de Software. Este planteo lo realiza el grupo de estudios LIDIS[15] [7] de cara a proponer una metodología de investigación propia del área, que será la utilizada en este proyecto y que se describirá en el siguiente inciso. Este análisis se incluye con la intención de poner en claro cuál es la real necesidad de utilizar una metodología de investigación en un proceso de desarrollo de software.

La Ingeniería de Software no es considerada en algún tipo de investigación en concreto o se confunde con cualquier tipo de proceso de desarrollo de software en algunos ámbitos [7][43]. Es una profesión de naturaleza tecnológica, que retoma teorías y conocimientos de diversas fuentes y aborda el desarrollo de software de calidad y a nivel industrial. Construye conocimiento en torno a las actividades que se realizan durante el proceso de desarrollo de software, definiendo así métodos, modelos y esquemas de funcionamiento que luego los profesionales del área aplican en sus actividades. Estos métodos, modelos y esquemas de funcionamiento se crean en base a la revisión y formalización de heurísticas surgidas de la experiencia real en procesos y productos de software.

Planteada esta situación es que se ve necesario definir métodos de investigación que aborden problemas y situaciones reales de la Industria de Software, que identifiquen, formalicen y teoricen sobre las mejores prácticas de la misma.

En la formulación de esta metodología, se realizó un análisis epistemológico en cuanto a la ingeniería, ingeniería de software, los métodos desarrollados en la historia de la ingeniería y la posibilidad de aplicarlos en la IS. También se consideran trabajos realizados por la comunidad científica internacional sobre caracterizaciones de los trabajos investigativos que se realizan en ingeniería de software [44][45]. Los atributos de calidad deseables de este tipo de trabajo [46][47][48] y las bases conceptuales para la realización de diseños de investigación en el área [49][50].

Se definió entonces un modelo de las actividades de investigación, identificando las estrategias de trabajo que se aplicarían para desarrollar las tareas investigativas y estableciendo también algunas áreas conceptuales en donde es necesario lograr mayores definiciones.

2.5.1. Modelo de Investigación en Ingeniería del Software: Una propuesta de investigación tecnológica

[15] Grupo Laboratorio de Investigación para el Desarrollo de Ingeniería de Software (LIDIS) - Jaime A. Chavarriaga L. Hugo F. Arboleda J. -Universidad San Buenaventura, Cali, Colombia - jaime,huarbole@usb.edu.co

A continuación se describen las etapas definidas por el Grupo de estudio LIDIS para la Metodología de Investigación en Ingeniera del Software [8]:

La investigación y desarrollo inicial

Representa el comienzo de cualquier iniciativa o línea de investigación. En ella se delimita el área de trabajo investigativo que se busca desarrollar en la iniciativa. Normalmente surge a partir de una serie de lluvia de ideas en las cuales se busca determinar las principales necesidades de la industria o las tendencias de la tecnología que deben ser analizadas. En algunos casos, puede surgir también a partir de una solución innovadora e interesante encontrada en la industria o por alguno de los investigadores.

Para el desarrollo de esta etapa, normalmente se requieren varios tipos de investigaciones: diagnósticos, revisiones del estado del arte, investigaciones académicas y desarrollo de soluciones en el laboratorio (pruebas de concepto). Una iniciativa culminará esta etapa cuando pueda establecer, por lo menos en ambientes controlados, la factibilidad técnica de una solución al problema propuesto.

En algunos centros de investigación existen procedimientos definidos para establecer las nuevas iniciativas o líneas de investigación. Algunas de estas, de hecho, pueden no superar la etapa inicial y no desarrollarse a través de proyectos piloto en empresas reales.

En algunos casos se define una etapa de estudio de factibilidad que puede durar unos ocho o nueve meses antes de constituirse una nueva iniciativa.

La investigación aplicada

Es la etapa en donde se busca madurar la tecnología definida en la etapa inicial, trabajando en el desarrollo de la tecnología y en su aplicación en situaciones reales. A medida que se tienen experiencias de aplicación de la tecnología por parte del grupo de investigación se encuentran las posibles fallas en el modelo de solución propuesto y se definen las adaptaciones que puedan ser requeridas para su aplicación en empresas. En esta etapa normalmente se desarrollan proyectos investigativos de naturaleza más empírica y aplicada, en muchos casos siguiendo también esquemas de investigación-acción-participativa. Una iniciativa culminará esta etapa cuando pueda configurar una solución madura a alguna problemática en ingeniería de software, incluyendo consideraciones sobre las limitaciones e implicaciones de su implementación en empresas reales que realicen procesos de desarrollo de software.

La transferencia

Etapa que busca amplificar el impacto de las nuevas tecnologías, prácticas y conocimientos, más allá de las empresas con las cuales se trabajó en el proceso de investigación aplicada. Para cumplir su objetivo, los grupos de investigación desarrollan otros tipos de actividades, tales como cursos, conferencias, y licenciamiento de la tecnología.

En la Ilustración 16. Etapas de la Metodología de Investigación se pueden observar las etapas con cada una de las acciones que pueden desarrollarse en estas.

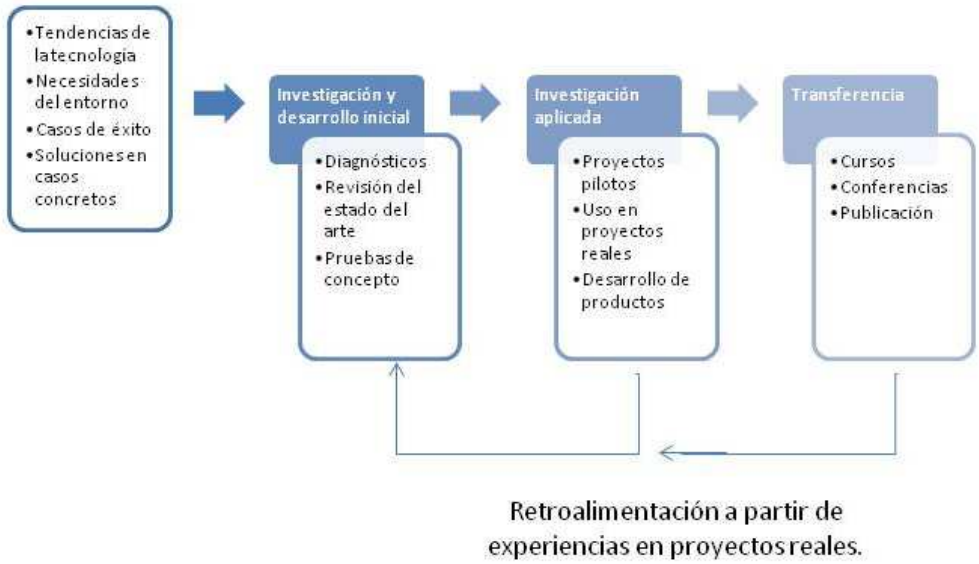


Ilustración 16. Etapas de la Metodología de Investigación

Capítulo 3:

Desarrollo de una aplicación móvil de test de orientación vocacional

3.1. Introducción

Como se ha mencionado en el Capítulo 1, el objetivo del presente Trabajo Final es desarrollar una aplicación móvil que sea utilizada como un recurso que contribuya en el proceso de orientación vocacional de la UNCA. Con su implementación se busca además ampliar la cantidad de alumnos del Nivel Secundario que accedan a la información sobre la oferta académica de la Universidad y también sobre los servicios ofrecidos tanto por la Dirección de Orientación Vocacional como por la Secretaría de Bienestar Universitario y Asuntos Estudiantiles. Considerando que gran parte de la población cuenta con un teléfono móvil con SO Android, se puede asegurar la usabilidad de la aplicación por parte de los interesados en realizar estudios de grado y pre-grado. Para el desarrollo de la aplicación móvil se empleó la metodología mencionada en la sección 2.4.5 del presente trabajo. La aplicación se desarrolló utilizando los lenguajes Kotlin y XML utilizando el compilador Dex D8 en el IDE Android Studio en su versión 3.1.

El desarrollo de la aplicación móvil corresponde a la etapa “La investigación aplicada” de la metodología de investigación en Ingeniería de Software.

3.2. Análisis

Siguiendo los lineamientos la metodología mencionada para el desarrollo de las aplicaciones móviles, en esta fase se realizaron las siguientes actividades:

3.2.1. Obtención de requerimientos

Para la obtención de los mismos se realizaron reuniones con el personal de la Dirección de Orientación Vocacional y con el Secretario de Bienestar Universitario y Asuntos Estudiantiles pertenecientes a la UNCA. Se indaga sobre las acciones llevadas a cabo en el proceso de orientación vocacional y se plantea la posibilidad de utilizar una aplicación móvil en el proceso de orientación vocacional como una herramienta de apoyo a utilizarse en las actividades del Programa UNCA+cerca, en los procesos de orientación vocacional y también como una herramienta de promoción de carreras.

Primera Reunión

Se realizó en las instalaciones de la Dirección de Orientación Vocacional sito Calle Prado 371, el día viernes 29 de junio de 2018 con las psicopedagogas pertenecientes a dicha Dirección: Lic. María Estefanía Morales y la Lic. Lorena Chávez López, la Directora de la presente tesis Dra. Ivanna Lazarte y quien suscribe alumna Andrea Segura.

En la misma se indaga sobre:

- Proceso que se lleva a cabo para realizar el proceso de orientación vocacional:
    - Etapas
    - Tipos de actividades
    - Utilización de test psicológicos

  - Tipos de test que se utilizan
  - Análisis de los mismos
  - Factores que se consideran que influyen en la elección de una carrera (familiar, actitudinal, económicos, etc.)

- Opinión sobre la utilización de un test genérico de intereses
- Actividades que realiza la Dirección de Orientación Vocacional
- Actividades que se realizan en conjunto con otros programas (UNCa + Cerca)
- Se plantea la inclusión de una aplicación móvil de test de orientación vocacional en el proceso de orientación vocacional.
- Se plantean posibles escenarios para la utilización de una aplicación móvil como herramienta adicional a las ya utilizadas.
- Se acuerda sobre la posibilidad de personalizar un test genérico de orientación vocacional.

Segunda Reunión

Se realizó en las instalaciones de la Dirección de Orientación Vocacional sito Calle Prado 371, el día viernes 13 de julio de 2018. Se llevó a cabo en presencia del Ing. Ahumada encargado de la Secretaria de Bienestar Estudiantil de la cual depende la Dirección de Orientación Vocacional, y también de las psicopedagogas Licenciadas María Estefanía Morales y Lorena Chávez López pertenecientes a dicha dirección. También estuvo presente la directora de la presente tesis Dra. Ivanna Lazarte y quien suscribe alumna Andrea Segura.

- Se indaga sobre las funcionalidades que debiera realizar la aplicación móvil.
- Se presentan los beneficios de la utilización de este tipo de herramienta: disponibilidad de datos sobre la oferta académica, alcanzabilidad en la distribución de estos, tabulación y obtención de resultados de forma inmediata, etc.
- Se hace entrega de distintos test genéricos para su análisis por parte de la Dirección de Orientación Vocacional.
- Se escoge un test a ser utilizado en la aplicación móvil para un análisis más profundo por parte de las licenciadas de la Dirección, el mismo corresponde a un test de intereses, el cual se basa en un listado de actividades (ochenta) que el usuario debe ir seleccionando de acuerdo a su interés. Sobre este se plantea la posibilidad de utilizar un número menor de actividades al test original y de adecuarlas al entorno vivencial del usuario.
- Se plantea la necesidad de contar con información sobre actividades, datos de contacto de las distintas entidades: Facultades, Direcciones, Secretaria, etc. de la UNCA para mostrarse dentro de la aplicación.

.

Tercera Reunión

Se realizó en las instalaciones de la Dirección de Orientación Vocacional sito Calle Prado 371, el día viernes 21 de septiembre de 2018. Se realiza con las psicopedagogas dependientes de la Dirección de Orientación Vocacional: Lic. María Estefanía Morales y la Lic. Lorena Chávez López y quien suscribe alumna Andrea Segura.

En la misma se realizó:

- Entrega por parte de la Dirección de Orientación Vocacional de las actividades modificadas para que sean acordes al léxico de nuestra región y los jóvenes destinatarios de la aplicación.
- Se acuerda conservar la cantidad de actividades del test original: 80 (ochenta).
- Entrega de descripción sobre las actividades que realiza la Dirección de Orientación Vocacional.

3.2.2. Clasificar los requerimientos

Luego de analizar la información obtenida en las reuniones realizadas con las psicopedagogas de la Dirección de Orientación Vocacional se logra la obtención de los requerimientos, los cuales se clasifican según lo especifica la metodología de desarrollo propuesta en el inciso 2.4.5.

Entorno

Este requerimiento hace referencia a lo que rodea al servicio, como las características técnicas del dispositivo móvil del usuario, que en este caso se requerirá un dispositivo de gama media, los cuales cuentan con un mínimo de 2 GB de RAM y almacenamiento interno que va desde los 8GB, extensible mediante la utilización de tarjeta de memoria. La aplicación estará desarrollada para el Sistema Operativo Android desde su versión Marshmallow 6.0 – 6.0.1 API 23 descripta anteriormente en el inciso 2.3.3.

Como se mencionó en el inciso 1.1 la aplicación móvil de test para orientación vocacional se desarrolla en marco del Proyecto de I+D “Desarrollo de una aplicación móvil de test de orientación vocacional para la Universidad Nacional de Catamarca". El mismo contempla el desarrollo de una aplicación web además de la aplicación móvil descripta en el presente trabajo final. Cabe aclarar que el desarrollo de la aplicación web se encuentra fuera del alcance del presente proyecto. Se describe la función de la aplicación web a modo de clarificar el proceso de envío, almacenamiento y procesamiento de los datos obtenidos en la aplicación móvil de test para orientación vocacional.

La aplicación web será desarrollada en el lenguaje de programación PHP versión 7.4 utilizando el Framework Laravel en su versión 5.6. Será alojada en un servidor Apache versión 2.4 y utilizará MySQL versión 5.7 como gestor de base de datos. La función principal de la aplicación web será recibir los datos enviados en formato JSON desde la aplicación móvil. Los datos receptados serán almacenados en una base de datos. Otras funciones de la aplicación web serán:

- Mostrar estadísticas básicas, como por ejemplo, porcentaje de resultados positivos para cada área.
- Posibilitar la descarga de un archivo de tipo .xls (Excel).

Mundo

Como se trata de una aplicación móvil que responde a una institución educativa de nivel universitario que ya cuenta con presencia en la Web mediante su página Web oficial[16], y también en las redes sociales Facebook, Twitter e Instagram, se seguirán los patrones de colores utilizados en todos estos medios. De esta manera, la aplicación móvil se observará como una herramienta más de comunicación de la institución.

Requerimientos Funcionales

A continuación, se describen los requerimientos funcionales de la aplicación, es decir todos aquellos que demandan una función dentro del sistema.

- Proveer de una aplicación móvil de test de orientación vocacional a la Dirección de Orientación Vocacional que tendrá como resultados posibles alguna de las siguientes áreas que se detallan en la Tabla 6 de acuerdo a las preferencias, gustos y aptitudes del usuario:

Área	Nombre
I	Arte y creatividad
II	Ciencias Sociales
III	Económica y Administrativa Financiera
IV	Ciencia y Tecnología
V	Ciencias Ecológicas, Biológicas y de Salud

Tabla 6. Áreas resultantes del Test de Orientación Vocacional

- Brindar información al usuario sobre la oferta académica de las carreras de grado y pre- grado que proporciona la UNCA en cada una de las áreas mencionadas anteriormente.

- Brindar información al usuario sobre las actividades que se llevan a cabo en la Dirección de Orientación Vocacional de la UNCA.

- Recopilar datos de los usuarios que permitan realizar estadísticas sobre los resultados del test de orientación vocacional, como por ejemplo cantidad de usuarios por áreas.

[16] www.unca.edu.ar

- Brindar información al usuario sobre la Secretaría de Bienestar Universitario y Asuntos Estudiantiles, relacionadas a los servicios y las becas que se ofrecen.

- Mostrar en un mapa las locaciones de las distintas unidades de la UNCA.

Requerimientos No Funcionales

En la siguiente tabla se pueden observar los requerimientos no funcionales de la aplicación móvil para Test de Orientación Vocacional.

Atributo	Detalle
Plataforma Mínima	Android versión 5.1
Tipo de aplicación	Aplicación móvil
Lenguaje de programación	Kotlin y XML
Gestor de base de datos	Para el desarrollo de esta aplicación no se utiliza un GBD
Herramienta de desarrollo	Android Studio
Tiempo de respuesta	Menor a 2 segundos para cualquier operación
Disponibilidad	Siempre disponible, para acceder a redes sociales, enviar datos al servidor y para comunicarse mediante el envió de un email se requiere conexión a internet.
Mantenibilidad y Portabilidad	1. Disponibilidad para dispositivos móviles Android a partir de la versión 5.1. 2. Será necesario disponer de una conexión a internet, ya sea por WiFi o por tarifa de datos, para la transferencia de los resultados del test. 3. La aplicación deberá ser lo más ligera posible para ocupar el mínimo espacio en la memoria del teléfono
Interfaz y usabilidad	1. La aplicación debe constar de una interfaz simple e intuitiva. 2. La introducción de datos debe estar estructurada guiando al usuario en la realización del test.

Tabla 7. Requerimientos no funcionales

3.2.3. Personalizar el servicio

Se propone el desarrollo de una aplicación liviana que no ocupe mucho espacio en la memoria del teléfono móvil, considerando que su utilización no será diaria.

Se buscará un diseño visual ligero, intuitivo, utilizando colores claros para los fondos, textos y colores fuertes para iconos y botones. Estos colores serán elegidos de acuerdo a la gama utilizada en los demás medios oficiales de la UNCA.

La información que se brindará será extraída de las páginas oficiales de la UNCA como así también de las siete facultades: Facultad de Ciencias Agrarias, Facultad de Ciencias de la Salud, Facultad de Derecho, Facultad de Tecnología y Ciencias Aplicadas, Facultad de Ciencias Exactas y Naturales, Facultad de Humanidades, Facultad de Ciencias Económicas y de Administración; y de la Escuela de Arqueología.

En cuanto a la elaboración del test de orientación vocacional que se utilizará, se trabaja en conjunto a la Dirección de Orientación Vocacional, tomando como base un test genérico del cual se conservará la lógica de ejecución para la obtención del resultado. Las preguntas serán modificadas adecuándolas al medio, es decir, considerando el entorno, la región y la edad de los usuarios que realizarán dicho test.

Esta modificación se plantea teniendo en cuenta que algunas actividades no se adecuan a la realidad de los jóvenes de esta provincia, además el modo en que están redactadas no es la indicada considerando que los potenciales usuarios son adolescentes. En la Tabla 8. Actividades del test de Orientación Vocacional se muestran las actividades que formarán parte del test. Cada una de estas actividades se presentan al usuario y éste debe ir indicando si le gusta o no para avanzar en el test.

N°	Actividad
1	Diseñar programas de computación y explorar nuevas aplicaciones tecnológicas para uso del internet.
2	Criar y cuidar animales domésticos y/o de campo.
3	Investigar sobre espacios verdes, medio ambiente y cambios climáticos.
4	Ilustrar, dibujar y animar digitalmente
5	Seleccionar, capacitar y motivar al personal de una organización/empresa.
6	Realizar excavaciones para descubrir restos del pasado
7	Resolver problemas de cálculo para construir un puente
8	Diseñar cursos para enseñar a la gente sobre temas de salud e higiene
9	Tocar un instrumento y/o componer música.
10	Planificar cuáles son las metas de una organización pública o privada a mediano y largo plazo.
11	Diseñar y planificar la producción masiva de artículos como muebles, autos, equipos de oficina, empaques y envases para alimentos y otros.
12	Diseñar logotipos y portadas de una revista.
13	Organizar eventos y atender a sus asistentes.
14	Atender la salud de personas enfermas.
15	Controlar ingresos y egresos de fondos y presentar el balance final de una institución.
16	Hacer experimentos con plantas (frutas, arboles, flores, etc).
17	Confeccionar planos para la realización de viviendas y/o edificios.
18	Investigar y probar nuevos productos farmacéuticos.
19	Hacer propuestas y formular estrategias para fomentar las relaciones económicas entre dos países
20	Pintar, hacer esculturas, ilustrar libros de arte, etc.

21	Elaborar campañas para introducir un nuevo producto al mercado.
22	Examinar y tratar los problemas visuales.
23	Defender clientes individuales o empresas en juicios de diferente naturaleza.
24	Diseñar maquinas que puedan simular actividades humanas.
25	Investigar las causas y efectos de trastornos emocionales como la depresión o ansiedad.
26	Supervisar las ventas en un centro comercial.
27	Colaborar en la rehabilitación de personas que tienen limitaciones físicas.
28	Prepararse para ser modelo profesional.
29	Aconsejar a las personas sobre planes de ahorro e inversiones.
30	Elaborar mapas, planos e imágenes para el estudio y análisis de datos geográficos.
31	Diseñar juegos interactivos electrónicos para computadora.
32	Realizar el control de calidad de los alimentos.
33	Ser dueño de un comercio.
34	Escribir artículos periodísticos, cuentos, novelas y otros.
35	Redactar guiones y libretos para un programa de televisión.
36	Organizar un plan de distribución y venta de un gran almacén.
37	Estudiar la diversidad cultural en el ámbito rural y urbano.
38	Gestionar y evaluar convenios (acuerdos) internacionales de cooperación entre países para su desarrollo social.
39	Crear campañas publicitarias.
40	Trabajar investigando la reproducción de peces, camarones y otros animales marinos.
41	Dedicarse a fabricar productos alimenticios de consumo masivo
42	Gestionar y evaluar proyectos de desarrollo en una institución educativa y/o fundación.
43	Rediseñar y decorar espacios físicos en viviendas, oficinas y locales comerciales.
44	Administrar una empresa de turismo y/o agencias de viaje.
45	Aplicar métodos alternativos a la medicina tradicional para atender personas con dolencias de diversa índole.
46	Diseñar ropa para niños, jóvenes y adultos.
47	Investigar organismos vivos para elaborar vacunas.
48	Manejar y/o dar mantenimiento a dispositivos/aparatos tecnológicos en aviones, barcos, radares, etcétera
49	Estudiar idiomas extranjeros –actuales y antiguos- para hacer traducción.
50	Restaurar piezas y obras de arte
51	Revisar y dar mantenimiento a artefactos eléctricos, electrónicos y computadoras.
52	Enseñar a niños de 5 años.
53	Investigar y/o sondear nuevos mercados.
54	Atender la salud dental de las personas
55	Tratar a niños, jóvenes y adultos con problemas psicológicos.
56	Se refiere al Marketing o las Ventas, pero es demasiado específico, hay que hacerlo más general, por ejemplo: Crear estrategias de promoción y venta de productos regionales en el mercado internacional.

57	Planificar y recomendar dietas para personas diabéticas y/o con sobrepeso.
58	Trabajar en una empresa petrolera en un cargo técnico como control de la producción.
59	Administrar una empresa (familiar, privada o pública)
60	Tener un taller de reparación y mantenimiento de carros, tractores, etcétera.
61	Ejecutar proyectos de extracción minera y metalúrgica.
62	Asistir a directivos de empresas multinacionales con la importación y exportación de productos.
63	Diseñar programas educativos para niños con discapacidad
64	Aplicar conocimientos de estadística en investigaciones en diversas áreas(social, administrativa, salud, etcétera.)
65	Sacar fotografías para publicitar un producto o servicio, para una muestra artística o para representar la noticia de un periódico.
66	Trabajar en museos y bibliotecas nacionales e internacionales.
67	Ser parte de un grupo de teatro.
68	Producir cortometrajes, spots publicitarios, programas educativos, de ficción, etcétera.
69	Estudiar la influencia entre las corrientes marinas y el clima y sus consecuencias ecológicas.
70	Conocer las distintas religiones, su filosofía y transmitirlas a la comunidad en general.
71	Asesorar a inversionistas en la compra de bienes/acciones en mercados nacionales e internacionales.
72	Explorar el espacio exterior (fuera de la atmosfera terrestre), los planetas, sus características y componentes.
73	Explorar el espacio sideral, los planetas, características y componentes.
74	Mejorar la imagen facial y corporal de las personas aplicando diferentes técnicas.
75	Decorar jardines de casas y parques públicos.
76	Administrar y renovar menúes de comidas en un hotel o restaurante.
77	Trabajar como presentador de televisión, locutor de radio y/o animador de programas de televisión.
78	Diseñar y ejecutar programas de turismo.
79	Planificar adecuadamente la ocupación del espacio físico de ciudades, países etc., utilizando imágenes de satélite, mapas.
80	Organizar, planificar y administrar instituciones educativas.

Tabla 8. Actividades del test de Orientación Vocacional

3.3. Diseño

3.3.1. Definir el Escenario

El término escenario refiere al estado conexión con el servidor, a partir del sistema de conexión y sincronización con este. En este caso, la sincronización se realizará para enviar desde la aplicación móvil al servidor el resultado del test de orientación vocacional y datos

del usuario: edad, procedencia, escuela secundaria, sexo, email, fecha y área. Datos que posteriormente serán procesados para la obtención de estadísticas dentro del servidor.

Se optó por un escenario semi-conectado, en el cual los procesos podrán ejecutarse con el dispositivo móvil desconectado, pero se requiere establecer conexión con el servidor en algún momento para la finalización del procedimiento, es decir, para la inserción de la información recabada del usuario en la base de datos.

3.3.2. Estructurar el software

Desde la metodología elegida para el desarrollo del presente trabajo se sugiere utilizar distintos esquemas para modelar el sistema en distintas perspectivas de acuerdo a sus necesidades. Se especificarán dos modelos: de Uso y de Interacción.

Modelo de Uso

El Modelo de Uso describe cada uno de los requisitos funcionales del software, por medio de diagramas de casos de uso. Se utilizó el software Dia[17], en su versión 0.97.2, para la creación de diagramas estructurados.

En la Ilustración 17, se observa el caso de uso corresponde a la aplicación móvil completa.

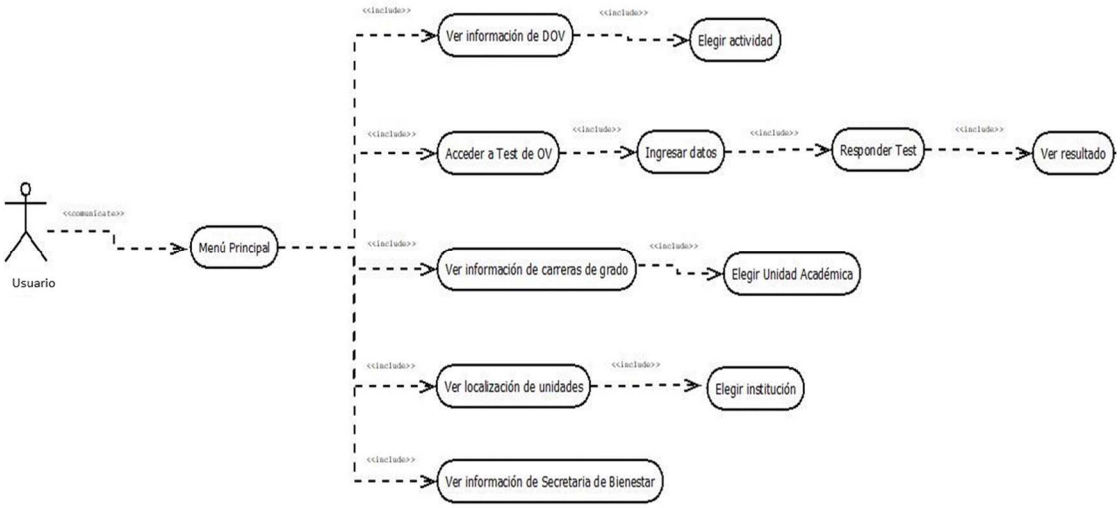


Ilustración 17. Modelo de Caso de uso de toda la aplicación móvil.

Las opciones que se presentan en el menú principal son las que determinan los casos de uso que se analizarán a continuación, como así también aquellos casos de uso que surgen en el proceso de realización del test de orientación vocacional.

[17] http://dia-installer.de/shapes/UML/index.html.es

Descripción del Caso de Uso: Ver información de DOV

El usuario accede a visualizar información sobre las actividades que realiza la Dirección de Orientación Vocacional.

Flujo de sucesos

Iniciador	**Usuario**
Precondición	No tiene.
Camino básico	1. El usuario solicita ingresar. 2. El usuario elige una de las actividades enlistadas. 3. El usuario visualiza una descripción de dicha actividad.
Pos condición	El usuario visualiza las distintas actividades realizadas por la DOV.

Tabla 9. Caso de uso Ver información de DOV.

Descripción del Caso de Uso: Acceder a Test de OV

El usuario ingresa a realizar el test de orientación vocacional.

Flujo de sucesos

Iniciador	**Usuario**
Precondición	No tiene.
Camino básico	1. El usuario ingresa a la opción de Test de OV. 2. El usuario completa los datos obligatorios. 3. El usuario responde el test marcando como mínimo 20 actividades. 4. El usuario solicita ver el resultado del test. 5. El usuario visibiliza el resultado del test.
Pos condición	El usuario obtiene el resultado del Test de OV.

Tabla 10. Caso de uso Acceder a Test de OV.

Descripción del Caso de Uso: Ingresar Datos

El usuario debe completar los campos obligatorios para poder acceder a responder el test de Orientación Vocacional.

Flujo de sucesos

Iniciador	**Usuario**
Precondición	Acceder a la opción Test de OV
Camino básico	1. El usuario completa los campos obligatorios. 2. El usuario presiona el botón Siguiente. 3. El usuario accede al test de Orientación Vocacional.
Camino Alternativo 1	1. El usuario presiona el botón volver atrás.

	2. El usuario vuelve al menú principal.
Pos condición	El usuario puede responder el test de Orientación Vocacional.

Tabla 11. Caso de Uso Ingresar Datos

Descripción del Caso de Uso: Responder Test

El usuario debe responder el test de Orientación Vocacional eligiendo un mínimo de 20 (veinte) actividades que sean de su interés.

Flujo de sucesos

Iniciador	**Usuario**
Precondición	Completar los campos obligatorios en Ingresar Datos.
Camino básico	1. El usuario escoge por lo menos 20 actividades del test de Orientación Vocacional.
Camino Alternativo 1	1. El usuario escoge menos de 20 actividades del test de Orientación Vocacional. 2. El usuario no puede obtener el resultado del test de Orientación Vocacional y las carreras afines.
Pos condición	El usuario obtiene el resultado del test de Orientación Vocacional.

Tabla 12. Caso de Uso Responder Test.

Descripción del Caso de Uso: Ver resultado

El usuario luego de escoger por lo menos 20 (veinte) actividades del test de Orientación Vocacional solicita ver el resultado del mismo, el cual es una de las siguientes áreas: Arte y creatividad, Ciencias Sociales, Economía y administración, Tecnología, Ecológicas, biológicas y de salud.

Flujo de sucesos

Iniciador	**Usuario**
Precondición	Escoger por lo menos 20 (veinte) actividades del test de orientación vocacional.
Camino básico	1. El usuario solicita ver el resultado luego de haber completado el test en su totalidad. 2. El usuario obtiene el resultado del test y las carreras afines.
Pos condición	El usuario obtiene el resultado del test.

Tabla 13. Caso de Uso Ver Resultado.

Descripción del Caso de Uso: Ver localización de unidades

El usuario ingresa a esta opción donde puede visualizar las localizaciones de las distintas unidades de la UNCA.

Flujo de sucesos

Iniciador	**Usuario**

Precondición	No tiene.
Camino básico	1. El usuario accede a esta opción. 2. EL usuario elige la unidad que quiere visibilizar en el mapa. 3. El usuario será georeferenciado por la aplicación en el punto que este elogió visualizar en el mapa.
Pos condición	El usuario obtiene información sobre la localización de las unidades.

Tabla 14. Caso de uso Ver localización de unidades.

Descripción del Caso de Uso: Ver información de Secretaria de Bienestar

El usuario ingresa y puede acceder a información sobre la Secretaria de Bienestar Universitario.

Flujo de sucesos

Iniciador	**Usuario**
Precondición	No tiene
Camino básico	1. El usuario ingresa a esta opción. 2. El usuario visualiza la información de la Secretaria de Bienestar Universitario.
Pos condición	El usuario obtiene información sobre la Secretaria de Bienestar Universitario.

Tabla 15. Caso de uso Ver información de Secretaría de Bienestar.

Modelo de Interacción

En la Ilustración 18. Diagrama de Secuencia para Proceso de Responder Test., se representa el proceso que debe llevar a cabo el usuario para poder obtener el resultado del test de orientación vocacional. En el mismo se ven implicados los casos de uso: Acceder a Test de OV, Ingresar Datos, Responder Test y Ver Resultado. Se realiza este diagrama de secuencia entendiendo que dicha herramienta permite visualizar de manera más clara la interacción cronológica entre los objetos que participan en el proceso de obtención del resultado del test. Se realiza el diagrama de secuencia puntualmente de este proceso dado que es el objetivo principal del presente trabajo.

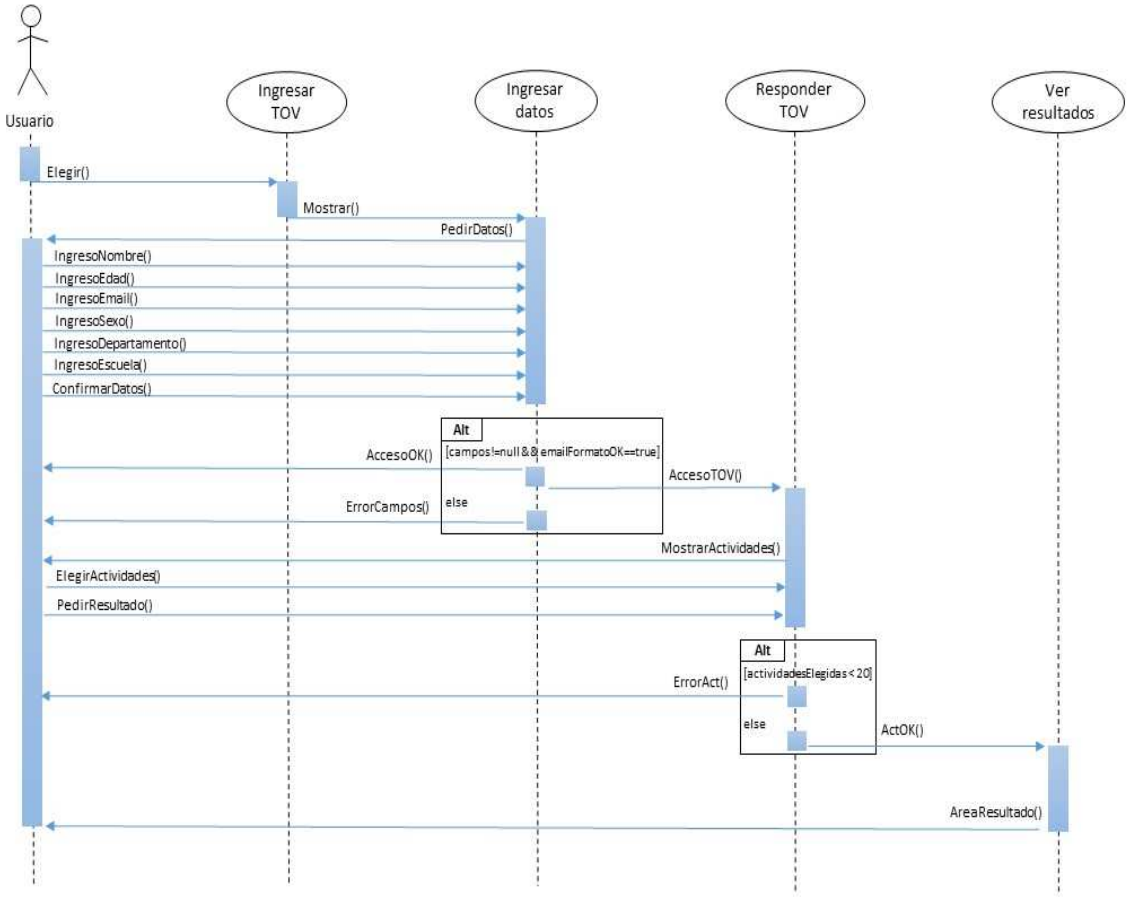


Ilustración 18. Diagrama de Secuencia para Proceso de Responder Test.

En el diagrama de secuencia de la Ilustración 18. Diagrama de Secuencia para Proceso de Responder Test. Existe un actor: Usuario y cinco objetos: AccederTOV, IngresarDatos, ResponderTest y VerResultado.

El usuario deberá escoger la opción Test de Orientación Vocacional en el Menú Principal. Luego de esto la aplicación le pedirá que ingrese sus datos personales. El usuario deberá ingresar: nombre, edad, email, sexo, escuela, departamento y luego confirmar los datos. Si todos los datos fueron ingresados correctamente se permitirá el acceso al test de orientación vocacional. El usuario deberá elegir de acuerdo a su interés no menos de veinte actividades y luego solicitar una respuesta. La aplicación verifica que el usuario cumpla con la cantidad mínima de actividades seleccionadas; si esto ocurre se mostrará al usuario el área resultante y las carreras afines. De no ser así, muestra al usuario un mensaje de error indicando que no llegó a la cantidad mínima de actividades escogidas.

3.3.3. Estructura de la tabulación del Test de Orientación Vocacional

Para la construcción del test de orientación vocacional de la aplicación móvil se utilizará un test de identificación de intereses vocacionales y profesionales elaborados por las

psicólogas Malca de Goldenberg [18]y Magali Merchán[19]. Se elige este test por la sencillez en su tabulación y porque se considera que las áreas resultantes permiten abarcar todas las carreras ofrecidas en la UNCA, además de mostrar cual es la población que no se está abarcando en dicha oferta al contar con el área de Arte y Creatividad como parte de los resultados posibles.

En la siguiente tabla se puede observar que todas las áreas tienen el mismo número de actividades.

AREAS	PREGUNTAS																TOTAL
ÁREA I Arte y Creatividad	4	9	12	20	28	31	35	39	43	46	50	65	67	68	75	77	
ÁREA II Ciencias Sociales	6	13	23	25	34	37	38	42	49	52	55	63	66	70	72	78	
ÁREA III Económica, Administrativa y Financiera	5	10	15	19	21	26	29	33	36	44	53	56	59	62	71	80	
ÁREA IV Ciencia y Tecnología	1	7	11	17	18	24	30	41	48	51	58	60	61	64	73	79	
ÁREA V Ciencias Ecológicas, Biológicas y de Salud	2	3	8	14	16	22	27	32	40	45	47	54	57	69	74	76	

Tabla 16. Actividades por áreas.

El usuario debe escoger aquellas actividades que son de su interés. Cada área cuenta con 16 actividades que le pertenecen. Por cada actividad que escoja se incrementa un punto para el área que corresponda. Finalizada la lectura de todas las actividades y escogidas las de su interés, se debe considerar como resultante aquella que obtenga mayor puntuación.

3.3.4. Interfaz Gráfica

A continuación, se muestra el diseño de las pantallas que formarán parte de la aplicación y una breve explicación de las mismas.

[18] Licenciada en Psicología - Directora de Departamento Bienestar Estudiantil - Universidad Casa Grande, Guayaquil, Ecuador.
[19] Licenciada en Psicología – Docente en Universidad Casa Grande, Guayaquil, Ecuador.

Menú principal

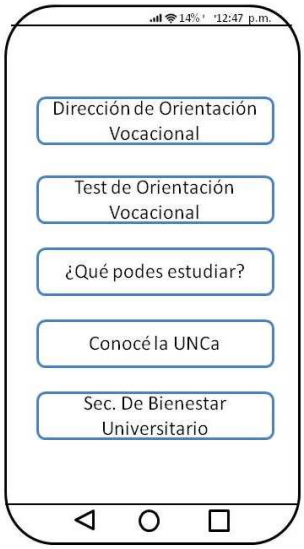


Ilustración 19. Interfaz Menú principal.

En la pantalla representada en la Ilustración 18 se encuentran cinco botones que permiten el acceso a las distintas funcionalidades de la aplicación móvil, las cuales conforman el menú principal.

.

Opción Dirección de Orientación Vocacional

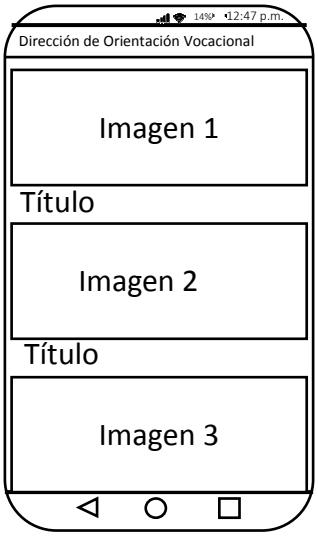


Ilustración 20. Interfaz Dirección de Orientación Vocacional.

En la Ilustración 20, se pueden observar las distintas actividades que realiza la Dirección de Orientación Vocacional, representadas mediante una imagen. Presionando en cada una de estas imágenes se puede acceder a una descripción de la misma.

Opción Test de Orientación Vocacional

En la siguiente Ilustración, se puede observar la pantalla diagramada para presentar al usuario las actividades que el podrá escoger como de su interés y luego presionar el botón

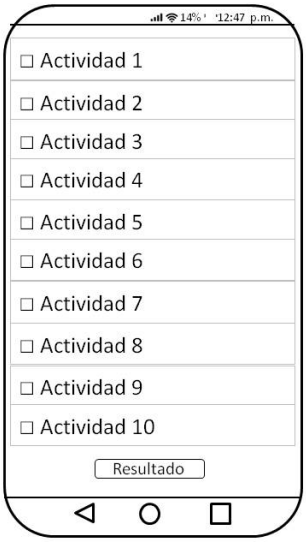


Ilustración 21. Interfaz pregunta de test

Resultado para visualizar el mismo.

Opción ¿Qué podés estudiar?

El fin principal de la Ilustración 22 es mostrar las diferentes Facultades con las que cuenta la UNCA y cuáles son las carreras de grado que se dictan en cada una de ellas.

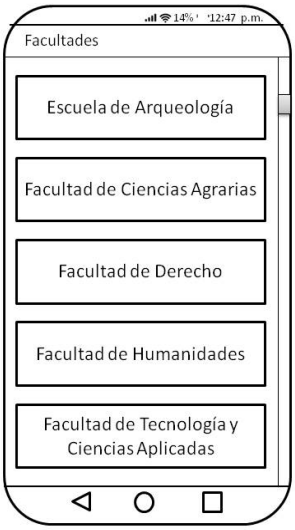


Ilustración 22. Interfaz ¿Qué podés estudiar?

Opción Conoce la UNCA

En la interfaz representada en la Ilustración 23. Interfaz conoce la UNCA. El usuario podrá visualizar un mapa en el cual por medio de un menú emergente al que se accede presionando los tres puntos del Appbar, en dicho menú podrá elegir en qué localización se ubicará el mapa. En las siguientes ilustraciones, por ejemplo se visualiza la ubicación de la Dirección de Orientación Vocacional. Para el desarrollo de esta interfaz se utilizó el SDK de MapBox el cual se describe en el anexo 6.1.1.

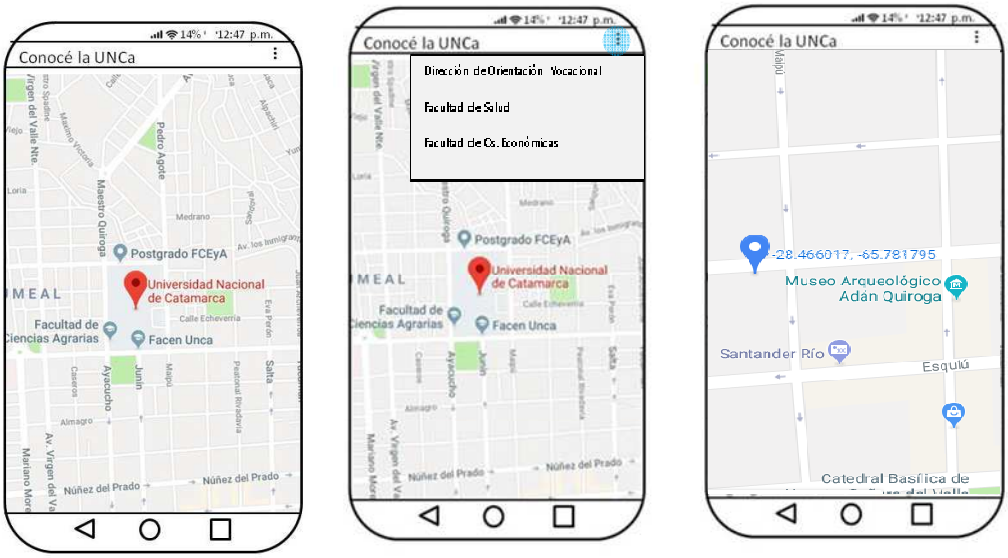


Ilustración 23. Interfaz conoce la UNCA.

Opción Secretaría de Bienestar Universitario y Asuntos Estudiantiles

En la interfaz que se muestra en la Ilustración 23, se puede ver información sobre la función de la Secretaría de Bienestar Universitario y Asuntos Estudiantiles y también sobre los servicios y becas que ofrece.

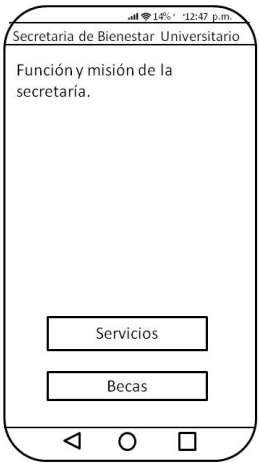


Ilustración 24. Interfaz Secretaría de Bienestar Universitario y Asuntos Estudiantiles.

3.3.5. Asignar recursos

Los recursos que se utilizaron en la elaboración de la aplicación móvil son:

- Notebook marca Sansumg Modelo Np300e5a, Procesador Intel Celeron, 4GB de RAM, al cual se adicionó una memoria de 4GB para cumplir con los requerimientos de Android Studio.
- Cable USB a Micro USB, que se utilizó para realizar las pruebas en el dispositivo móvil.
- Software Android Studio.
- Software procesador de texto.
- Servicio de Internet.
- Servicio de energía eléctrica.
- Dispositivos móviles de gama media.

Capítulo 4:

Despliegue de la aplicación móvil

4.1. Introducción

En este capítulo se describirá de qué manera se lleva a cabo el desarrollo de la aplicación móvil, la creación de un nuevo proyecto en Android Studio, las partes de un proyecto, descripción del ciclo de vida de una Activity, descripción de algunas de las clases utilizadas, y documentación del código correspondiente al test de orientación vocacional.

4.2. Codificación de la aplicación móvil

Crear un nuevo proyecto en Android Studio

Para iniciar un proyecto en Android Studio, utilizando como lenguaje de programación a Kotlin, lo primero que se debe realizar es la instalación del plugin (complemento) de dicho lenguaje. Para ello se inicia el IDE y se selecciona en la pantalla principal la opción Configure y luego en Plugins, como se muestra en la Ilustración 25. Pantalla Inicial de Android Studio.

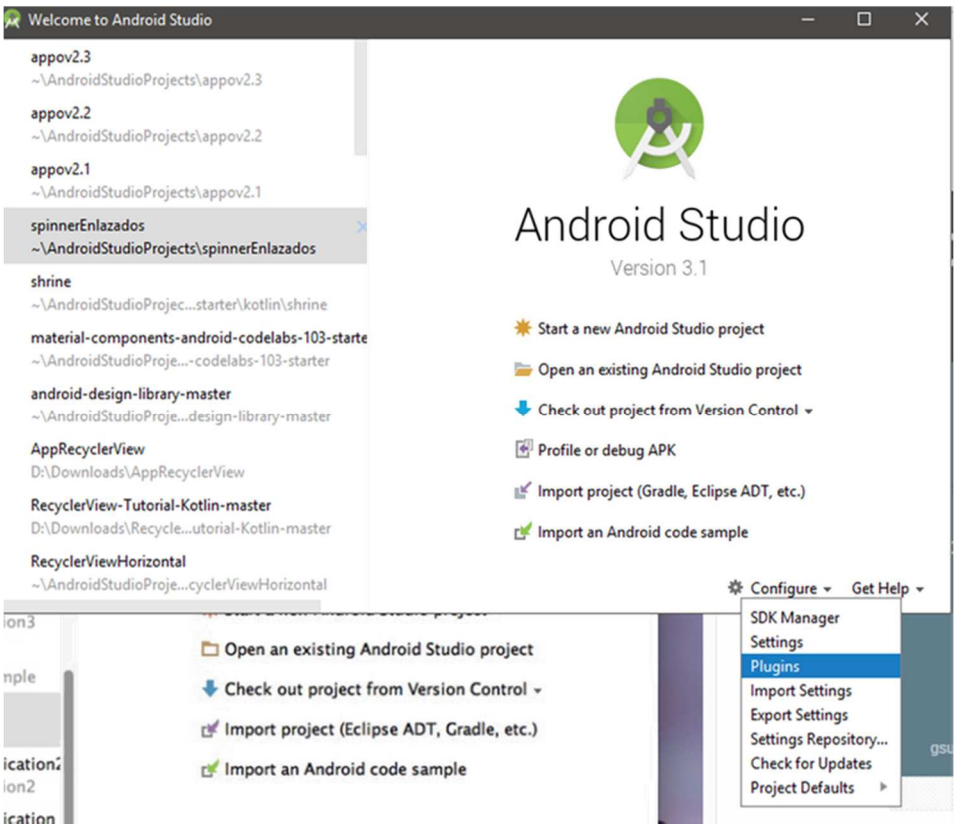


Ilustración 25. Pantalla Inicial de Android Studio.

Luego se presiona en Install JetBrains Plugins como se muestra en la Ilustración 26. Pantalla Android Studio Plugin.

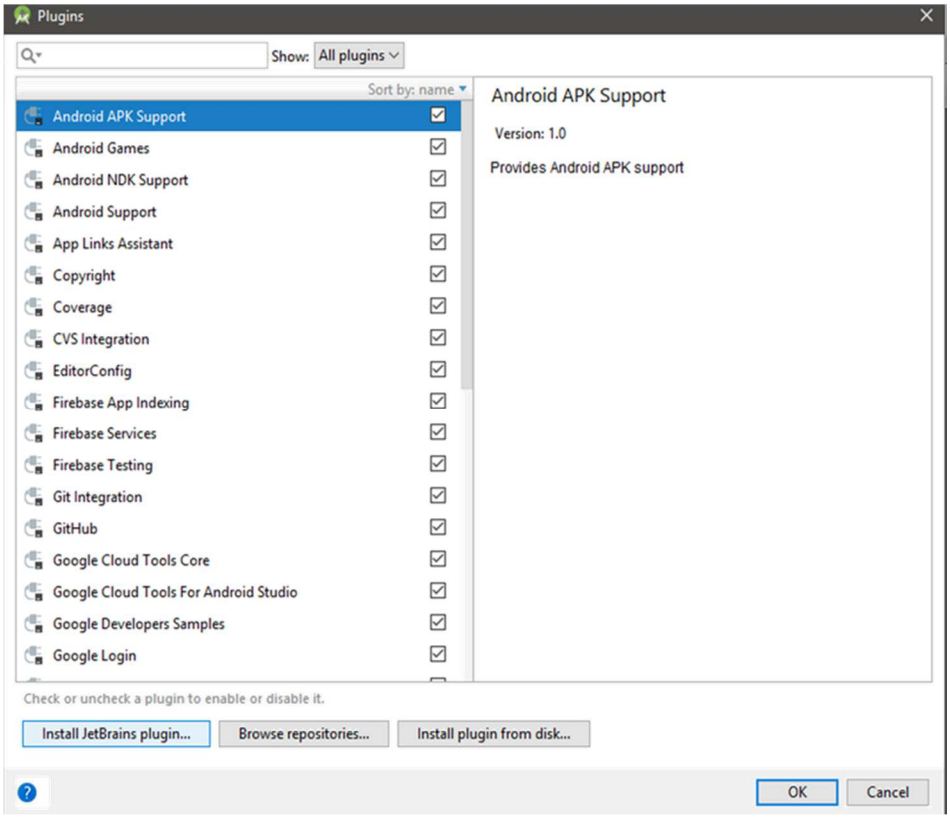


Ilustración 26. Pantalla Android Studio Plugin

Del listado de opciones se debe elegir Kotlin y presionar en instalar. Se debe reiniciar Android Studio para empezar.

A partir de esto se puede iniciar un nuevo proyecto desde la pantalla de inicio, eligiendo esa opción. Se verá la pantalla de la Ilustración 27. Pantalla Android Studio - Create Android Project, donde está la opción de incluir soporte para Kotlin y C++. En esta pantalla se puede configurar:

- **Aplication Name:** Este será el nombre de la aplicación. El nombre se puede cambiar después en el proyecto, pero es necesario para que se cree el Package name.
- **Company Domain:** Nombre de la empresa para la cual se desarrolla. El cual se puede modificar si se requiere.
- **Project Location:** Ubicación donde se guardará el proyecto.
- **Package Name:** Este package name es único, es la "firma" que identifica la aplicación. Una vez definido no se puede cambiar.

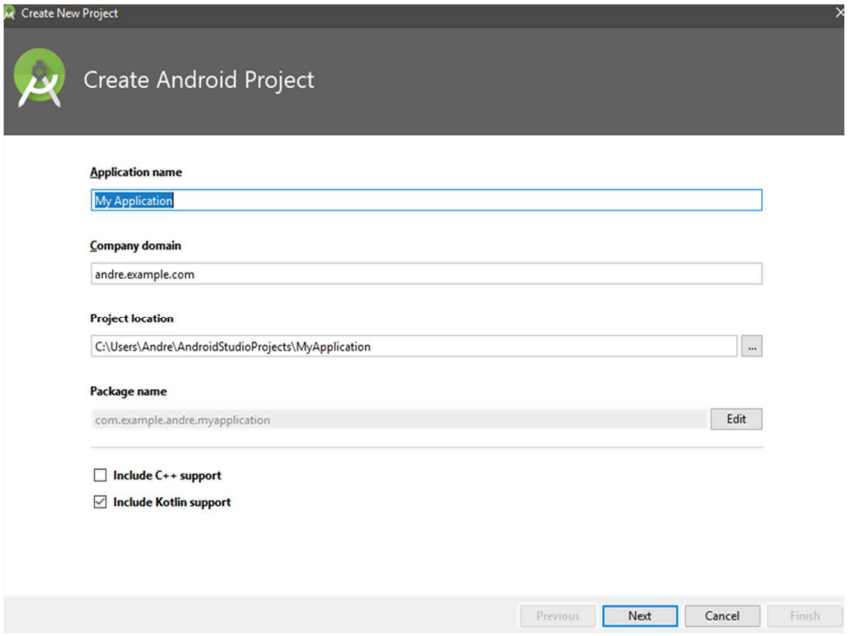


Ilustración 27. Pantalla Android Studio - Create Android Project

Luego de crear el proyecto Android se podrá elegir para qué dispositivos móviles estará diseñada la aplicación y cuál será la versión mínima de Android que admitirá. Esta pantalla se muestra en la Ilustración 28. Target Android Devices. Para esta aplicación de Test de Orientación Vocacional se utilizó como versión mínima 6.0 - 6.0.1 API 23 (Marshmallow).

Se elige el tipo de pantalla inicial, se escoge: Empty Activity (Actividad vacía) y finaliza la creación del proyecto.

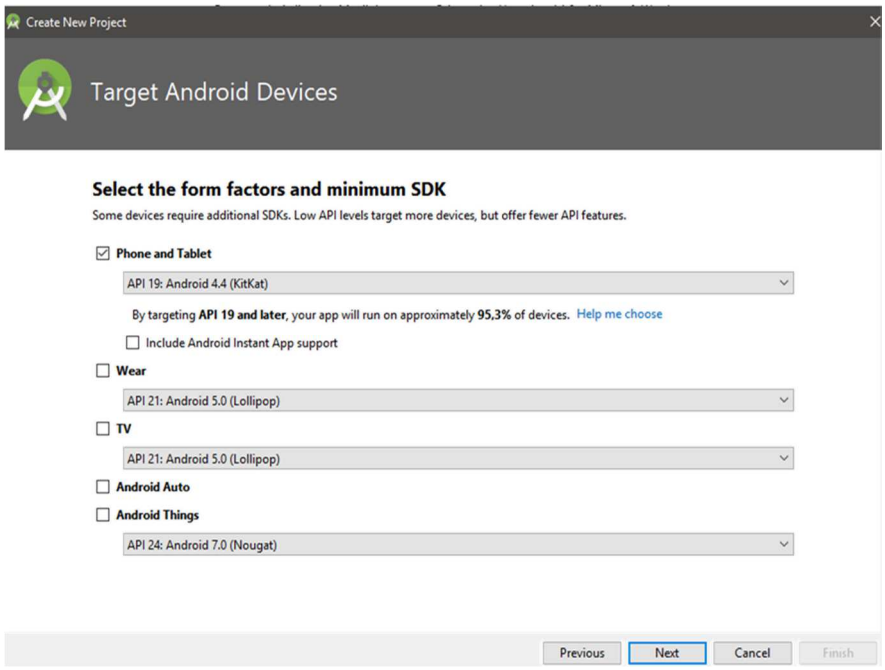


Ilustración 28. Target Android Devices

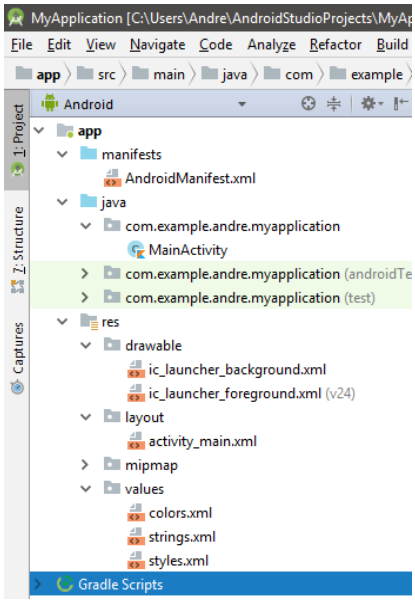


Ilustración 29. Proyecto de Android Studio - Partes.

En la Ilustración 29. Proyecto de Android Studio - Partes. Se observan las carpetas que conforman un proyecto en Android Studio: descriptor de la aplicación (Manifests), el código fuente en Kotlin (java), una serie de archivos con recursos (res) y archivos para construir el módulo (Gradle Scripts). Cada elemento se almacena en una carpeta específica, que se indicó entre paréntesis.

A continuación, en la Tabla 17. Partes de un Proyecto de Android Studio. Se realiza la descripción de cada elemento.

AndroidManifest.xml	Este archivo describe la aplicación Android. En él se indican las activitys, intents, los servicios y los proveedores de contenido de la aplicación. También se declaran los permisos que requerirá la aplicación. Se indica la versión mínima de Android para poder ejecutarla, la versión de la aplicación, etc.
java	Carpeta que contiene el código fuente de la aplicación. Como se puede observar los archivos se almacenan en carpetas según el nombre de su paquete.
MainActivity	Clase Kotlin con el código de la actividad inicial.
ApplicationTest	Clase Kotlin pensada para insertar código de testeo de la aplicación utilizando el API JUnit.

res	Carpeta que contiene los recursos usados por la aplicación.
drawable	En esta carpeta se almacenan los archivos de imágenes (JPG o PNG) y descriptores de imágenes en XML.
layout	Contiene archivos XML con las vistas de la aplicación. Las vistas permitirán configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación, es decir los layout.
menu	Archivos XML con los menús utilizados en algunas activitys. Contiene los ítems que lleva cada menú utilizado en las pantallas. Cada archivo representa un menú.
values	También se utilizan archivos XML para indicar valores usados en la aplicación, de esta manera se podrán cambiar desde estos archivos sin necesidad de ir al código fuente.
Gradle Scripts	En esta carpeta se almacenan una serie de archivos Gradle que permiten construir la aplicación.

Tabla 17. Partes de un Proyecto de Android Studio.

Activity – Ciclo de vida

Una aplicación en Android va a estar formada por un conjunto de elementos básicos de visualización, conocidos como activitys. Son estas las que realmente controlan el ciclo de vida de las aplicaciones, dado que el usuario no cambia de aplicación, sino de actividad. El sistema va a mantener una pila con las actividades previamente visualizadas, de forma que el usuario va a poder regresar a la actividad anterior pulsando la tecla "atrás". Para entender muchas funciones que tienen las actividades en Android, se deben conocer cuál es su ciclo, lo cual se puede observar en la Ilustración Ciclo de Vida de un Activity.

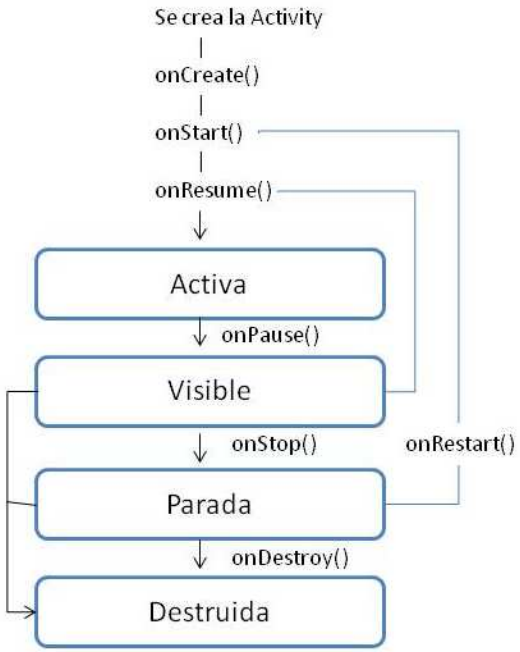


Ilustración 30. Ciclo de vida de un Activity.

Los métodos que se mencionan en la Ilustración 28 se definen a continuación:

- **onCreate(Bundle):** Se llama en la creación de la actividad. Se utiliza para realizar todo tipo de inicializaciones, como la creación de la interfaz de usuario o la inicialización de estructuras de datos.
- **onStart():**indica que la actividad está a punto de ser mostrada al usuario.
- **onResume():** Se llama cuando la actividad va a comenzar a interactuar con el usuario.
- **onPause():** Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra aplicación es lanzada.
- **onStop():** La actividad ya no es visible para el usuario. Se debe tener en cuenta que, si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.
- **onRestart():** Indica que la actividad va a volver a ser representada después de haber pasado por onStop().
- **onDestroy():** Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, cuando el usuario pulsa el botón "Volver" o cuando se llama al método finish().

4.2.1. Documentar el código

En esta sección se mostrará la documentación del código correspondiente a la activity VerCarreras.kt. En la misma se realiza la obtención del resultado del test y el envío de los datos al servidor. Se incluye esta activity considerando que este atañe a uno de los objetivos específicos del presente proyecto.

```
package com.example.andre.appov

import android.content.Intent
import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import android.widget.*
import com.android.volley.AuthFailureError
import com.android.volley.Response
import com.android.volley.VolleyError
import com.android.volley.toolbox.StringRequest
import kotlinx.android.synthetic.main.activity__carrera__unca.*
import org.json.JSONException
import org.json.JSONObject
import java.util.*

//Nombre de la Activity y de la clase de la que hereda.
class Activity_Carrera_Unca : AppCompatActivity() {

    //TODO SE DECLARA EL URL DONDE SE ALOJA EL SERVIDOR
    val addUrl : String = "http://192.168.1.10:8010/app_tov/addAlumno.php"
    private var tv: TextView? = null

    //TODO Se declara una variable para cada área resultado del test.
    private var arte_crea_area0 : Int = 0 //ARTE Y CREATIVIDAD 0
    private var sociales_area1 :Int = 0 //CIENCIAS SOCIALES 1
    private var economia_area2: Int = 0 //ECONOMIA Y ADMINISTRACION 2
    private var tecnologia_area3: Int = 0 //TECNOLOGIA 3
    private var eco_bio_salud_area4: Int = 0 //Ciencias Ecológicas,
```

```
Biológicas  y de Salud 4

    override fun onBackPressed() {
      return
    }

    //TODO Se sobreescribe el método onCreate, se declara a que layout
corresponde.
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity__carrera__unca)
        /*Se declara el textView que se utilizara para mostrar los
resultados*/
        tv = findViewById(R.id.tv) as TextView

        /*Se declara y asocia al Id button_volver al view de
id.button_volver*/
        val button_volver = findViewById<Button>(R.id.button_volver)

        /*Se utiliza un bucle for para recorrer el ArrayList que contiene
las actividades y checkbox*/
        for (i in 0 until CustomAdapterTest.public_modelArrayList!!.size) {
            /*Se selecciona solo los que están seleccionados y se extrae su
índice*/
            if
(CustomAdapterTest.public_modelArrayList!!.get(i).getSelecteds()) {
                /*De acuerdo a su índice se incrementara el valor +1 en el
área que corresponda*/
                when (i){//--i para todos por que no toma desde 0 como el
array
                    3,8,11,19,27,30,34,38,42,45,49,64,66,67,74,76 ->
{arte_crea_area0 = arte_crea_area0+1}
                    5,12,22,24,33,36,37,41,48,51,54,62,65,69,71,77 ->
{sociales_area1=sociales_area1+1 }
                    4,9,14,18,20,25,28,32,35,43,52,55,58,61,70,79 ->
{economia_area2=economia_area2 +1}
                    0,6,10,16,17,23,29,40,47,50,57,59,60,63,72,78 ->
{tecnologia_area3=tecnologia_area3+1 }
                    1,2,7,13,15,21,26,31,39,44,46,53,56,68,73,75 ->
{eco_bio_salud_area4=eco_bio_salud_area4+1}
                }
            }
        }
        /*Recorrido el ArrayList se declara j y se le asigna el valor que
devuelve la función resultadoTest()*/
        val j: Int = resultadoTest()
        /*De acuerdo al valor de j será el texto que mostrarán los textView
*/
        when(j){
            0-> {tv!!.text = getString(R.string.area1);
textView_descripcion_carrera.text=getString(R.string.DescripcionArea1)
                textView17.text= getString(R.string.AREA1)}

            1-> {tv!!.text = getString(R.string.area2) ;
textView_descripcion_carrera.text=getString(R.string.DescripcionArea2)
                textView17.text= getString(R.string.AREA2)}

            2-> {tv!!.text = getString(R.string.area3) ;
textView_descripcion_carrera.text=getString(R.string.DescripcionArea3)
                textView17.text= getString(R.string.AREA3)}
```

```
            3-> {tv!!.text = getString(R.string.area4);
textView_descripcion_carrera.text=getString(R.string.DescripcionArea4)
                textView17.text= getString(R.string.AREA4)}

            4-> {tv!!.text = getString(R.string.area5) ;
textView_descripcion_carrera.text=getString(R.string.DescripcionArea5)
                textView17.text= getString(R.string.AREA5)}
        }

     /*Al presionar el botón volver*/
     button_volver.setOnClickListener({
        /*Llama a la función que envía los datos del usuario al servidor*/
         addActivity()
         /*Lanza el intent para volver al menú inicial*/
         val intent = Intent(
                 this,
                 Activity_Menu :: class.java)
                startActivity(intent)
         /*Elimina la activity_Carrera_Unca de la pila de activitys*/
         finish()
        })
    }
    //TODO FUNCIONES

    /*Esta función no recibe parametros pero devuelve un Int, que será el
resultado del test.
    * se declara un array de enteros en el cuál se cargan las variables de
cada área*/
    private fun resultadoTest(): Int {
        /*Ingreso todos los valores finales de cada área a un array*/
        var resArray : Array <Int> =
                   arrayOf(arte_crea_area0,
                           sociales_area1,
                           economia_area2,
                        tecnologia_area3,
                     eco_bio_salud_area4)
        /*Saco cual obtuvo mayor respuestas positivas*/
        val resFinal = resArray.max()
        /*Obtengo el índice de ese elemento*/
        val i = resArray.indexOf(element = resFinal)
        /*Devuelve el numero de área resultante*/
        return i
    }

        /*Función para agregar los datos del usuario*/
    private fun addActivity() {
        /*OBJETO INTENT*/
        val bundle: Intent = intent

        var nombre3 = bundle.getStringExtra("nombre")
        var edad3 = bundle.getStringExtra("edad")
        var email3 = bundle.getStringExtra("email")
        var sexo3 = bundle.getStringExtra("sexo")
        var origen3 = bundle.getStringExtra("origen")
        var escuela3 = bundle.getStringExtra("escuela")
        var currentTime = Calendar.getInstance().time
        /* ASIGNO LOS VALORES A ENVIAR
        deben ser todos String por que en el Hash Map clave y valor lo
son.*/
```

```
        val getNombre = nombre3
        val getEdad = edad3
        val getEmail = email3
        val getSexo = sexo3
        val getOrigen = origen3
        val getEscuela = escuela3
        val getArea = resultadoTest().toString()
        val getFecha = currentTime.toString()

        /*Indica el método utilizado, en este caso POST y la dirección URL
del servidor */
        val stringRequest = object : StringRequest(Method.POST,addUrl,
Response.Listener<String>{
            response ->
            try {
                val obj = JSONObject(response)
                Toast.makeText(applicationContext,
obj.getString("message"), Toast.LENGTH_SHORT).show()
            }catch (e: JSONException){
                e.printStackTrace()
            }

        }, object : Response.ErrorListener{
            override fun onErrorResponse(volleyError: VolleyError) {
                Toast.makeText(applicationContext, volleyError.message,
Toast.LENGTH_LONG).show()
            }
        }){
            @Throws(AuthFailureError::class)
            override fun getParams(): Map<String, String> {
                val params = HashMap<String, String>()
                //TODO "CLAVE", VALOR
                params.put("nombre", getNombre)
                params.put("edad", getEdad)
                params.put("email", getEmail)
                params.put("sexo", getSexo)
                params.put("origen", getOrigen)
                params.put("escuela", getEscuela)
                params.put("fecha", getFecha)
                params.put("area", getArea)
                return params
            }
        }
        VolleySingleton.instance?.addToRequestQueue(stringRequest)
    }
}
```

Como se puede observar en esta activity las acciones que realiza es la de recibir en primera instancia los datos del usuario que posteriormente enviará hacia el servidor, acompañados del resultado del test.

4.2.2. Codificar ayudas

Siguiendo la premisa de un diseño intuitivo y sencillo, se considera que no es necesaria la codificación de ayudas para la navegación del usuario en la aplicación. Pero sí es necesaria la codificación de las instrucciones de cómo proceder en la resolución del test y qué recaudos tomar durante el proceso de orientación vocacional. Es importante recalcar que el

test sirve como una herramienta de ayuda a la orientación vocacional y que debe ser acompañado por un profesional.

Ilustración 31. Pantalla de instrucciones para realizar el test de orientación vocacional.

4.3. Pruebas de funcionamiento

En esta sección se mencionarán las pruebas realizadas sobre la aplicación móvil, los cambios realizados en la misma de acuerdo a las peticiones por parte de la Dirección de Orientación Vocacional, como así también el análisis de las 6M´s realizado por profesionales y estudiantes avanzados de la carrera de ingeniería en informática.

Antes de describir las pruebas, se muestran algunas pantallas de la aplicación con las que el usuario va a interactuar.

Ilustración 32 Pantalla Menú Principal.

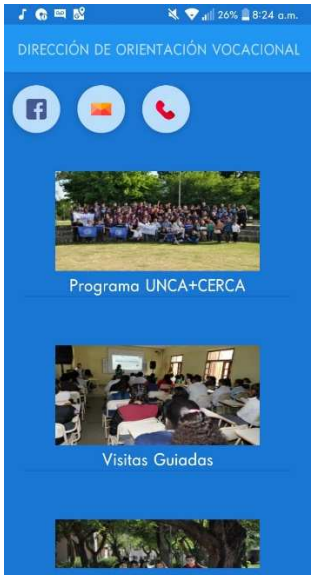


Ilustración 33 Pantalla Dirección de Orientación Vocacional.

Ilustración 34 Pantalla Datos Alumnos.

Ilustración 35 Pantalla Test.

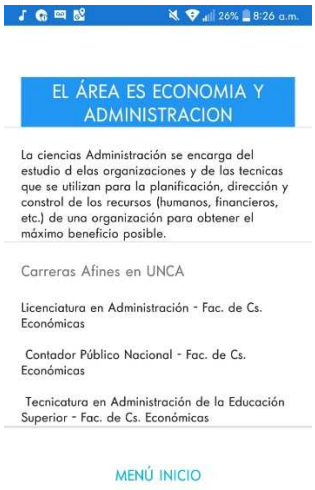


Ilustración 36 Pantalla Resultado Test.

Ilustración 37 Pantalla Bienestar Universitario y Estudiantil.

4.3.1. Dispositivos reales

En esta sección se procede a realizar las pruebas en dispositivos reales. Se tendrá en cuenta las características: versión de Android, tamaño de pantalla, RAM, almacenamiento, medio de transferencia de la aplicación, conexión: Con/sin y 4G/WiFi.

Dispositivo	Android	Pantalla	RAM	ROM	Conexión	Responsive	Tiempo de Respuesta	Funcionamiento mapa	Obtención Resultado TOV	Acceso información Carreras	Acceso Información Sec. Bienestar	Envío de datos usuario
LG Q6ª	7.1.1	5.5''	2GB	16GB	SI	√	√	√	√	√	√	√
LG K10	6.0	5.3''	1.5GB	16GB	SI	√	√	√	√	√	√	√

Tabla 18. Pruebas funcionales en dispositivos reales

4.3.2. Análisis de las 6M´s

Utilizando el método evaluación de potencial de éxito para servicios de tercera generación denominada 6M's, se cuantificaron seis atributos denominados movimiento (movement), momento (moment), yo (me), método (method), dinero (money) y máquinas (machines), los cuales se describen en la Tabla 5. Atributos de la Evaluación de las 6M's Para evaluar la usabilidad se cuantificaron cinco atributos denominados facilidad de aprendizaje, eficiencia, recuerdo en el tiempo, manejo de errores y satisfacción.

A cada uno de los usuarios consultados se entregó un instructivo de como deberá realizar este análisis y una tabla con las características para que pudiera puntuar. El instrumento aplicado en la evaluación incluyó once preguntas cerradas, y valoró el cumplimiento de cada atributo entre uno (Malo) y cinco (Excelente).

La cuantificación se realizó con una muestra de veinte personas con conocimientos en la utilización e implementación de tecnologías: estudiantes avanzados de Ingeniería en

Informática, profesores de computación, técnicos informáticos e ingenieros en informática. Este proceso consistió en la interacción de los veinte usuarios con la aplicación móvil en el escenario real, haciendo uso de los teléfonos celulares LG Q^{α} y LG K^{10}, a través de una red WiFi privada. Cada usuario, a través del instrumento de evaluación (encuesta), se le asignó una puntuación entre uno y cinco a cada atributo, de acuerdo con el nivel de cumplimiento que experimentó cuando interactuó con la aplicación. El modelo de encuesta realizada se incluye en el apéndice 6.2 de este Trabajo Final.

En la Ilustración 38. Gráfico de barras de los resultados del Análisis de las 6M's, se representa el valor promedio para cada atributo, obtenido con el análisis de las encuestas.

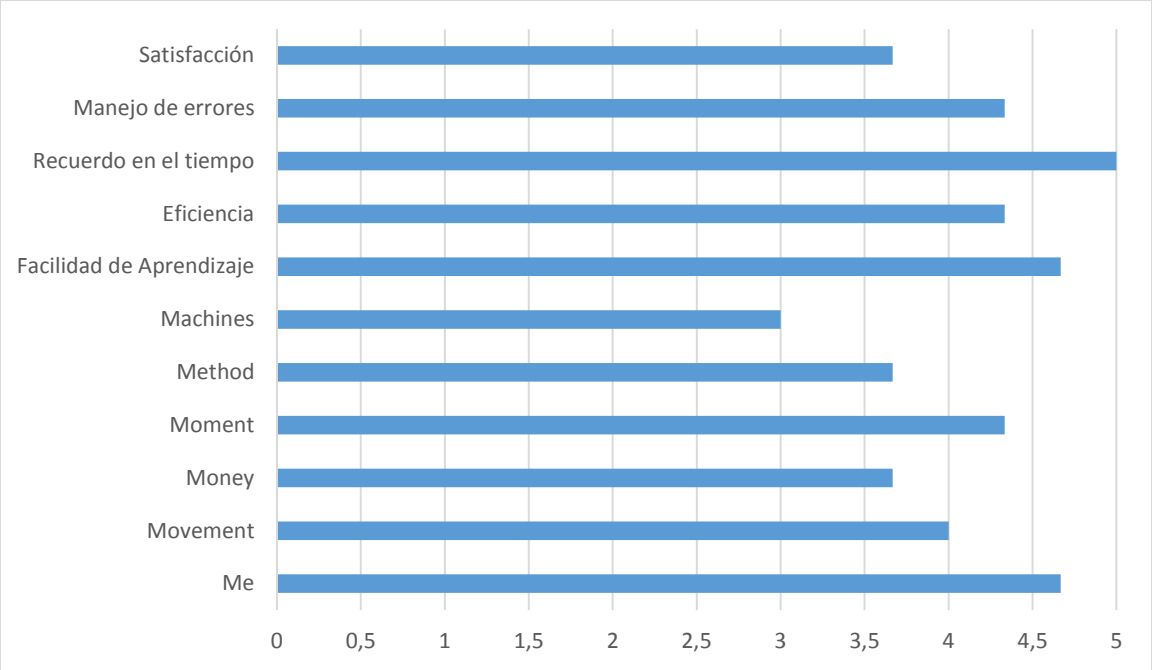


Ilustración 38. Gráfico de barras de los resultados del Análisis de las 6M's

Los resultados estadísticos de evaluar los atributos de las 6M's y la usabilidad sustentan que la aplicación móvil es fácil de manipular, es atractiva al usuario, es eficiente y presenta un manejo adecuado de errores, es decir, que el usuario puede recuperarse rápidamente de los errores que pudiera cometer al utilizarla garantizando la usabilidad de la misma.

El uso de tecnologías de código abierto evitó invertir recursos económicos en el pago de licencias, y requirió de un recurso humano con vastos conocimientos en el área de programación. El costo será mínimo en el caso de utilización de datos móviles debido al envío de los datos del alumno al servidor.

El puntaje más bajo fue obtenido en el atributo Machines dado que el equipamiento utilizado para el desarrollo de la aplicación móvil no es el óptimo, ya que por ejemplo el microprocesador con el que se trabaja está por debajo de los requerimientos básicos para el uso de Android Studio y otra cuestión que no pudo realizarse por esta misma causa es la virtualización para el uso del emulador de Android Studio.

4.4. Entrega de la aplicación móvil

La aplicación móvil será entregada de acuerdo al Reglamento General de Trabajo Final para carreras de grado de la Facultad de Tecnología y Ciencias Aplicadas – Ordenanza N° 008-2015. Se presenta copia impresa y copia digital del informe como así también archivo APK de la aplicación móvil y su código fuente desarrollado en el lenguaje de programación Kotlin y en el lenguaje de etiquetas XML. Se entrega también el manual de usuario y se pone a disposición de la Dirección de Orientación Vocacional para su utilización.

4.4.1. Manual de usuario

El manual de usuario estará destinado para el personal de la Dirección de Orientación Vocacional, el mismo será entregado en formato impreso y digital en formato PDF. En este se describirán las funciones de la aplicación móvil y consideraciones a tener en cuenta para su utilización: requisitos para instalación, formato de los datos ingresados, etc.

El manual de usuario se encuentra en el Apéndice 6.3 Manual de usuario de Test de Orientación Vocacional del presente trabajo final.

4.4.2. Distribución

La aplicación móvil se pondrá a disposición de la Dirección de Orientación Vocacional de la UNCA, la cual podrá ser utilizada en el proceso de orientación vocacional llevado a cabo en dicha dirección.

En caso de requerir un mayor alcance, se recomienda su publicación en la tienda oficial de Android Play Store, como se realizó con la aplicación móvil del SIU Guaraní. También se podrá publicar un link de descarga desde la página web oficial de la UNCA como se hizo con la aplicación móvil de la Secretaria de Ciencia y Tecnología.

Capítulo 5:

Conclusiones

5. Conclusiones

Atendiendo al objetivo general que enuncia: “Desarrollar una aplicación móvil que permita obtener información relevante sobre los intereses y habilidades de los estudiantes del Nivel Secundario, mediante un test de orientación vocacional, y que sirva de herramienta para ayudarlos al momento de elegir una carrera.”

Esto implicó la recuperación de estudios o investigaciones previas que se relacionan respecto a lo temático y metodológico vinculados a la orientación vocacional. En este contexto, cabe destacar que en la Dirección Técnica de Programa de Orientación dependiente de la Universidad de Buenos Aires, se le asigna un papel protagónico a las TICs a la hora de pensar en innovaciones dentro del campo de la orientación vocacional, tanto para realizar la formación de formadores como para la aplicación directa sobre los alumnos orientados. Es por esto que se determina como fundamental la adaptación de los métodos y técnicas de la orientación vocacional de la UNCA a los tiempos actuales donde la tecnología es gran parte del día a día.

Otra de las actividades implicadas fue el estudio de los distintos lenguajes y nuevas tecnologías para el desarrollo de aplicaciones móviles. Si bien existen muchos lenguajes para el desarrollo de éstas, Kotlin resultó ser un lenguaje por sobre todo conciso, cuya curva de aprendizaje resultó ser bastante más ligera que de otro lenguaje de programación orientado a objeto, contando con la ventaja que durante el cursado de la carrera se obtuvo una base en el lenguaje Java. En cuando al IDE utilizado para el desarrollo, Android Studio resultó ser bastante amigable para el aprendizaje de un nuevo lenguaje debido a su editor de código inteligente, permitió que el trabajo sea más rápido y productivo ya que este proporciona o sugiere la terminación de código no sólo para Kotlin, que es el utilizado en esta oportunidad, sino también para otros lenguajes como Java y C / C ++. En la versión utilizada de Android Studio, resalta la facilidad con que se puede convertir código desde Java hacia Kotlin. Sólo es necesario pegar el código en Java y el IDE automáticamente lo detecta y pregunta al desarrollador si desea convertirlo a Kotlin. La compatibilidad Kotlin/Java también ayudó a la hora de utilizar librerías, ya que las desarrolladas para Java también pueden ser utilizadas en Kotlin. En este proyecto por ejemplo se utilizó la librería Volley para el envío de solicitudes HTTP.

La funcionalidad principal de esta aplicación móvil es el test de orientación vocacional. Teniendo en cuenta al tipo de usuario que está dirigida, se optó por realizar un diseño más ligero y que no sea extenuante. Por ejemplo, en algunos test visitados de manera *online*, el usuario debía responder una pregunta/actividad por pantalla. Lo cual puede tornarse sumamente aburrido pensando en que las actividades a analizar en el test elegido son 80 (ochenta). Se optó por enlistar las actividades con su *checkBox* correspondiente en una sola pantalla. En cuanto al número de preguntas/actividades al principio se consultó sobre minimizarlo; pero desde la Dirección de Orientación Vocacional se recomendó conservar todas las preguntas/actividades aduciendo que los estudiantes que se acercaban respondían tests de hasta 150 preguntas. Por lo cual, siguiendo la recomendación de las experimentadas en la materia, se conservó la cantidad.

Por lo expresado, y considerando la investigación citada en el capítulo 2.2.4, el aporte de una herramienta tecnológica como lo es una aplicación móvil a la Dirección de Orientación Vocacional, servirá como bisagra para la utilización de nuevas herramientas TICs en el

proceso de orientación vocacional y en las demás actividades en las que se considere necesario, ya que actualmente los tests se realizan de forma manual. Por otro lado, el alumno orientado al tener la aplicación en su smartphone contará con información sobre la oferta académica de cada Facultad y de la Escuela de Arqueología, como así también sobre los distintos servicios que ofrece la Secretaría de Bienestar Universitario y Asuntos Estudiantiles a través de sus distintas dependencias: Dirección de Salud Universitaria, Dirección de Deporte, etc.

El resultado de este Trabajo Final da inicio a una serie de trabajos que pueden realizarse en distintas etapas y/o contextos. Considerando que en esta primera entrega se trabajó en conjunto con la Dirección de Orientación Vocacional perteneciente a la Secretaría de Bienestar Universitario y Asuntos Estudiantiles; la información que se brinda sobre esta y la Secretaria a la cual pertenece es más precisa en comparación con la información sobre las Unidades Académicas que conforman la UNCA. Incluso se brindan medios de comunicación directos, email, Facebook y teléfono fijo, desde la aplicación hacia la Dirección de Orientación Vocacional y hacia la Secretaría de Bienestar Universitario y Asuntos Estudiantiles. Entonces, se pueden plantear etapas a futuro que consideren un trabajo en conjunto con las Facultades y Escuela de Arqueología, incluyendo los anexos de la UNCA en los departamentos Belén y Santa Rosa, donde se puedan ofrecer desde la aplicación canales de comunicación directos para los usuarios, y que la respuesta por parte de las Unidades Académicas esté asegurada. Esto también permitiría mantener actualizada la información brindada y un mayor alcance de usuarios.

Por último, cabe destacar, que esta aplicación también puede ser replicada a nivel provincial, nucleando la información de los niveles terciarios y universitarios tanto públicos como privados en una única aplicación móvil y también presente en la Web.

Apéndices

6. Apéndice

En el presente apéndice se describirán aquellas herramientas que fueron utilizadas para el desarrollo de la aplicación móvil: SDK, librerías, software, etc.

6.1. Herramientas utilizadas

6.1.1. SDK MapBox

Actualmente a la hora de trabajar con mapas en aplicaciones móviles la opción más utilizada es la API de Google Maps, siendo también la más conocida por los usuarios de *smartphones*. Pero también existen otras variantes como ser el SDK (Kit de desarrollo de software) de MapBox y el framework MapKit de Apple. En este trabajo, para desarrollar la funcionalidad que permite mostrar las ubicaciones de las unidades académicas, secretarías, rectorado y demás dependencias de la UNCA, se utilizó el SDK de Map Box, debido a su simplicidad y flexibilidad, además, de otras características que lo hacen atractivo, las cuales son:

Open source. Todo el código es abierto y basado en estándares abiertos. En la actualidad existen más de 500 repositorios en Github. Esto permite ver funcionalidades que están en desarrollo, consultar sobre algún problema que se pueda observar y también realizar contribuciones. Se la considera una plataforma hecha por desarrolladores para desarrolladores.

Multiplataforma. Existen SDKs disponibles para prácticamente todas las plataformas: Android, iOS, Web, Qt, Unity y MacOS. Todas las funcionalidades son accesibles desde cada una de ellas dado que comparten una misma base.

In-app. No hay necesidad de acceder a otra aplicación para realizar alguna acción; todas se llevan a cabo en la mima app. Esto es sumamente positivo ya que permite brindar una experiencia de navegación completa sin que el usuario deba salir de la app desarrollada, lo cual permite no “perder” a los usuarios.

Para utilizar los servicios de Mapbox en una aplicación se debe obtener el *token* de acceso, al cual puede accederse tras registrarse en la página oficial, donde se puede acceder a guías rápidas de instalación para Android e IOs. Al completar la guía rápida ya estará todo listo para comenzar a programar utilizando esta herramienta [33].

A continuación, se muestra el código utilizado para incluir un mapa en una activity, puntos de referencia y para situar la cámara en un punto específico. Este código pertenece a la activity: Activity_Mapa.kt:

```
package com.example.andre.appov

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import com.mapbox.mapboxsdk.Mapbox
import com.mapbox.mapboxsdk.annotations.MarkerOptions
import com.mapbox.mapboxsdk.camera.CameraPosition
```

```
import com.mapbox.mapboxsdk.camera.CameraUpdateFactory
import com.mapbox.mapboxsdk.geometry.LatLng
import com.mapbox.mapboxsdk.maps.MapView
import kotlinx.android.synthetic.main.activity_main8.*

class Activity_Mapa : AppCompatActivity() {
    //Se declara el mapView
    private lateinit var mapView: MapView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        /*Token de acceso es configurado aquí, se encuentra guardado como
un              recurso string*/
        Mapbox.getInstance(applicationContext,
getString(R.string.access_token))

        /*Esto contiene el MapView en XML y debe llamarse después de
configurar el token de acceso.*/
        setContentView(R.layout.activity_main8)

        /*Inicializa la vista del mapa*/
        mapView = findViewById(R.id.mapView)
        mapView.onCreate(savedInstanceState)

        //TODO TOOLBAR
        /* Establecer la barra de herramientas como barra de acción de
apoyo*/
        setSupportActionBar(toolbar)
        // Soporte del actionBar
        val actionBar = supportActionBar
        // Titulo actionBar
        actionBar!!.title = "Conocé la UNCa"
        // Elevación actionBar
        actionBar.elevation = 4.0F

        //TODO SE AGREGAN LOS MARCADORES AL MAPVIEW. LONG= X ^ LAT=Y.
        /*BIENESTAR*/
        mapView.getMapAsync { mapboxMap ->
            mapboxMap.addMarker(MarkerOptions()
                    .position(LatLng(-28.4594186, -65.7828000))
                    .title("Secretaria de Bienestar Universitario")
                    .snippet("UNCa")
            )
        }
        /*EXACTAS*/
        mapView.getMapAsync { mapboxMap ->
            mapboxMap.addMarker(MarkerOptions()
                    .position(LatLng(-28.4594000, -65.7828000))
                    .title("Facultad de Exactas")
                    .snippet("UNCa")
            )
        }
        /*HUMANIDADES*/
        mapView.getMapAsync { mapboxMap ->
            mapboxMap.addMarker(MarkerOptions()
                    .position(LatLng(-28.4602600, -65.7830000))
                    .title("Facultad de Humanidades")
                    .snippet("UNCa")
            )
        }
```

```
/*TECNOLOGIA*/
mapView.getMapAsync { mapboxMap ->
    mapboxMap.addMarker(MarkerOptions()
            .position(LatLng(-28.4602600, -65.7821000))
            .title("Facultad de Tecnología y Cs. Aplicadas")
            .snippet("UNCa")
    )
}
/*SALUD*/
mapView.getMapAsync { mapboxMap ->
    mapboxMap.addMarker(MarkerOptions()
            .position(LatLng(-28.4594078, -65.7839000))
            .title("Facultad de Salud")
            .snippet("UNCa")
    )
}
/*DERECHO*/
mapView.getMapAsync { mapboxMap ->
    mapboxMap.addMarker(MarkerOptions()
            .position(LatLng(-28.4593000, -65.7828000))
            .title("Facultad de Derecho")
            .snippet("UNCa")
    )
}
/*ARQUEOLOGIA*/
mapView.getMapAsync { mapboxMap ->
    mapboxMap.addMarker(MarkerOptions()
            .position(LatLng(-28.4592000, -65.7829900)
            .title("Escuela de Arqueología")
            .snippet("UNCa")
    )
}
/*AGRARIAS*/
mapView.getMapAsync { mapboxMap ->
    mapboxMap.addMarker(MarkerOptions()
            .position(LatLng(-28.460151, -65.784200))
            .title("Facultad de Cs. Agrarias")
            .snippet("1er Piso")
    )
}
/*ECONOMICAS*/
mapView.getMapAsync { mapboxMap ->
    mapboxMap.addMarker(MarkerOptions()
            .position(LatLng(-28.459951, -65.784200))
            .title("Facultad de Cs. Económicas")
            .snippet("2do Piso")
    )
}
/*COMEDOR UNIVERSITARIO*/
mapView.getMapAsync { mapboxMap ->
    mapboxMap.addMarker(MarkerOptions()
            .position(LatLng(-28.4602800, -65.780926))
            .title("Comedor Universitario")
            .snippet("UNCa")
    )
}
/*RESIDENCIA UNIVERSITARIA*/
mapView.getMapAsync { mapboxMap ->
    mapboxMap.addMarker(MarkerOptions()
            .position(LatLng( -28.466000, -65.782120))
```

```
                .title("Residencia Universitaria")
                .snippet("UNCa")
        )
    }
    /*SALUD UNIVERSITARIA*/
    mapView.getMapAsync { mapboxMap ->
        mapboxMap.addMarker(MarkerOptions()
                .position(LatLng(-28.459999, -65.782941))
                .title("Salud Universitaria")
                .snippet("Patio Este")
        )
    }
    /*DIRECCIÓN DE ORIENTACIÓN VOCACIONAL*/
    mapView.getMapAsync { mapboxMap ->
        mapboxMap.addMarker(MarkerOptions()
                .position(LatLng( -28.466000, -65.782220))
                .title("Dirección de Orientación Vocacional")
                .snippet("UNCa")
        )
    }
    /*RECTORADO*/
    mapView.getMapAsync { mapboxMap ->
        mapboxMap.addMarker(MarkerOptions()
                .position(LatLng(-28.465646, -65.775816))
                .title("Rectorado")
                .snippet("UNCa")
        )
    }
    /*EXTENSIÓN UNIVERSITARIA*/
    mapView.getMapAsync { mapboxMap ->
        mapboxMap.addMarker(MarkerOptions()
                .position(LatLng(-28.466200, -65.785126))
                .title("Dirección de Extensión Universitaria")
                .snippet("UNCa")
        )
    }
}
//TODO FUNCIONES MAPBOX
public override fun onStart() {
    super.onStart()
    mapView.onStart()
}
public override fun onResume() {
    super.onResume()
    mapView.onResume()
}
public override fun onPause() {
    super.onPause()
    mapView.onPause()
}
public override fun onStop() {
    super.onStop()
    mapView.onStop()
}
override fun onLowMemory() {
    super.onLowMemory()
    mapView.onLowMemory()
}
override fun onDestroy() {
    super.onDestroy()
```

```
        mapView.onDestroy()
    }
    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        if (outState != null){
            mapView.onSaveInstanceState(outState)
        }

    }

    //TODO FUNCIONES TOOLBAR
    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        // Inflate para el menú que se utiliza en el actionBar
        val inflater = menuInflater
        inflater.inflate(R.menu.menu, menu)
        return super.onCreateOptionsMenu(menu)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        // Según sea la opción elegida será la posición que tomará la
cámara.
        when (item.itemId) {
        //TODO UBICA EL MAPA SOBRE FACULTAD DE TECNOLOGIA
            R.id.action_tecno -> {
                mapView.getMapAsync { mapboxMap ->

                    val position = CameraPosition.Builder()
                            //Ubica la cámara en la nueva posición
                            .target(LatLng(-28.4602600, -65.7821000))
                            //Proporciona el zoom
                            .zoom(17.0)
                            //Rota la cámara
                            .bearing(180.0)
                            //Establece la inclinación de la cámara
                            .tilt(30.0)
                            //Crea una CamaraPosition para el build
                            .build()
                    mapboxMap.animateCamera(CameraUpdateFactory
                            .newCameraPosition(position), 7000)
                }
                return true
            }
        //TODO UBICA EL MAPA SOBRE ARQUEOLOGIA
            R.id.action_arque -> {
                //TODO UBICA EL MAPA SOBRE ESCUELA DE ARQUEOLOGIA
                mapView.getMapAsync { mapboxMap ->

                    val position = CameraPosition.Builder()
                            .target(LatLng(-28.4592000, -65.7829900))
                            .zoom(17.0)
                            .bearing(180.0)
                            .tilt(30.0)
                            .build()
                    mapboxMap.animateCamera(CameraUpdateFactory
                            .newCameraPosition(position), 7000)
                }
                return true
            }
        //TODO UBICA EL MAPA SOBRE SECRETARIA DE BIENESTAR
            R.id.action_bienestar -> {
```

```
            mapView.getMapAsync { mapboxMap ->
                val position = CameraPosition.Builder()
                        .target(LatLng(-28.4594186, -65.7828000))
                        .zoom(17.0)
                        .bearing(180.0)
                        .tilt(30.0)
                        .build()
                mapboxMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(position), 7000)
            }
            return true
        }
    //TODO UBICA EL MAPA SOBRE COMEDOR UNIVERSITARIO
        R.id.action_comedor -> {
            mapView.getMapAsync { mapboxMap ->
                val position = CameraPosition.Builder()
                        .target(LatLng(-28.4602800, -65.780926))
                        .zoom(17.0)
                        .bearing(180.0)
                        .tilt(30.0)
                        .build()
                mapboxMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(position), 7000)
            }
            return true
        }
    //TODO UBICA EL MAPA SOBRE FACULTAD DE DERECHO
        R.id.action_derecho -> {
            mapView.getMapAsync { mapboxMap ->
                val position = CameraPosition.Builder()
                        .target(LatLng(-28.4593000, -65.7828000))
                        .zoom(17.0)
                        .bearing(180.0)
                        .tilt(30.0)
                        .build()
                mapboxMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(position), 7000)
            }
            return true
        }
    //TODO UBICA EL MAPA SOBRE DOV
        R.id.action_dov -> {
            mapView.getMapAsync { mapboxMap ->
                val position = CameraPosition.Builder()
                        .target(LatLng( -28.466000, -65.782220))
                        .zoom(17.0)
                        .bearing(180.0)
                        .tilt(30.0)
                        .build()
                mapboxMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(position), 7000)
            }
            return true
        }
    //TODO UBICA EL MAPA SOBRE FACULTAD DE ECONOMIA
        R.id.action_eco -> {
            mapView.getMapAsync { mapboxMap ->
                val position = CameraPosition.Builder()
                        .target(LatLng(-28.459951, -65.784200))
                        .zoom(17.0)
```

```
                .bearing(180.0)
                .tilt(30.0)
                .build()

            mapboxMap.animateCamera(CameraUpdateFactory
                .newCameraPosition(position), 7000)
        }
        return true
    }
//TODO UBICA EL MAPA SOBRE FACULTAD DE EXACTAS
    R.id.action_exactas -> {
        mapView.getMapAsync { mapboxMap ->

            val position = CameraPosition.Builder()
                .target(LatLng(-28.4594000, -65.7828000))
                .zoom(17.0)
                .bearing(180.0)
                .tilt(30.0)
                .build()
            mapboxMap.animateCamera(CameraUpdateFactory
                .newCameraPosition(position), 7000)
        }
        return true
    }
//TODO UBICA EL MAPA SOBRE EXTENSION UNIVERSITARIA
    R.id.action_extension -> {
        mapView.getMapAsync { mapboxMap ->
            val position = CameraPosition.Builder()
                .target(LatLng(-28.466200, -65.785126))
                .zoom(17.0)
                .bearing(180.0)
                .tilt(30.0)
                .build()
            mapboxMap.animateCamera(CameraUpdateFactory
                .newCameraPosition(position), 7000)
        }
        return true
    }
//TODO UBICA EL MAPA SOBRE FACULTAD DE HUMANIDADES
    R.id.action_huma -> {
        mapView.getMapAsync { mapboxMap ->
            val position = CameraPosition.Builder()
                .target(LatLng(-28.4602600, -65.7830000))
                .zoom(17.0)
                .bearing(180.0)
                .tilt(30.0)
                .build()
            mapboxMap.animateCamera(CameraUpdateFactory
                .newCameraPosition(position), 7000)
        }
        return true
    }
//TODO UBICA EL MAPA SOBRE RECTORADO
    R.id.action_rectorado -> {
        mapView.getMapAsync { mapboxMap ->
            val position = CameraPosition.Builder()
                .target(LatLng(-28.465646, -65.775816))
                .zoom(17.0)
                .bearing(180.0)
                .tilt(30.0)
```

```
                        .build()
                    mapboxMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(position), 7000)
                }
                return true
            }
        //TODO UBICA EL MAPA SOBRE FACULTAD DE SALUD
            R.id.action_salud -> {
                mapView.getMapAsync { mapboxMap ->
                    val position = CameraPosition.Builder()
                        .target(LatLng(-28.4594078, -65.7839000))
                        .zoom(17.0)
                        .bearing(180.0)
                        .tilt(30.0)
                        .build()
                    mapboxMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(position), 7000)
                }
                return true
            }
        //TODO UBICA EL MAPA SOBRE SALUD UNIVERSITARIA
            R.id.action_salud_univ -> {
                mapView.getMapAsync { mapboxMap ->
                    val position = CameraPosition.Builder()
                        .target(LatLng(-28.459999, -65.782941))
                        .zoom(17.0)
                        .bearing(180.0)
                        .tilt(30.0)
                        .build()
                    mapboxMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(position), 7000)
                }
                return true
            }
        }
        return super.onOptionsItemSelected(item)
    }
}
```

En el siguiente fragmento de código, en la parte superior, se encuentran las líneas que agregan marcadores de las distintas ubicaciones al mapa. En este fragmento se muestra cómo se agrega el marcador de la Facultad de Tecnología y Ciencias Aplicadas.:

```
        /*TECNOLOGIA*/
        mapView.getMapAsync { mapboxMap ->
            mapboxMap.addMarker(MarkerOptions()
                    .position(LatLng(-28.4602600, -65.7821000))
                    .title("Facultad de Tecnología y Cs. Aplicadas")
                    .snippet("UNCa")
            )
        }
```

Desde el *toolBar* se despliega un menú donde se encuentran las distintas ubicaciones que se pueden visitar con la cámara de mapBox. Solo con presionar una de las opciones la cámara se dirigirá automáticamente a la elección. En el siguiente fragmento de código se muestra cómo ubicar la cámara sobre la Facultad de Tecnología y Ciencias Aplicadas.:

```
            R.id.action_tecno -> {
                mapView.getMapAsync { mapboxMap ->
```

```
                val position = CameraPosition.Builder()
                        //Ubica la cámara en la nueva posición
                        .target(LatLng(-28.4602600, -65.7821000))
                        //Proporciona el zoom
                        .zoom(17.0)
                        //Rota la cámara
                        .bearing(180.0)
                        //Establece la inclinación de la cámara
                        .tilt(30.0)
                        //Crea una CamaraPosition para el build
                        .build()
                mapboxMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(position), 7000)
            }
            return true
        }
```

En cuanto al diseño en XML, se maqueta de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:mapbox="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    >

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:minHeight="?attr/actionBarSize"
        android:padding="1dp"
        />

    <com.mapbox.mapboxsdk.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        mapbox:mapbox_cameraTargetLat="-28.459005"
        mapbox:mapbox_cameraTargetLng="-65.783215"
        mapbox:mapbox_cameraZoom="15.00"
        />
</LinearLayout>
```

Se utiliza un linearLayout con orientación vertical para enmarcar los elementos que conforman esta pantalla. Se utiliza un toolBar para contener el menú desplegable donde estarán las opciones y un mapa al cual se le indica su ubicación inicial con los atributos mapbox:mapbox_cameraTargetLat="-28.459005 y mapbox:mapbox_cameraTargetLng="-65.783215", como así también el zoom inicial de la cámara con el atributo mapbox:mapbox_cameraZoom=”15.00” [33].

6.1.2. Library Volley

Volley [42] es una librería desarrollada por Google para facilitar la comunicación de red en las aplicaciones Android, debido a la ausencia en el SDK de Android de una clase de red capaz de funcionar sin interferir con la experiencia del usuario.

Está totalmente enfocado en las peticiones, evitando la creación de código repetitivo para manejar tareas asíncronas por cada petición o incluso para parsear los datos que vienen del flujo externo.

Al ser una librería desarrollada y mantenida por Google se podría considerar como la forma "estándar" de trabajar con este tipo de peticiones y esto da la confianza de que la librería va a tener un soporte y un mantenimiento garantizado.

Hasta el lanzamiento de Volley, la clase canónica Java java.net.HttpURLConnection y Apache org.apache.http.client eran las únicas herramientas disponibles para los programadores de Android para desarrollar un sistema RESTful[20] entre un cliente y un backend remoto.

En niveles inferiores de API en Android, HttpUrlConnection y HttpClient cuentan con algunos problemas conocidos y errores que nunca fueron corregidos. Además, HttpClient quedó en desuso en la última actualización de la API (API 22), lo que significa que ya no se mantendrá y puede eliminarse en una versión futura. Considerando que la aplicación móvil está desarrollada para las API 23 en adelante no se considera oportuno la utilización de HttpUrlConnection y HttpClient.

Volley ofrece los siguientes beneficios:

- Programación automática de solicitudes de red.
- Múltiples conexiones de red concurrentes.
- Soporte para la priorización de solicitudes.
- API de solicitud de cancelación. Puede cancelar una sola solicitud o puede establecer bloques o ámbitos de solicitudes para cancelar.
- Facilidad de personalización, por ejemplo, para reintento y retroceso.
- Herramientas de depuración y rastreo.

Se integra fácilmente con cualquier protocolo y sale de la caja con soporte para cadenas en bruto, imágenes y JSON. Al proporcionar soporte integrado para las funciones que necesita, evita la necesidad de escribir código repetitivo y permite concentrarse en la lógica específica de la aplicación.

No es recomendable para grandes operaciones de descarga o transmisión, ya que guarda todas las respuestas en la memoria durante el análisis.

La biblioteca central de Volley está desarrollada en GitHub y contiene el canal principal de despacho de solicitudes, así como un conjunto de utilidades comunes.

Para agregar esta librería al proyecto se debe ingresar las siguientes líneas de código en el archivo build.gradle de su aplicación:

```
dependencies {
    ...
```

[20] Sistema que implementa la arquitectura REST cuya sigla deriva de "Representational State Transfer" que traducido vendría a ser "transferencia de representación de estado".

```
    //volley
    implementation 'com.android.volley:volley:1.0.0'
}
```

Otra cuestión a tener en cuenta es agregar en el archivo AndroidManifiest el permiso para que la aplicación pueda conectarse a la red:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.andre.appov">

    <uses-permission android:name="android.permission.INTERNET" />
…
</manifest>
```

Posee varios componentes que optimizan la administración de las peticiones generadas desde las aplicaciones Android. La gestión comienza en una Cola de Peticiones que recibe cada una de las peticiones generadas, donde son previamente priorizadas para su realización.
Luego son seleccionadas por un elemento llamado Cache Dispatcher, cuya función es comprobar si la respuesta de la petición actual puede ser obtenida de resultados previos guardados en caché. Si es así, entonces se pasa a parsear la respuesta almacenada y luego se presenta al hilo principal. En caso negativo, se envía la petición a la Cola de Conexiones Pendientes, donde reposan todas aquellas peticiones que están por ejecutarse.
Luego entra en juego un componente llamado Network Dispatcher, el cual se encarga de seleccionar las peticiones pendientes de la cola, para realizar las respectivas transacciones HTTP hacia el servidor. Si es necesario, las respuesta de estas peticiones se guardan en caché, luego se parsean y finalmente se publican en el hilo principal. En la Ilustración se puede observar este circuito.

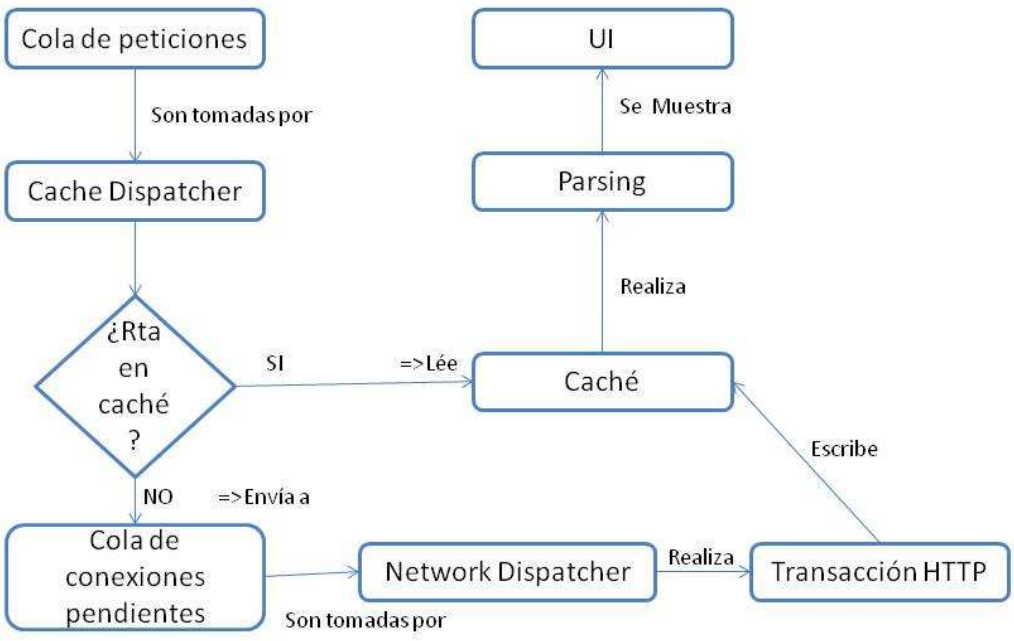


Ilustración 39. Funcionamiento Librería Volley

6.1.3. Herramienta de evaluación de aplicación móvil

Análisis de las 6M´s y de usabilidad

6.2. Documento de evaluación

Para probar la aplicación móvil para Test de Orientación Vocacional se proveerá al usuario de un *smarthphone*, el cual cuenta con las siguientes características:

LG Q6[α]	Android	API	RAM	ROM	Pantalla
	Nougat 7.1.1	24 -25	2GB	16GB	5.5”

Tabla 19. Características de smartphone LG Q6

En cuanto a los recursos utilizados para el desarrollo de la aplicación:

- Notebook marca Sansumg Modelo Np300e5a, Procesador Intel Celeron, 4GB de RAM se adicionó una memoria de 4GB para cumplir con los requerimientos de Android Studio.
- Cable USB a Micro USB, se utilizó para realizar las pruebas en el dispositivo móvil, ya que la notebook no posee la función de virtualización requerida para utilizar el emulador de Android Studio.
- Software Android Studio.
- Dispositivos móviles de gama media.

Las métricas que se utilizarán para evaluar la aplicación móvil están descriptas en la Tabla 21, las primeras 6 (seis) se corresponden con el análisis de las 6 M’s mientras que las restantes son las utilizadas para evaluar la usabilidad en un análisis propuesto por Jakob Nielsen[21].

Instrucciones

El usuario evaluador deberá utilizar la aplicación móvil en todas sus funciones con el fin de emitir su opinión en referencia a las métricas descriptas en la Tabla 21.

Luego de esto deberá puntuar el nivel de cumplimiento del atributo evaluado en un margen de cinco niveles que van desde Malo a Excelente, como se describe en la Tabla 20.

Los resultados de la encuesta serán cuantificados de acuerdo con el nivel de cumplimiento del atributo evaluado que experimentó el usuario al momento de interactuar con la aplicación.

Escala de Evaluación

Tiene 5 (cinco) niveles, los cuales al cuantificarse se corresponden con un orden descendiente.

Malo	○
Regular	○
Bueno	○
Muy Bueno	○

[21] NIELSEN, Jakob – “Usability Engineering” – Editorial Morgan Kaufmann - Copyright © 1993

Excelente	○

Tabla 20. Calificaciones posibles en el Análisis de las 6M's

Tabla evaluativa

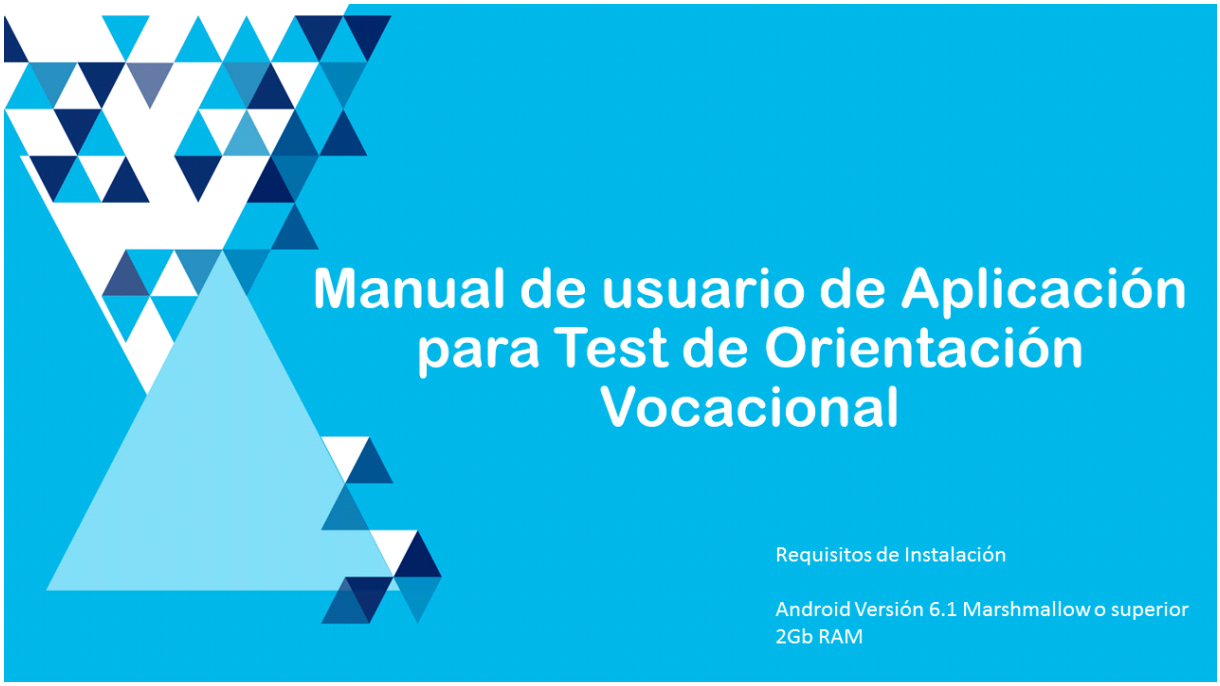
Atributo	Descripción	Puntuación
Me / Yo	Se consideran todos los aspectos asociados al personal, a la mano de obra. Su capacitación, motivación, habilidad en su trabajo, etc.	○ ○ ○ ○ ○
Movement / Movimiento	Analiza la movilidad de la aplicación.	○ ○ ○ ○ ○
Money / Dinero	Analiza si existe algún costo en la utilización, licencias, utilización de la red.	○ ○ ○ ○ ○
Moment / Momento	Rapidez con la que realiza los procesos requeridos por el usuario.	○ ○ ○ ○ ○
Method / Método	Se evalúa la forma en la que se realizan las acciones. Al evaluar los métodos, estamos evaluando la forma en cómo se produce independiente de la mano de obra.	○ ○ ○ ○ ○
Machines / Máquinas	Herramientas con las que se cuenta para dar salida al producto final. Software, hardware, etc. Se evalúa capacidad suficiente para cumplir su función, eficiencia, modernidad, etc.	○ ○ ○ ○ ○
Facilidad de aprendizaje	Cuán rápido el usuario puede empezar a realizar un trabajo productivo la primera vez que utiliza la aplicación.	○ ○ ○ ○ ○
Eficiencia	Productividad del usuario con el uso de la aplicación.	○ ○ ○ ○ ○

Recuerdo en el tiempo	Capacidad del sistema de utilizar siempre la aplicación sin tener que recordar su funcionamiento.	○ ○ ○ ○ ○
Manejo de errores	Errores cometidos durante el uso del sistema y cuan fácil el usuario se recupera de ellos.	○ ○ ○ ○ ○
Satisfacción	Opinión subjetiva del usuario sobre la aplicación	○ ○ ○ ○ ○

Tabla 21. Evaluación de las 6 M's

Muchas gracias por su colaboración en la elaboración de mi trabajo final.

6.3. Manual de usuario de Test de Orientación Vocacional



Menú Inicial

Menú principal

El primer icono nos permite enviar un email a la Sub Secretaria de Asuntos Estudiantiles de la Universidad Nacional de Catamarca. Y los siguientes el acceso a redes sociales : Facebook e Instagram.

Dirección de Orientación Vocacional

Seleccionamos la opción de Dirección de Orientación Vocacional


Direción de Orientación Vocacional
Estos tres botones nos enlazan a:
Facebook
Email
Muestra el número de teléfono de la Dirección
Muestra un listados de las actividades realizadas por la Dirección. Seleccionando una de estas se podrá ver su descripción.
DIRECCIÓN DE ORIENTACIÓN VOCACIONAL
Programa UNCA+CERCA
Visitas Guiadas


Test de
Orientación
Vocacional

Test de Orientación Vocacional

Seleccionamos la opción de Test de Orientación Vocacional

Test de Orientación Vocacional

En esta pantalla el alumno debe ingresar sus datos con los siguientes formatos:

Letras hasta un total de 30 caracteres. Ej.: María Victoria Díaz

Números hasta un total de 2 caracteres. Ej.: 22, 17, etc.

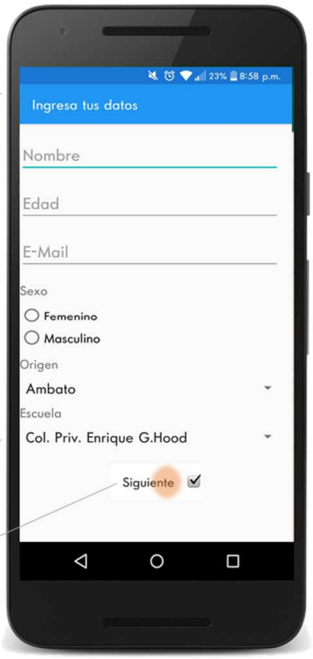
Dirección de correo electrónico. Ej.: nombre@dominio.com

Dirección de correo electrónico. Ej.: nombre@dominio.com

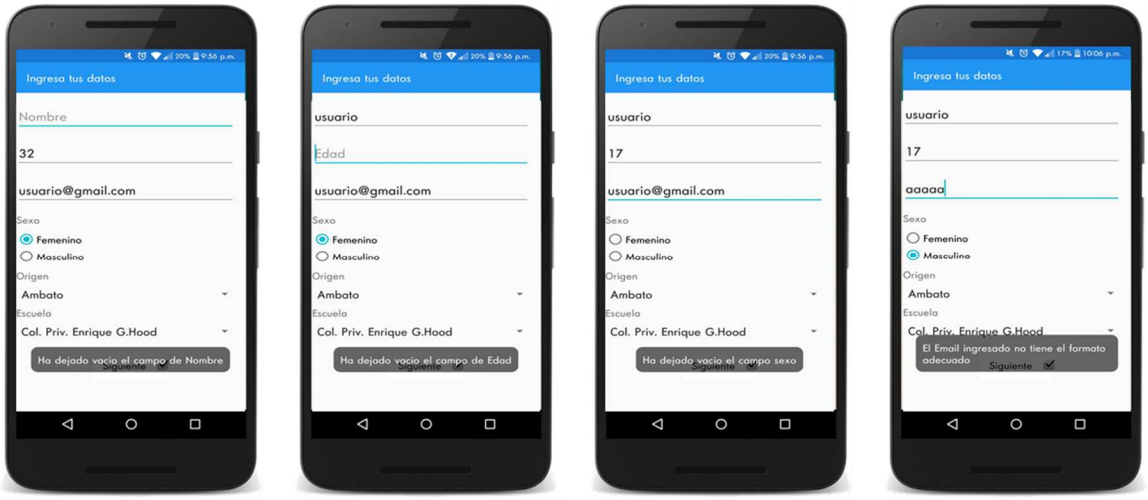
Elegir del listado de departamentos.

Elegir del listado de escuelas y colegios.

Si todos los datos están correctamente ingresados con su formato podrá ingresar al test. De lo contrario deberá revisar los errores que se describen a continuación.



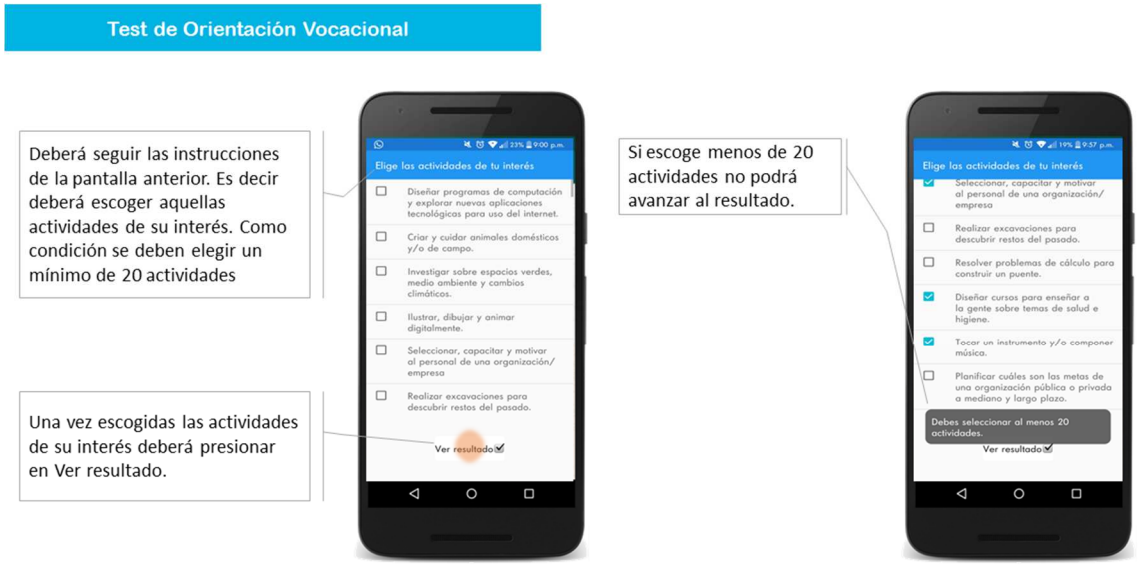
Test de Orientación Vocacional



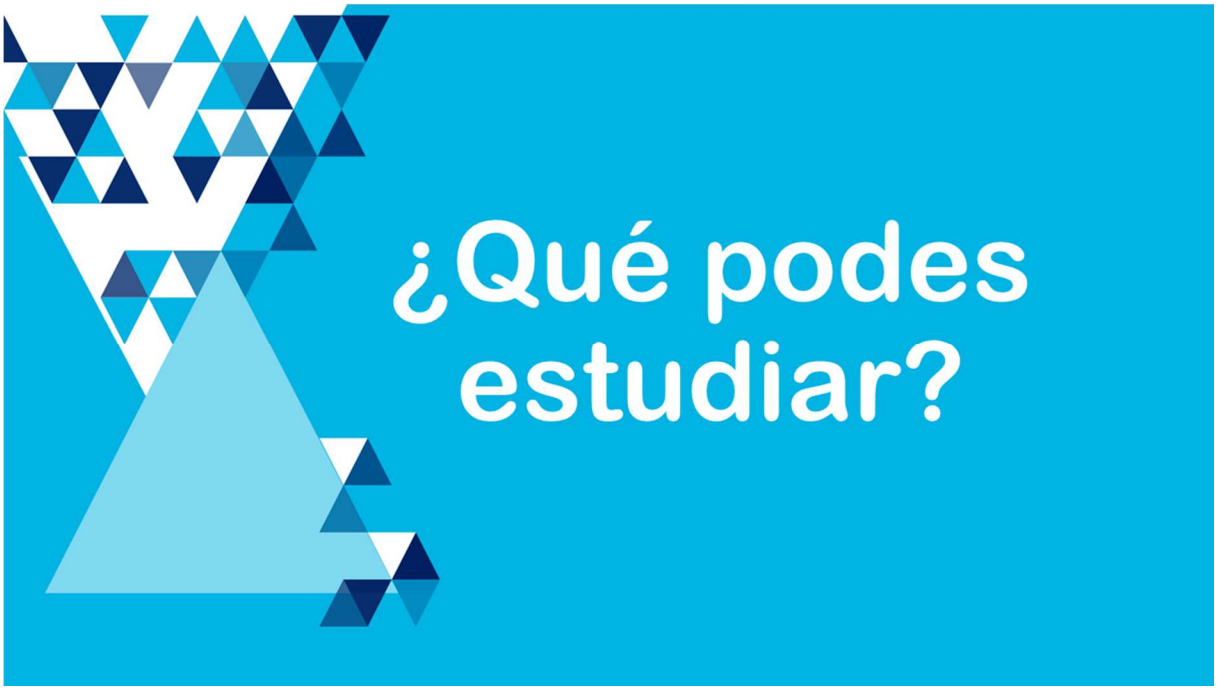
Test de Orientación Vocacional

Aquí podremos observar una serie de recomendaciones para los alumnos que realicen este test y de como mejorar el proceso de orientación vocacional consultando a un profesional del área.

Para acceder al test se debe presionar en el botón Empezar Test.


Test de Orientación Vocacional
Deberá seguir las instrucciones de la pantalla anterior. Es decir deberá escoger aquellas actividades de su interés. Como condición se deben elegir un mínimo de 20 actividades
Una vez escogidas las actividades de su interés deberá presionar en Ver resultado.
Elige las actividades de tu interés
Diseñar programas de computación y explorar nuevas aplicaciones tecnológicas para uso del internet.
Criar y cuidar animales domésticos y/o de campo.
Investigar sobre espacios verdes, medio ambiente y cambios climáticos.
Ilustrar, dibujar y animar digitalmente.
Seleccionar, capacitar y motivar al personal de una organización/ empresa
Realizar excavaciones para descubrir restos del pasado.
Ver resultado
Si escoge menos de 20 actividades no podrá avanzar al resultado.
Elige las actividades de tu interés
Seleccionar, capacitar y motivar al personal de una organización/ empresa
Realizar excavaciones para descubrir restos del pasado.
Resolver problemas de cálculo para construir un puente.
Diseñar cursos para enseñar a la gente sobre temas de salud e higiene.
Tocar un instrumento y/o componer música.
Planificar cuáles son las metas de una organización pública o privada a mediano y largo plazo.
Debes seleccionar al menos 20 actividades.
Ver resultado


Test de Orientación Vocacional
El Resultado mostrará:
Área Resultante.
Carreras afines dentro de la oferta académica de la Universidad Nacional de Catamarca.
Para volver a la pantalla de Inicial se presiona el botón Inicio.
Resultado de Test
ECONOMIA Y ADMINISTRACION
Carreras Afines en UNCA
Licenciatura en Administración
Contador Público Nacional
Tecnicatura en Administración de la Educación Superior
Inicio



¿Qué podes estudiar?

Seleccionamos la opción de ¿Qué podes estudiar?

¿Qué podes estudiar?

Se presenta un listado con las facultades, se debe escoger una para acceder a su información

¿Qué podes estudiar?

Presionando este botón se abrirá la página de la facultad elegida, lo cual permitirá acceder a mas información.

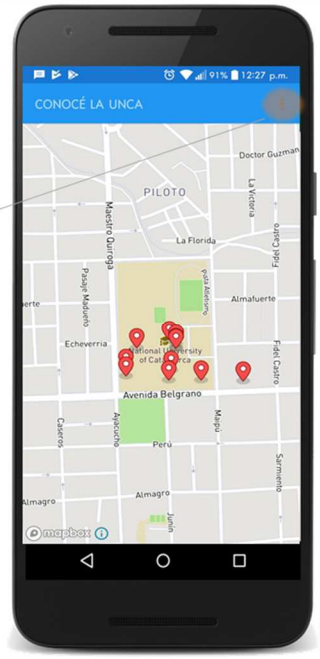
Se muestran todas las carreras de pre grado y grado que ofrece cada facultad.

Conoce la UNCa

Seleccionamos la opción de Test de Orientación Vocacional

Conoce la UNCa

Se presiona en este ícono para ver las localización de las facultades, Rectorado, Dirección de Orientación Vocacional, etc.



Conoce la UNCa

Seleccionamos la opción de Test de Orientación Vocacional

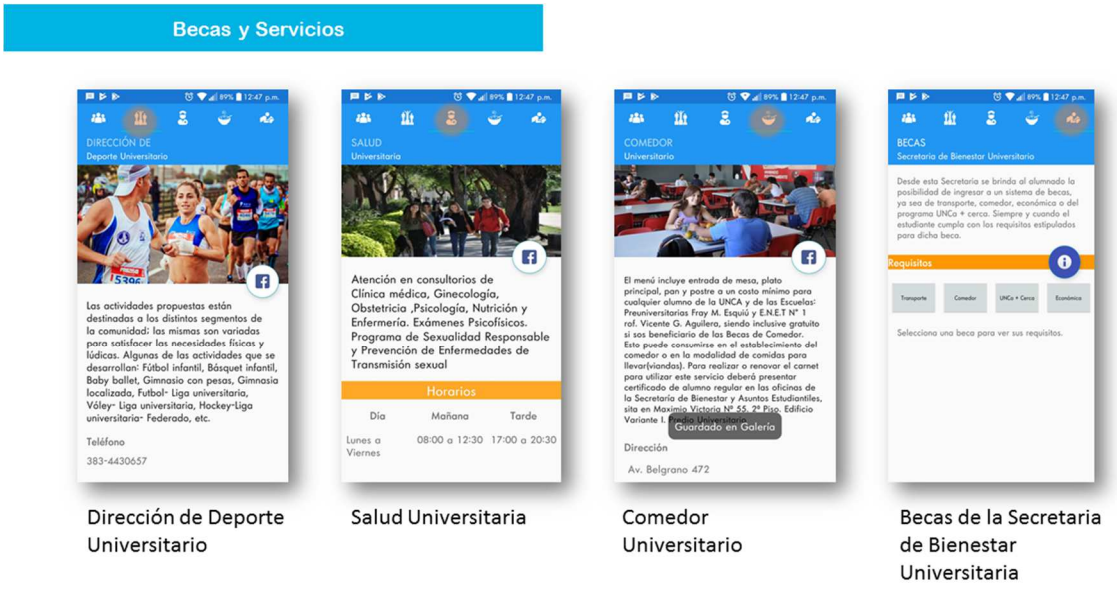
Conoce la UNCa

Se podrá observar la ubicación seleccionada.

Becas y Servicios

En la primera pantalla podremos ver información sobre la Secretaria de Bienestar Universitario.

Como así también iconos que nos conectan a sus redes sociales: Facebook e Instagram y un tercer ícono que nos permite enviar un email a la secretaria utilizando la aplicación de Gmail.



Dirección de Deporte Universitario

Salud Universitaria

Comedor Universitario

Becas de la Secretaria de Bienestar Universitaria

Referencias

[1] RAMIREZ VIQUE, Robert. "Desarrollo de aplicaciones basadas en Android". [En línea]. En: *Tecnología y desarrollo en dispositivos móviles*, Universitat Oberta de Catalunya, 2011. [Fecha de consulta: 14 de diciembre 2017]. Disponible en http://cort.as/-IchM

[2] MORILLO POZO, Julián David. "Introducción a los dispositivos móviles". [En línea]. En: *Tecnología y desarrollo en dispositivos móviles*, Universitat Oberta de Catalunya. [Fecha de consulta: 14 de diciembre 2017]. Año 2011. Disponible en http://cort.as/-IckJ

[3] ANDROID STUDIO. [Fecha de consulta: 9 diciembre 2017]. Disponible en https://developer.android.com/studio/index.html

[4] MORILLO POZI, Julián David. "Entornos de programación móviles". [En línea]. En: *Tecnología y desarrollo en dispositivos móviles*, Universitat Oberta de Catalunya. [Fecha de consulta: 14 de diciembre 2017]. Año 2011. Disponible en http://cort.as/-Icig

[5] GONZALEZ, Julio B. y LESSIRE, Omaira. "Aspectos más recientes en orientación vocacional". En: *Revista Iberoamericana de Educación.* [En línea]. [Fecha de consulta: 9 de diciembre 2017]. Año 2004 Disponible en: https://rieoei.org/historico/deloslectores/876Gonzalez.PDF

[6] KOTLIN LANGUAGE DOCUMENTATION. [En línea]. JetBrain. Disponible en: http://kotlinlang.org/docs/reference/

[7] CAMPOS, F. J. "Reingeniería de la Metodología en MIFISIS 2002". [En línea]. En: *I Workshop sobre Métodos de Investigación y Fundamentos Filosóficos en Ingeniería del Software y Sistemas de Información*. Universidad Rey Juan Carlos. Noviembre 18 de 2002. Citado por [8]

[8] JAIME, A.; CHAVARRIAGA, L.; HUGO, F.; ARBOLEADA, J. "Modelo de Investigación en Ingeniería del Software: Una propuesta de investigación tecnológica". [En línea]. Grupo LIDIS Universidad San Buenaventura, Cali, Colombia. Disponible en: http://cort.as/-IcmD

[9] GASCA MANTILLA, Maira Cecilia; CAMARGO ARIZA, Luis Leonardo y MEDINA DELGADO, Byron. "Metodología para el desarrollo de aplicaciones móviles". [En línea]. En: Tecnura, vol. 18, núm. 40, pág. 20-35. Universidad Distrital Francisco José de Caldas - Bogotá, Colombia. Año 2014. Disponible en: http://www.redalyc.org/articulo.oa?id=257030546003

[10] MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT. [En línea]. [Fecha de consulta: 17 de marzo de 2018] Disponible en: http://agilemanifesto.org/

[11] PÁGINA OFICIAL AGILE ALLIANCE. [En línea]. [Fecha de consulta: 17 de marzo de 2018]. Disponible en: https://www.agilealliance.org

[12] AHONEN, Tomi; BARRET, Joe; GOLDING, Paul. "Services for UMTS, creating killer applications in 3G". Año 2002. Editorial West Sussex: John Wiley & Sons. ISBN 10: 0471485500 / ISBN 13: 9780471485506.

[13] GÓMEZ CASTRO, Ricardo A.; GALVIS PANQUEVA, Álvaro H.; MARIÑO DREWS, Olga. "Ingeniería de software educativo con modelaje orientado por objetos: un medio para desarrollar micromundos interactivos". En: Informática Educativa Vol. 11. UNIANDES - LIDIE pág.9-30. Año 1998.

[14] MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT - ARGENTINA. [En línea]. [Fecha de consulta: 17 de marzo de 2018]. Disponible en: http://www.agiles.org/argentina

[15] CATALDI, Z., LAGE, F., PESSACQ, R. y GARCIA MARTINEZ, R. "Ingeniería de Software Educativo". [En línea]. Laboratorio de Sistemas Operativos y Bases de Datos. Departamento de Computación. Facultad de Ingeniería UBA - Facultad de Ingeniería. UNLP. Centro de Ingeniería del Software e Ingeniería del Conocimiento (CAPIS) ITBA. Laboratorio de Sistemas Inteligentes. Departamento de Computación. Facultad de Ingeniería UBA. Disponible en http://laboratorios.fi.uba.ar/lsi/c-icie99-ingenieriasoftwareeducativo.pdf

[16] RUBIER, F.; VALENCIA, O.; RIASCOS, M.; NIÑO, Z.; y MIGUEL, A. "Método para la creación de micro mundos inmersivos". [En línea]. Grupo de I+D en Tecnología de la Información. Departamento de Sistemas. Facultad de Ingeniería Electrónica y Telecomunicaciones. Universidad del Cauca, Colombia. Revista Avances en Sistemas e Informática, Vol. 8 N°2– Medellín – ISSN 1657-7663. Año 2011 Disponible en: https://revistas.unal.edu.co/index.php/avances/article/viewFile/26724/27033

[17] BRESLAY, Andrey. "Idiomas: JVM - Idioma del mes: Kotlin". [En línea]. Revista Dr. Dobb´s, The world of the development. Año 2012. Disponible en: http://www.drdobbs.com/jvm/language-of-the-month-kotlin/232600836

[18] GOETE, Emanuel. "Ceylon". [En Línea]. Año 2011. Disponible en: http://emanuelpeg.blogspot.com.ar/2011/08/ceylon.html

[19] HICKEY, Rick. "The Clojure Programming Language". Copyright 2008-2017. Disponible en: https://clojure.org/

[20] THE SCALA PROGRAMMING LANGUAGE. [En línea]. Copyright © 2002-2018. École Polytechnique Fédérale Lausanne (EPFL) Lausanne, Switzerland. Disponible en: http://www.scala-lang.org/

[21] HEISS, Janice J. "El advenimiento de Kotlin: una conversación con JetBrains 'Andrey Breslav". [En línea]. Oracle Technology Network. Abril, 2013. Disponible en: http://www.oracle.com/technetwork/articles/java/breslav-1932170.html

[22] MAHAPATRA, Lisa. “Android vs. iOS: ¿Cuál es el sistema operativo móvil más popular en su país?”. [En línea].En: Revista International Bussiness Times. Año 2013. Disponible en: http://www.ibtimes.com/android-vs-ios-whats-most-popular-mobile-operating-system-your-country-1464892

[23] PÁGINA OFICIAL APPLE. “iPad. Del 1 al 10, un 11”. [En línea]. Copyright © 2018 Apple Inc. Disponible en: https://www.apple.com/es/ios/ios-11/

[24] LOCKHEIMER, Hiroshi. “Android es para todos”. [En línea]. Responsable de Android, Chrome OS y Chromecast. Disponible en: https://www.android.com/everyone/

[25] “Noticias sobre AOSP”. [En línea]. El androide libre. [Fecha de Consulta: 21 de abril de 2018]. Disponible en: https://elandroidelibre.elespanol.com/tag/aosp

[26] DOCUMENTACIÓN ANDROID OS. [En línea]. “Arquitectura”. [Fecha de consulta: 21 de abril de 2018]. Disponible en: http://cort.as/-IcpC

[27] DOCUMENTACIÓN ANDROID OS. [En línea]. “Características”. [Fecha de consulta: 21 de abril de 2018] Disponible en: http://androidos.readthedocs.io/en/latest/data/caracteristicas/

[28] RAMIREZ VIQUE, Robert. "Métodos para el desarrollo de aplicaciones móviles". [En línea]. En: Tecnología y desarrollo en dispositivos móviles, Universitat Oberta de Catalunya, 2011. [Fecha de consulta: 23 de abril 2018]. Disponible en http://cort.as/-AM_1

[29] REINO ROMERO, Alfredo. “Introducción a XML en castellano”. [En línea]. Versión 2.0. Enero, 2000. Disponible en: http://www.cyta.com.ar/elearn/edita/material/xml_1.pdf

[30] SANTA CRUZ, Daniel. “Cómo funciona la nueva app para hacer un test vocacional”. [En línea]. [Fecha de Consulta: 20 noviembre de 2018]. Disponible en: https://www.lanacion.com.ar/2141140-como-funciona-la-nueva-app-para-hacer-un-test-vocacional. Copyright © LA NACION.

[31] DOMINGAME. “Test de Orientación Vocacional”. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.softF.Vocacional&hl=es_AR. Desarrollado por: DominGame. Contacto: osellafederico@gmail.com.

[32] MGBEMENA, Chike. “Kotlin Desde Cero: Variables, Tipos Básicos y Arreglos” - Kotlin From Scratch: Nullability, Loops, and Conditions. 11 agosto 2017. [En línea]. [Fecha de consulta: Consultado 24 de diciembre 2018]. Disponible en: https://code.tutsplus.com/series/kotlin-from-scratch--cms-1209

[33] WEB OFICIAL MAPBOX. [En línea]. [Fecha de Consulta: 25 de diciembre 2018]. Disponible en: https://www.mapbox.com/about/

[34] PARRA MENA, Yadira. “Las pruebas vocacionales”. Revista Educación y Ciencia Vol. 2 No. 6. Diciembre, 1992. [En línea]. Disponible en: http://www.educacionyciencia.org/index.php/educacionyciencia/article/view/53

[35] MIRA Y LÓPEZ, E. - "Manual de orientación profesional". Año 1965, Argentina. Editorial Kapelusz. Citado por [34].

[36] TYLER, L. – "La función del orientador" - Año 1979, México, DF. Editorial Trillas. Citado por [34].

[37] GALTON, Francis -" Hereditary genius: an inquiry into its laws and consequences" - Año 1896, Londres - Editorial: Macmillan. Citado por [34].

[38] QUATTROCCI, Paula Raquel; GARCÍA, Alejandra Elisa; SCHITTNER, J. Vanesa. "Aprender a hacer orientación usando TIC. Aprender a utilizar TIC para orientación". [En línea]. Universidad de Buenos Aires, Secretaría de Asuntos Académicos, Dirección Técnica Programa de Orientación al Estudiante (DOE) - Congreso Iberoamericano de Ciencia, Tecnología, Innovación y Educación. Noviembre, 2014. Disponible en: https://www.oei.es/historico/congreso2014/memoriactei/1117.pdf

[39] REPETTO, E. y otros. "Acreditación de Competencias de los Orientadores Profesionales en contextos no escolares: El Proyecto Europeo EAS (European Accreditation Sheme)". REOP, Vol. 20, núm.3, 3º Cuatrimestre, pág. 225-237

[40] SOBRADO FERNÁNDEZ, L.; CEINOS SANZ, C. y GARCÍA MURIAS, R. "Utilización de las TIC en orientación profesional: Experiencias innovadoras ". En Revista Mexicana de Orientación Educativa Volumen IX.

[41] BRUN, Mario. "Las tecnologías de la información y las comunicaciones en la formación inicial docente de América Latina. Santiago de Chile: CEPAL". [En línea]. Disponible en: http://cort.as/-IcrF

[42] ANDROID DEVELOPERS. "Como transmitir datos de red mediante Volley". [En línea] Disponible en: https://developer.android.com/training/volley

[43] GLASS, R. "The Software Research Crisis", IEEE Software, Noviembre 1994. Citado por [8]

[44] GALAN, F.J. CAÑETE, J.M. "¿Qué se entiende en España por Investigación en Ingeniería de Software?" En MIFISIS 2002. I Workshop sobre Métodos de Investigación y Fundamentos Filosóficos en Ingeniería del Software y Sistemas de Información. Universidad Rey Juan Carlos. Noviembre 18 de 2002. Citado por [8]

[45] MARCOS, E. "Investigación en Ingeniería de Software vs. Desarrollo de Software". En: MIFISIS 2002. I Workshop sobre Métodos de Investigación y Fundamentos Filosóficos en Ingeniería del Software y Sistemas de Información. Universidad Rey Juan Carlos. Noviembre 18 de 2002.Citado por [8].

[46] SHAW, M. "What makes Good Research in Software Engineering?". European Joint Conference on Theory and Practice of Software ETAPS 2002. Abril 2002. Citada por [8]

[47] SHAW, M. "Designing Good Research Projects in Software Engineering… and getting results accepted for publication". Carnegie Mellon University. Citado por [8]

[48] SHAW, M. "Writing Good Software Engineering Research Papers". Minitutorial. Internacional Conference on Software Engineering, ICSE 2003. Mayo 2003. Citado por [8]

[49] ZELKOWITZ, M. Wallace, D. "Experimental Models for Validating Technology". IEEE Computer. Mayo 1998, pág. 23-31. Citado por [8]

[50] ZELKOWITZ, M. Wallace, D. “Experimental Validation in Software Engineering”. Conference of Empirical Assessment & Evaluation in Software Engineering, Keele University. Marzo 1997. Citado por [8].

[51] GONZÁLEZ FIEGEHEN, L. E., y ESPINOZA DÍAZ, O. (2008). “Deserción en educación superior en América Latina y el Caribe”. [En línea]. En: Revista Paideia, vol. 45, pág.33-46. Disponible en: http://cort.as/-IePr

[52] SANCHEZ AMAYA, G., NAVARRO SALCEDO, W., y GARCÍA VALENCIA, A. (2009). “Factores de deserción estudiantil en la Universidad Surcolombiana”. [En línea]. En: Paideia Surcolombiana, pág. 97-103. Disponible en: http://cort.as/-IeQa

[53] CIANO, N., CASTIGNANI, M. L., y GARCÍA, M. N. “Exploración del abandono universitario en estudiantes de las carreras de la Facultad de Psicología de la Universidad Nacional de La Plata”. [En línea]. En: 3er Congreso Internacional de Investigación. (2011) Universidad Nacional de La Plata. Facultad de Psicología. Disponible en: http://cort.as/-IeRr

[54] SANTOYO, M. K. F., y PATIÑO, J. C. S. “Motivos de deserción de estudiantes de Licenciatura durante su primer año cursado, en el Instituto Tecnológico Superior de Irapuato. Factores asociados al abandono. Tipos y perfiles de abandono”. [En línea]. En: VII Conferencia Latinoamericana sobre el Abandono en la Educación Superior (CLABES). Año 2017. Disponible en: http://cort.as/-IeSa

[55] HUESCA RAMIREZ, M.G., y CASTAÑO CORVO, M. B. “Causas de deserción de alumnos de primeros semestres de una universidad privada”. [En línea]. En: Revista Mexicana de Orientación Educativa, vol. 5(12), pág. 34-39. Año 2007. Disponible en: http://cort.as/-In0Z

[56] ESCOBAR, Juan Diego. “Elección profesional y deserción universitaria. Del re direccionamiento del plan de vida”. En: Revista Electrónica Psyconex: Psicología, Psicoanálisis y Conexiones. Vol. 5 (8). Año 2013. ISSN 2145-437X. Disponible en: http://cort.as/-In0u

[57] INFANTE TAVÍO, N. I., MENDO ALCOLEA, N., & VAZQUEZ SANCHEZ, M. “Factores determinantes de la deserción escolar en el Policlínico Docente". En: Revista Médica de Santiago de Cuba: Medisan, Cuba. Año 2012. Disponible en: http://scielo.sld.cu/pdf/san/v16n4/san06412.pdf

[58] BETANCOURTH SANCHEZ, L. J. “Orientación vocacional y profesional en la juventud colombiana”. En: Tesis para optar al título de Especialista en Docencia Universitaria. Universidad Militar Nueva Granada. Colombia. Año 2016. Disponible en: http://cort.as/-InG2

[59] MARGAIN PÉREZ, A.K. y MURILLO GARCÍA, F. “La orientación vocacional como estrategia para favorecer el éxito escolar”. [En línea]. En: III Seminario del Sistema de Información de Estudiantes, Egresados y Empleadores (SIEEE). Año 2014 Disponible en: https://www.azc.uam.mx/sieee/cuartoseminario/ponencias/ponencia06.pdf

[60] DE LEÓN MENDOZA, T. y RODRIGUEZ MARTINEZ, R. “El efecto de la orientación vocacional en la elección de carrera”. [En línea]. En: Revista Mexicana de Orientación Educativa, vol. 5(13), pág. 10- 16. Año 2008. Disponible en: http://cort.as/-InIR

[61] ALBEZA MARTIN, M. A. y MENDOZA PONTIFFE, L. “Vocación y Permanencia Educativa”. [En línea]. En: IV Encuentro Nacional de Servicios de Orientación Universitaria “Hacia una Mirada Interdisciplinaria”. Año 2016. Disponible en: http://cort.as/-InJ9

[62] ANDROID – Página oficial [En línea] – Disponible en: https://www.android.com/versions/pie-9-0/#adaptive-technologies